National
Semiconductor

400070

Microcontroller
Databook

A Corporate Dedication to Quality and Reliability

National Semiconductor is an industry leader in the manufacture of high quality, high reliability integrated circuits. We have been the leading proponent of driving down IC defects and extending product lifetimes. From raw material through product design, manufacturing and shipping, our quality and reliability is second to none.

We are proud of our success . . . it sets a standard for others to achieve. Yet, our quest for perfection is ongoing so that you, our customer, can continue to rely on National Semiconductor Corporation to produce high quality products for your design systems.

Charlie Sporck

Charles E. Sporck
President, Chief Executive Officer
National Semiconductor Corporation

Wir fühlen uns zu Qualität und Zuverlässigkeit verpflichtet

National Semiconductor Corporation ist führend bei der Herstellung von integrierten Schaltungen hoher Qualität und hoher Zuverlässigkeit. National Semiconductor war schon immer Vorreiter, wenn es galt, die Zahl von IC Ausfällen zu verringern und die Lebensdauern von Produkten zu verbessern. Vom Rohmaterial über Entwurf und Herstellung bis zur Auslieferung, die Qualität und die Zuverlässigkeit der Produkte von National Semiconductor sind unübertroffen.

Wir sind stolz auf unseren Erfolg, der Standards setzt, die für andere erstrebenswert sind. Auch ihre Ansprüche steigen ständig. Sie als unser Kunde können sich auch weiterhin auf National Semiconductor verlassen.

La Qualité et La Fiabilité:

Une Vocation Commune Chez National Semiconductor Corporation

National Semiconductor Corporation est un des leaders industriels qui fabrique des circuits intégrés d'une très grande qualité et d'une fiabilité exceptionelle. National a été le premier à vouloir faire chuter le nombre de circuits intégrés défectueux et a augmenter la durée de vie des produits. Depuis les matières premières, en passant par la conception du produit sa fabrication et son expédition, partout la qualité et la fiabilité chez National sont sans équivalents.

Nous sommes fiers de notre succès et le standard ainsi défini devrait devenir l'objectif à atteindre par les autres sociétés. Et nous continuons à vouloir faire progresser notre recherche de la perfection; il en résulte que vous, qui êtes notre client, pouvez toujours faire confiance à National Semiconductor Corporation, en produisànt des systèmes d'une très grande qualité standard.

Un Impegno Societario di Qualità e Affidabilità

National Semiconductor Corporation è un'industria al vertice nella costruzione di circuiti integrati di altà qualità ed affidabilità. National è stata il principale promotore per l'abbattimento della difettosità dei circuiti integrati e per l'allungamento della vita dei prodotti. Dal materiale grezzo attraverso tutte le fasi di progettazione, costruzione e spedizione, la qualità e affidabilità National non è seconda a nessuno.

Noi siamo orgogliosi del nostro successo che fissa per gli altri un traguardo da raggiungere. Il nostro desiderio di perfezione è d'altra parte illimitato e pertanto tu, nostro cliente, puoi continuare ad affidarti a National Semiconductor Corporation per la produzione dei tuoi sistemi con elevati livelli di qualità.

Charlie Sporck

Charles E. Sporck
President, Chief Executive Officer
National Semiconductor Corporation

MICROCONTROLLER
DATABOOK

1989 Edition

TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

Abuseable™
Anadig™
ANS-R-TRAN™
APPS™
ASPECT™
Auto-Chem Deflasher™
BCP™
BI-FET™
BI-FET II™
BI-LINE™
BIPLAN™
BLC™
BLX™
Brite-Lite™
BTL™
CheckTrack™
CIM™
CIMBUS™
CLASIC™
Clock✓Chek™
COMBO™
COMBO I™
COMBO II™
COPS™ microcontrollers
Datachecker®
DENSPAK™
DIB™
Digitalker®
DISCERN™
DISTILL™
DNR®
DPVM™
ELSTAR™
Embedded System Processor™
E-Z-LINK™
FACT™
FAIRCAD™
Fairtech™
FAST®
5-Star Service™
GENIX™
GNX™
HAMR™
HandiScan™
HEX 3000™
HPC™
I^{3}L®
ICM™
INFOCHEX™
Integral ISE™
Intelisplay™
ISE™
ISE/06™
ISE/08™
ISE/16™
ISE32™
ISOPLANAR™
ISOPLANAR-Z™
KeyScan™
LMCMOS™
M^{2}CMOS™
Macrobus™
Macrocomponent™
MAXI-ROM®
Meat✓Chek™
MenuMaster™
Microbus™ data bus
MICRO-DAC™
µtalker™
Microtalker™
MICROWIRE™
MICROWIRE/PLUS™
MOLE™
MST™
Naked-8™
National®
National Semiconductor®
National Semiconductor Corp.®
NAX 800™
Nitride Plus™
Nitride Plus Oxide™
NML™
NOBUS™
NSC800™
NSCISE™
NSX-16™
NS-XC-16™
NTERCOM™
NURAM™
OXISS™
P^{2}CMOS™
PC Master™
Perfect Watch™
Pharma✓Chek™
PLAN™
PLANAR™
Plus-2™
Polycraft™
POSilink™
POSitalker™
Power + Control™
POWERplanar™
QUAD3000™
QUIKLOOK™
RAT™
RTX16™
SABR™
Script✓Chek™
SCX™
SERIES/800™
Series 900™
Series 3000™
Series 32000®
Shelf✓Chek™
SofChek™
SPIRE™
STAR™
Starlink™
STARPLEX™
Super-Block™
SuperChip™
SuperScript™
SYS32™
TapePak®
TDS™
TeleGate™
The National Anthem®
Time✓Chek™
TINA™
TLC™
Trapezoidal™
TRI-CODE™
TRI-POLY™
TRI-SAFE™
TRI-STATE®
TURBOTRANSCEIVER™
VIP™
VR32™
WATCHDOG™
XMOS™
XPU™
Z STAR™
883B/RETS™
883S/RETS™

IBM® is a registered trademark of International Business Machines Corporation.
PAL® is a registered trademark of and used under license from Monolithic Memories, Inc.
UNIX® is a registered trademark of AT&T Bell Laboratories.
DEC™, VAX™ and VMS™ are trademarks of Digital Equipment Corporation.
MS-DOS™ is a trademark of Microsoft Corporation.

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation 2900 Semiconductor Drive, P.O. Box 58090, Santa Clara, California 95052-8090 (408) 721-5000
TWX (910) 339-9240

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied, and National reserves the right, at any time without notice, to change said circuitry or specifications.

National Semiconductor

Microcontroller Introduction

Practical Solutions to Real Problems

Microcontrollers have always been driven by customer need rather than technological capability.

They were designed to meet specific needs with specific performance in specific applications with specific cost.

That also meant, however, that your choices were limited to what was available on the market—which meant possibly having to compromise your design objectives because you couldn't get exactly the microcontroller you needed.

No more.

Now you can get a microcontroller from National that spans a wide range of system solutions—to go almost anywhere your design imagination takes you.

Whether you need a low-cost 4-bit workhorse or a 16-bit 30 MHz powerhouse, whether you want ½ kbyte of ROM or over 64 kbytes, whether you're building a simple singing greeting card or a complex telecommunications network, we have a microcontroller for the job.

With on-board CPU, memory, internal logic, and I/Os, National microcontrollers are helping more and more designers lower system costs and shrink system size.

And as technology brings more peripheral functions onto the chip, including user-programmable memory, fast SRAM, timers, UARTs, comparators, A/D converters, and LAN interfaces, the microcontroller will become the cost-efficient choice for even such real-time "microprocessor" applications as laser printers, ISDN, and digital signal processing.

That's why National continues to lead the industry in the development of microcontroller technology.

That's why we have 8-bit and 16-bit controller cores.

That's why we're scaling our common M2CMOS™ process for submicron feature sizes, hypermegahertz frequencies, and unparalleled performance levels.

That's why we offer you "Hot-Line" applications support and a 24-hour-a-day digital information service.

That's why we offer you IBM®-PC and DEC™- VAX™-based development tools and high-level-language (C) compilers

And that's why we've committed the full resources of our company to provide you with the most complete, most reliable, most cost-effective systems solution for all your needs.

This databook is a reflection of that committment.

It will give you an overview of microcontrollers in general and of National's microcontrollers in particular.

It will help you evaluate your microcontroller options from both a business perspective and an engineering perspective.

It will help you make reasoned judgements about selecting the best microcontroller for your needs.

And it will show you what the microcontroller future holds in store for all of us.

If you'd like more information, or you'd like to find out how to put a microcontroller to work in your own application, just contact your local National Semiconductor Sales Office.

National Semiconductor

How to Select a Microcontroller

Microcontrollers have evolved far beyond their origins as control chips in calculators.

Today, microcontrollers can be the perfect solution for simplifying a wide range of designs. And for giving those designs a clear competitive advantage in the marketplace.

Whether used for simple logic replacement or as an integral part of a high performance system, a microcontroller can reduce system costs, shrink system size, and shorten system design cycles. And yet deliver performance often superior to "traditional" digital solutions.

Still, all microcontrollers are not created equal. And it's important to consider a number of factors before committing to a particular device:

1. Is the microcontroller optimized for your specific application in terms of speed, performance, features, and cost?
2. Is it code-efficient, and based on a true microcontroller architecture for the highest performance and efficiency?
3. Is it fabricated in the most advanced CMOS process technology, and is it fully scalable to maintain its performance edge in the future?
4. Is it supported by a comprehensive family of development tools that run on standard platforms such as the IBM-PC and DEC VAX?
5. Is it backed by a dedicated team of professionals who are available not only to provide expert training for new users, to get them on-line quickly and efficiently, but also to provide technical guidance for even the most experienced user?
6. Is it designed for the future, with the capability of expanding on-chip functionality.

If you answered "yes" to all these questions, then you already know that there's only one company with the product depth and technology capability to provide you with a microcontroller optimized for your specific application.

National Semiconductor.

You'll find National Microcontrollers in:

Laser Printers
Disc Controllers
Telecommunications Systems
Keyboards
Airplane Multiplex Systems
Car Radios
Engine Control Systems
Anti-Skid Brake Systems
Armaments
Factory Automation
Medical Equipment
Fuses
Scales
Refrigerators
Security Systems
Garage Door Openers
Camera Aperture Controls
Office Copiers
Cable TV Converters
Televisions
Video Recorders
Solar Heating Controls
Thermostats
Climate Control Systems
Intelligent Toys
Kitchen Timers

National Semiconductor

Why Select a National Microcontroller

National has created the most complete selection of 4-, 8-, and 16-bit microcontrollers of any company in the industry.

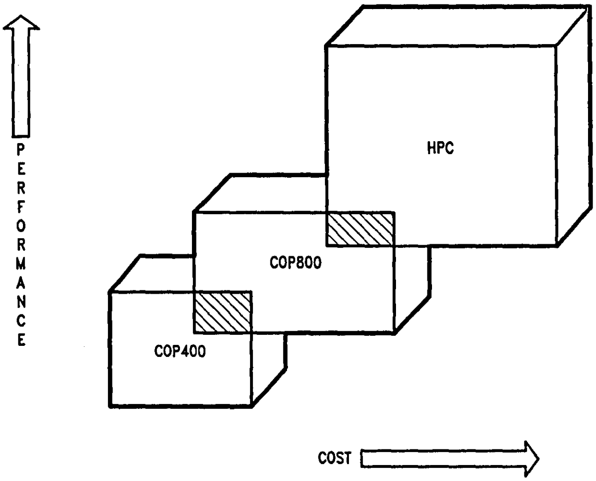
Which means that no matter what the specific needs of your application are, you can find a National microcontroller to meet them.

Our COP400 family offers the lowest-cost, 4-bit solutions for timing, counting, and control functions.

Our COP800 family offers low-cost, feature-rich, 8-bit solutions.

And our High Performance microController (HPC™) family offers the highest performance with the world's fastest 16-bit CMOS solution.

Microcontroller Family of Products



TL/XX/0071-1

With a full range of performance- and feature-options, National's microcontroller families can be customized to meet the needs of your specific application.

1.0 COMMON FEATURES FOR A CUSTOM FIT

All our microcontrollers are designed to provide not just a one-time-only solution, but a continuum of solutions to meet the changing demands of your product and the marketplace.

Our COP400 family, for example, which consists of over 60 devices, is designed with a common instruction set, so you can migrate from one member of the family to others without having to recode, so you can take efficient advantage of the application-specific flexibility of the COP400 family's programmable I/O options.

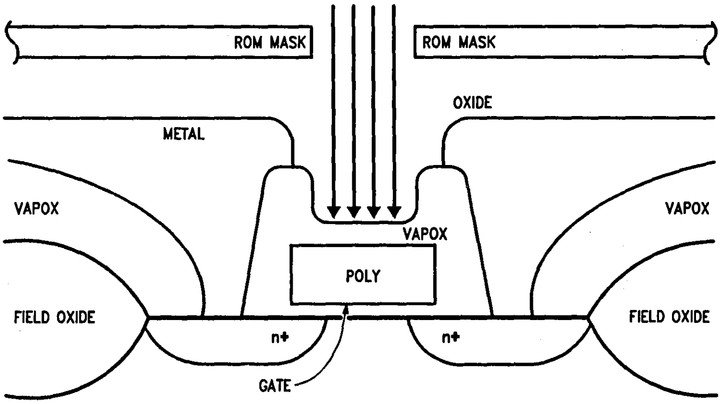
Our COP800 and HPC families, on the other hand, are each designed around a common CPU core that then can be surrounded by a variety of standard functional building blocks such as RAM, ROM, user programmable memory, fast SRAM, DMA, UART, comparator, A/D, HDLC, and I/O.

This unique core approach allows us to offer you a microcontroller with the exact combination of CPU power and peripheral function you need for your specific application. So you don't have to compromise your design parameters by using an inappropriate device, and you don't have to compromise your cost parameters by paying for performance and features you don't need.

This core concept also allows us to bring new microcontroller products to market fast and at a lower cost to help you keep pace with the rapidly changing conditions in your own market.

And it allows us to implement designs for both the COP800 and the HPC cores, for the highest levels of integration and flexibility in your own proprietary design.

COP400 Post-Metal ION Implant



TL/XX/0071-2

2.0 TRUE MICROCONTROLLER ARCHITECTURE

Our microcontrollers are designed as true controllers, not modified microprocessors.

The COP400 family is designed with a two-bus Harvard architecture; the COP800 family with a memory-mapped, modified Harvard architecture, and the HPC family with a memory-mapped, von Neumann architecture.

All three control-oriented families, however, are optimized for high code efficiency. Most instructions are only 1 byte long—yet each can typically execute several functions. This "function-dense" code provides a substantial increase in memory efficiency and processing speed.

3.0 ADVANCED PROCESS AND PACKAGING TECHNOLOGIES

National offers you not only the right microcontroller for your needs, but also the right process technology for your microcontroller.

COP400 devices are available in both high-speed NMOS and low-power CMOS fabrications, while the higher-performance COP800 and HPC families are both fabricated in National's advanced M^{2}CMOS process.

M^{2}CMOS. This double-metal CMOS process offers significant design advantages. It combines the speed of NMOS, the ruggedness of bipolar, and the low power consumption of bulk CMOS to produce fast, dense, highly efficient, highly scalable devices for a wide variety of integrated-circuit designs.

It's for these reasons that M^{2}CMOS has become the standard process technology for all of National's advanced-technology LSI and VLSI products, including microprocessors, gate arrays, standard cells, telecommunications devices, linear devices and, of course, microcontrollers.

Post-Metal Programming (PMP). This is a new process technology available from no other semiconductor manufacturer in the world. It offers the fastest, guaranteed prototype programmed-ROM turn-time in the industry.

PMP is a high-energy implantation process that allows microcontroller ROM to be programmed **after** final metallization.

This is a true innovation, because ROM is usually implemented in the second die layer, with nine or ten other layers then added on top. And that means the ROM pattern must be specified early in the production process, and completed prototype devices won't be available typically for six weeks.

With PMP, however, dice can be fully manufactured through metallization and electrical tests (only the passivation layers need to be added), and held in inventory. Which means ROM can be programmed late in the production cycle, **making prototypes available in only two weeks!**

And production parts can follow in as little as four weeks.

PMP allows you to adapt to fast-changing market conditions and to take maximum advantage of narrow windows of opportunity.

And shorter production lead times can simplify your inventory control and reduce safety stock by up to 20%, giving you significant cost reductions.

Currently, Post-Metal Programming is available for selected members of the COP400 family, and will be expanded to the COP800 and HPC familes in the near future.

Military versions. All National microcontrollers have CMOS parts available in the full military temperature range (−55°C to +125°C).

In addition, parts are available that have been certified under MIL-STD-883, Rev. C, the most rigorous non-JAN screening flow in the electronics industry.

Packaging. One major reason that National microcontrollers demonstrate such consistently high levels of reliability is that we've developed special advanced packaging processes to protect the die.

For example, we've designed a unique leadframe with "locking holes" that helps block any penetrating moisture from reaching the die itself.

And the leadframes themselves are made of an unusual high-strength copper alloy that has a lower thermal resistance (θ_{JA}) than typical Alloy 42-leadframes.

We've also employed a unique low-stress, high-purity epoxy molding compound for our packages, which gives them a coefficient of expansion that nearly matches that of the leadframes. As a result, many of our microcontrollers are also offered in plastic packages for military-temperature-range operation.

Reliability is built-in at the die level as well. Our M^{2}CMOS microcontrollers are fabricated on dedicated lines at our world-class, six-inch wafer-fab facility in Arlington, Texas. With its Class-10 clean rooms and automated-handling system, Arlington has set a standard of reliability equalled by few other companies in the industry.

And this reliability is available to you in a wide variety of microcontroller packages, ranging in size from 20 to 84 pins.

Package types include plastic and ceramic DIPs, small outline (S.O.) surface mounts, plastic and ceramic leaded chip carriers, and pin grid arrays.

Or, you can select the world's most advanced, high-density packaging option, TapePak™.

TapePak comines the advantages of an automated tape-and-reel-type delivery system with built-in testing pads for reliability and a unique plastic package carrier. The result is a surface-mounted package that can be as small as 1/10 the size of conventional surface mounts, with lead spacings of 20 mils.

4.0 FULL DEVELOPMENT SUPPORT

Even the right microcontroller, of course, is useless without the right development tool to put that controller to work in your application.

That's why National offers you a full range of development support. Ready-to-run evaluation boards. Emulators. Software. Prototyping devices. Training and seminars for beginning and advanced users. Everything you need to take your design from concept to reality.

And you don't need an expensive, dedicated, development environment to do it. With our development systems, a standard IBM PC or DEC VAX becomes a full-featured platform.

And with our comprehensive library of prewritten routines, from keyboard scanners to Fast Fourier Transforms, you can reduce software programming to a minimum. This "user-friendly" service can help you bring your design to market quickly and cost-effectively.

5.0 FULL APPLICATIONS SUPPORT

At National, we believe that applications support should be immediate and "hands-on".

That's why we established the unique Dial-A-Helper program.

With a computer, modem, and telephone, you can tie directly into our Microcontroller Applications Group for fast, direct assistance in developing your design.

You can leave messages on our electronic bulletin board for our Applications Engineers, who will respond to you directly.

You can access applications files.

You can download those files for later reference.

Or, if you're having a real problem, you can actually turn the control of your Microcontroller On-Line Emulator development system over to our engineering staff, who can perform remote diagnostic routines to locate and eliminate any bugs.

The point is, when you buy a microcontroller from National, you're buying more than silicon—you're buying the commitment of an entire company of dedicated professionals who share a single goal: to help you put that silicon to **work**.

6.0 THE FUTURE

National's microcontrollers were designed to meet two objectives: to adapt to your evolving needs, and to adapt to evolving technology.

Both "evolutions," however, are leading to the same goal: the complete "system-on-chip" solution.

The key to achieving this goal, of course, is a common, advanced, scalable process technology.

That's why both the COP800 and HPC families are fabricated in our high-performance double-metal CMOS process. This is a highly scalable technology that can accommodate die shrinks to submicron feature sizes, increasing performance and cutting power consumption with each step.

Moreover, because M^{2}CMOS is now the standard process technology for all new National LSI and VLSI devices, the COP800 and HPC cores are able to support one of the broadest range of functional blocks available from any semiconductor manufacturer—all aligned on the same set of design rules.

So you can standardize your designs on just one or two core processors, and, as we introduce new technologies and functions, you can maintain that design knowledge base while taking advantage of these new, higher levels of functional integration.

And because National gives you the option of using standard parts or designing with our functional blocks—both supported by common design tools and a common process—you can create highly competitive, highly secure, highly optimized solutions in minimal space at minimal cost in minimal time.

And that's the name of the game.

National Semiconductor

Product Status Definitions

Definition of Terms

Data Sheet Identification	Product Status	Definition
Advance Information	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	This data sheet contains preliminary data, and supplementary data will be published at a later date. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. National Semiconductor Corporation reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

National Semiconductor Corporation reserves the right to make changes without further notice to any products herein to improve reliability, function or design. National does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others.

Table of Contents

Table of Contents (Continued)

Section 3 COPS Applications

Section 4 HPC Family

Section 5 HPC Applications

Table of Contents (Continued)

Section 5 HPC Applications (Continued)

Section 6 MICROWIRE and MICROWIRE/PLUS Peripherals

Section 7 Microcontroller Development Tools

Section 8 Appendices/Physical Dimensions

Alpha-Numeric Index

Alpha-Numeric Index (Continued)

Alpha-Numeric Index (Continued)

Alpha-Numeric Index (Continued)

Section 1
COP400 Family

Section 1 Contents

National Semiconductor

The 4-Bit COP400 Family: Optimized for Low-Cost Control

National's COP400 family offers the broadest range of low-priced, 4-bit microcontrollers on the market.

Key Features

- High-performance 4-bit microcontroller
- 4 μs–16 μs instruction-cycle time
- ROM-efficient instruction set
- On-chip ROM from 0.5k to 2k
- On-chip RAM from 32 x 4 to 160 x 4
- More than 60 compatible devices in family
- Common pin-outs
- NMOS and P²CMOS™
- MICROWIRE™ serial interface
- Wide operating voltage range: +2.4V to +6.3V
- Military temp range available: −55°C to +125°C
- 20- to 28-pin packages (incl. 20-, 24-pin SO and 28-pin PLCC)

And far from being "old technology," 4-bit microcontrollers are meeting significant market needs in more applications than ever before. In fact, National shipped more than 40 million 4-bit devices last year alone. The reason for the continuing strength of the COP400 family is its versatility. You can select from over 60 different, compatible devices. You can select devices with unit costs **below 50 cents**—the lowest-priced microcontrollers in the world. You can select devices with a wide variety of ROM and RAM combinations, from 0.5k ROM and 32 x 4 RAM to 2k ROM and 160 x 4 RAM.

And every COP400 family member shares the same powerful, ROM-efficient instruction set and the same pin-out, so you can migrate between devices without re-engineering.

And like all of National's microcontrollers, the COP400 can be optimized to meet your specific application needs, with a variety of I/O options, pin-outs, and package types, from DIPs to SMDs.

COPS™ microcontrollers can be used to replace discrete logic in high-volume consumer products and low-volume industrial products allowing you to add features, miniaturize and reduce component count.

Key Applications

- Consumer electronics
- Automotive
- Industrial control
- Toys/games
- Telephones

Wide Acceptance

COPS wide acceptance comes from innovative products. National has built on this established family with continued and enhanced devices.

- The first under-a-dollar microcontroller led to a broader range of automotive and consumer applications.
- The first high-speed, low-power CMOS microcontrollers with 0.5k ROM provides design flexibility at low cost.
- The first microcontroller implementing MICROWIRE/PLUS™ allowing two-way communication across only three lines.
- The first under $.50 microcontroller providing excellent cost/performance benefits for applications impossible before.
- The first microcontroller implementing Post-Metal Programming (PMP™) for quick turns prototyping and production.

PMP

Post-Metal Programming (PMP), another NSC microcontroller first. Takes advantage of:

- Seasonal or volatile market demand
- Narrow windows of opportunity in highly competitive markets
- Simplified inventory control
- Reduced safety stock

Get all the advantages of custom-programmed microcontrollers with all the business advantages of low cost, quick-turn prototyping and production.

The secret is an entirely new process technology called Post-Metal Programming.

PMP (Continued)

INSIDE PMP

Post-Metal Programming is a high energy implantation process that allows the ROM layer of a microcontroller to be programmed after final metallization. That means every die layer can be fully fabricated, except for the passivation layers, and held in inventory. Then when you request a ROM pattern, a ROM implant mask is generated and the buried ROM layer is programmed with an ion beam.

The wafer is passivated and cut into dice which are then packaged on a quick-turn line.

So in only two weeks, you've got prototypes.

4-WEEK PRODUCTION QUANTITIES

Wafer fab accounts for the majority of prototyping and production time for integrated circuits.

With PMP, however, the dice are essentially complete and in inventory.

So we can take your approved prototypes right into full production in as little as four weeks.

WINNING THE TIME-TO-MARKET RACE

The electronics market won't wait for anyone. If your competitors make a move, you've got to respond now.

You can't wait around for proof-of-design prototypes. Even a week can make a difference between success or failure. Between gaining market share or losing it. Between staying ahead of the other guys or falling behind. With PMP, you can stretch that lead by *weeks*. In fact, if you compare the quick-turn PMP process to conventional prototype-and-production timetables, you'll see that *you can actually gain as much as 3½ months over your competitors!*

NO EXTRA COST

PMP is available at *no extra cost.*

Compare that with the traditional "alternative" for quick-turn prototyping of user-programmable ROM. EPROM and EEPROM can easily drive your unit costs up to as much as $6!

And when you consider the additional cost-savings of being able to reduce your safety stock in inventory, knowing you can get quick-turns in a few weeks, the PMP process and National Semiconductor microcontrollers not only make good *engineering* sense, they make good *business* sense.

System Solutions

The COP400 family provides a flexible, cost-effective system solutions to all applications requiring timing, counting, or control functions.

And, bottom line, if a 4-bit controller can do the job, why pay more?

Development Support

DEVELOPMENT SYSTEM

The Microcomputer On-Line Emulator Development System is a low cost development system and emulator for COPs microcontroller products. The Development System consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the COP400 Development System is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem or to connect to other Development Systems in a multi-Development System environment.

The Development System can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

See AN-456 for more information.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP400	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COPS-PB1	Personality Board	COP400 Personality Board Users Manual	420408189-001
	MOLE-COPS-IBM	Assembler Software for IBM	COP400 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424409497-002 420040416-001
	424410284-001	Programmers Manual		424410284-001

COP400 Family of Microcontrollers

				Description		Features									Development Tools		
				Memory		I/O											
Commercial Temp Version 0°C to +70°C	Industrial Temp Version −40°C to +85°C	Military Temp Version −55°C to +125°C	Technology	ROM (Bytes)	RAM (Digits)	I/O Pins	Serial I/O	Interrupt	Stack	Time Base Counter	Micro Bus	Typ. 5V Operat. Power	Max Standby at 3.3V	Size (Pins)	ROMless Device	Piggyback	Data Sheet Page
COP413L*	COP313L		NMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401L-X13/R13		1-73
COP414L*	COP314L		NMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401LN		1-100
COP410L	COP310L		NMOS Low Power	0.5k	32	19	Yes	No	2 Level	No	No	15 mW	7.5 mW	24	COP401LN		1-52
COP411L	COP311L		NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	15 mW	7.5 mW	20	COP401LN		1-52
COP413C	COP313C		CMOS Low Power	0.5k	32	15	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-86
COP413CH	COP313CH		CMOS Hi Speed	0.5k	32	15	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-86
COP410C	COP310C	COP210C (Note 1)	CMOS Hi Speed	0.5k	32	19	Yes	No	2 Level	No	No	1 mW	0.1 mW	24	COP404CN	COP444CP	1-37
COP411C	COP311C	COP211C (Note 1)	CMOS Hi Speed	0.5k	32	16	Yes	No	2 Level	No	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-37
COP420	COP320		NMOS Hi Speed	1.0k	64	23	Yes	1 Source	3 Level	Yes	No	100 mW	N/A mW	28	COP402N	COP420P	1-115
COP421	COP321		NMOS Hi Speed	1.0k	64	19	Yes	No	3 Level	Yes	No	100 mW	N/A mW	24	COP402N	COP420P	1-115
COP422	COP322		NMOS Hi Speed	1.0k	64	16	Yes	No	3 Level	Yes	No	100 mW	N/A mW	20	COP402N	COP420P	1-115
COP424C*	COP324C	COP224C (Note 2)	CMOS Hi Speed	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	28	COP404CN	COP444CP	1-166
COP425C*	COP325C	COP225C (Note 2)	CMOS Hi Speed	1.0k	64	19	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	24	COP404CN	COP444CP	1-166
COP426C*	COP326C	COP226C (Note 2)	CMOS Hi Speed	1.0k	64	16	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	20	COP404CN	COP444CP	1-166
COP420L*	COP320L		NMOS Low Power	1.0k	64	23	Yes	1 Source	3 Level	Yes	Yes	45 mW	9.9 mW	28	COP404LSN-5	COP444LP	1-139
COP421L*	COP321L		NMOS Low Power	1.0k	64	19	Yes	No	3 Level	Yes	No	45 mW	9.9 mW	24	COP404LSN-5	COP444LP	1-139
COP422L*	COP322L		NMOS Low Power	1.0k	64	16	Yes	No	3 Level	Yes	No	45 mW	9.9 mW	20	COP404LSN-5	COP444LP	1-139
COP440	COP340		NMOS Hi Speed	2.0k	160	35	Yes	4 Sources	4 Level	Yes	Yes	205 mW	9.9 mW	40	COP404N	COP440R	1-186
COP441	COP341		NMOS Hi Speed	2.0k	160	23	Yes	4 Sources	4 Level	Yes	Yes	205 mW	9.9 mW	28	COP404N	COP440R	1-186
COP442	COP342		NMOS Hi Speed	2.0k	160	19	Yes	2 Sources	2 Level	Yes	No	205 mW	9.9 mW	24	COP404N	COP440R	1-186
COP444C*	COP344C	COP244C (Note 2)	CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	28	COP404CN	COP444CP	1-166
COP445C*	COP345C	COP245C (Note 2)	CMOS Hi Speed	2.0k	128	19	Yes	No	3 Level	Yes	No	1 mW	0.1 mW	24	COP404CN	COP444CP	1-166
COP444L	COP344L		NMOS Low Power	2.0k	128	23	Yes	1 Source	3 Level	Yes	No	65 mW	9.9 mW	28	COP404LSN-6	COP444LP	1-209
COP445L	COP345L		NMOS Low Power	2.0k	128	19	Yes	No	3 Level	Yes	No	65 mW	9.9 mW	24	COP404LSN-6	COP444LP	1-209

Note 1: Datasheet found on page 1-8.

Note 2: Datasheet found on page 1-20.

*Microcontrollers available with Quick-Turns Post-Metal Programming (PMP).

COPS Family Development Tools

Commercial Temp Version 0°C to +70°C			Technology	Description		Features									Supplementary Description	Data Sheet Page
				Memory		I/O		Interrupt	Stack	Time Base Counter	Micro Bus	Typ. 5V Operat. Power	Max Standby at 3.3V	Size (Pins)		
				ROM (Bytes)	RAM (Digits)	I/O Pins	Serial I/O									
ROMless																
COP401L-X13			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	Has XTAL Oscillator Option	1-247
COP401L-R13			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	Has RC Oscillator Option	1-247
COP401L			NMOS Low Power	0.5k	32	16	Yes	No	2 Level	No	No	100 mW	7.5 mW	40	ROMless Version of COP410L	1-233
COP402			NMOS Hi Speed	1.0k	63	20	Yes	1 Source	3 Level	Yes	No	50 mW	N/A mW	40	Has Interrupt, No Microbus	1-260
COP404LSN-5			NMOS Low Power	1.0k	128	20	Yes	1 Source	3 Level	Yes	No	125 mW	N/A mW	40	W/Push-Pull Mem Interface	1-302
COP404			NMOS Hi Speed	2.0k	160	23	Yes	4 Sources	4 Level	Yes	Yes	35 mW	15 mW	48	ROMless Version of COP440	1-278
COP404C			CMOS Hi Speed	2.0k	128	23	Yes	1 Source	3 Level	Yes	Yes	1 mW	0.1 mW	48	CMOS ROMless Device	1-285
PIGGYBACK																
COP420P			NMOS Hi Speed	1.0k	64	23	Yes	3 Sources	3 Level	Yes	No	50 mW	N/A mW	28	Includes: CPU, RAM, I/O	1-316
COP444LP			NMOS Low Power	2.0k	128	23	Yes	3 Sources	3 Level	Yes	No	125 mW	N/A mW	28	and EPROM Socket	1-316
COP444CP			CMOS Hi Speed	2.0k	128	23	Yes	1 Source	1 Level	Yes	Yes	1 mW	1 mW	28	Will Accept Standard EPROM	1-316

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

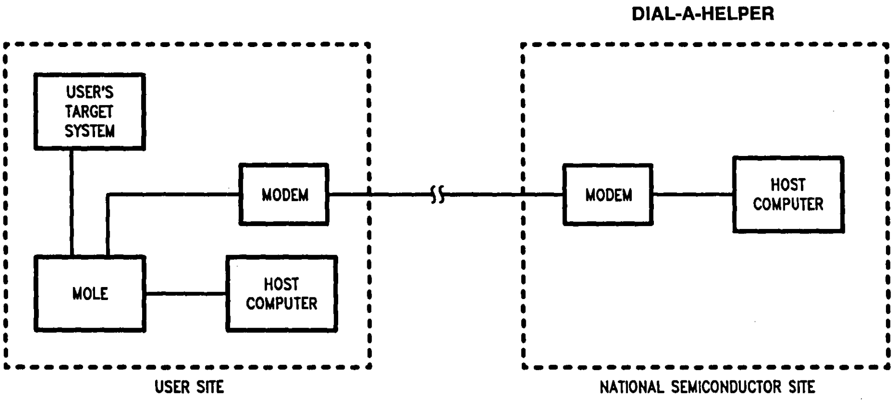
Information System Package Contains

DIAL-A-HELPER Users Manual P/N
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 baud	
	Set-Up:	Length:	8-bit
		Parity:	None
		Stop bit:	1
	Operation:	24 hrs., 7 days	



TL/XX/0072-1

National Semiconductor

COP210C/COP211C Single-Chip CMOS Microcontrollers

General Description

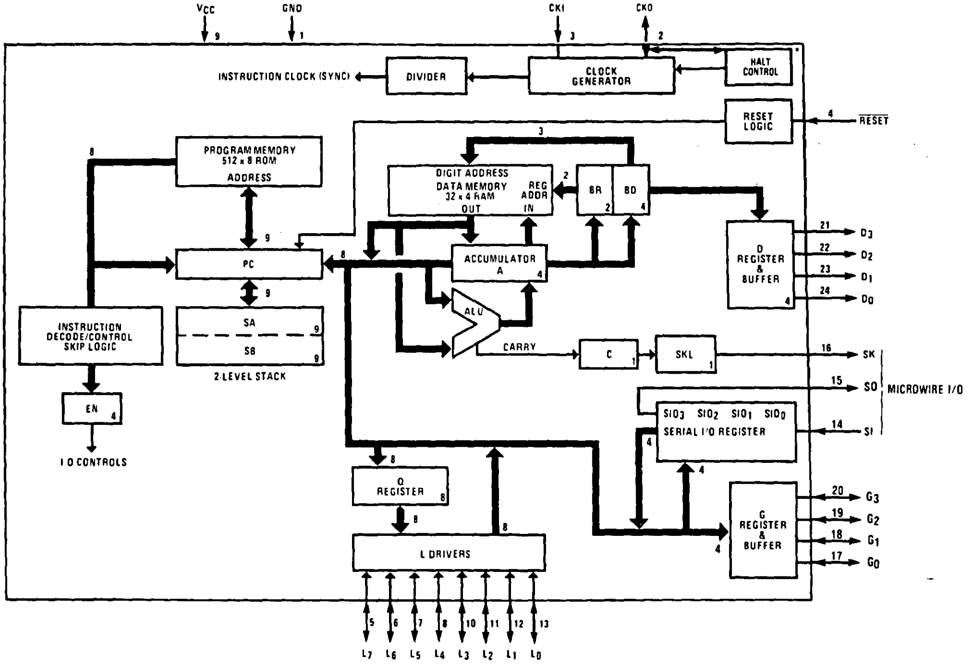
The COP210C and COP211C fully static, single-chip CMOS microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP211C is identical to the COP210C but with 16 I/O lines instead of 20. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller-oriented processor at a low end-product cost.

The COP404C should be used for exact emulation.

Features

- Lowest power dissipation (500 μW typical)
- Low cost
- Power-saving HALT mode with Continue function
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 20 I/O lines (COP210C)
- Two-level subroutine stack
- DC to 4.4 μs instruction time
- Single supply operation (4.5V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of the COP400 family
- Military temperature (−55°C to +125°C) devices

Block Diagram



TL/DD/8444-1

FIGURE 1. COP210C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Maximum Allowable Voltage	$V_{CC} = 6V$
Voltage at Any Pin	$-0.3V$ to $V_{CC}+0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Maximum Allowable Power Consumption	150 mW
Operating Temperature Range	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-55°C \le T_A \le +125°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Supply Current (Note 1)	$V_{CC} = 5.0V$, t_c = Min (t_c is instruction cycle time)		4	mA
Power Supply Ripple (Notes 3, 4)	Peak to Peak		0.25	V
HALT Mode Current (Note 2)	$V_{CC} = 5.0V$, $F_{IN} = 0$ kHz		120	μA
Input Voltage Levels				
$\overline{\text{RESET}}$, CKI				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		−10	+10	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels	Standard Outputs (except CKO)			
LSTTL Operation	$V_{CC} = 5.0V \pm 10\%$			
Logic High	$I_{OH} = -100\ \mu A$	2.7		V
Logic Low	$I_{OL} = 400\ \mu A$		0.6	V
CMOS Operation				
Logic High	$I_{OH} = -10\ \mu A$	V_{CC}-0.2		V
Logic Low	$I_{OL} = 10\ \mu A$		0.2	V
Allowable Sink/Source Current per Pin (Note 5)			5	mA
CKO Current Levels (As Clock Out)				
Sink ÷4	CKI = V_{CC}, $V_{OUT} = V_{CC}$	0.2		mA
÷8		0.4		mA
÷16		0.8		mA
Source ÷4	CKI = 0V, V_{OUT} = 0V	−0.2		mA
÷8		−0.4		mA
÷16		−0.8		mA
Allowable Loading on CKO (as HALT I/O pin)			50	pF
Current Needed to Override HALT (Note 6)				
To Continue	$V_{IN} = 0.2\ V_{CC}$		2.0	mA
To Halt	$V_{IN} = 0.7\ V_{CC}$		3.0	mA
TRI-STATE or Open Drain Leakage Current		−10	+10	μA

Note 1: Supply Current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}. L lines in TRI-STATE mode and tied to ground, all other outputs low and tied to ground.

Note 3: Voltage change must be less than 0.25V in a 1 ms period.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: SO Output sink current must be limited to keep V_{OL} less than 0.2 V_{CC}.

Note 6: When forcing HALT, current is only needed for a short time (approximatey 200 ns) to flip the HALT flip-flop.

COP210C/COP211C

AC Electrical Characteristics $-55°C \leq T_A \leq +125°C$ unless otherwise specified

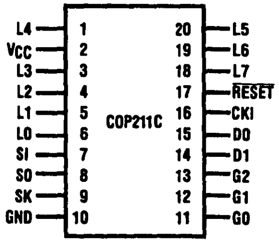
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)		4.4	DC	μs
Operating CKI Frequency ÷4 mode ÷8 mode ÷16 mode		DC DC DC	0.9 1.8 3.6	MHz MHz MHz

Instruction Cycle Time RC Oscillator (Note 4)	R = 30k ±5%
	C = 82 pF ±5% (÷4 Mode)
		6	18	μs

Inputs (See *Figure 3*) t_{SETUP} (Note 4)	G Inputs, SI Input, All Others } $V_{CC} \geq 4.5V$ G Inputs SI Input All Others	 tc/4 + 0.8 0.33 1.9		 μs μs μs
t_{HOLD}		0.40		μs
Output Propagation Delay t_{PD1}, t_{PD0}	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5k$		1.4	μs

Connection Diagrams

S.O. Wide and DIP



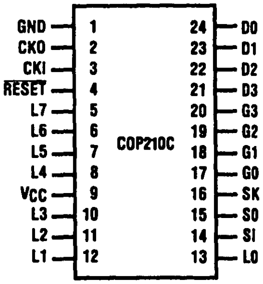
TL/DD/8444-2

Order Number COP211C-XXX/D, See NS Hermetic Package Number D20A

Order Number COP211C-XXX/N, See NS Molded Package Number N20A

Order Number COP211C-XXX/WM See NS Surface Mount Package Number M20B

S.O. Wide and DIP



TL/DD/8444-3

Order Number COP210C-XXX/D, See NS Hermetic Package Number D24C

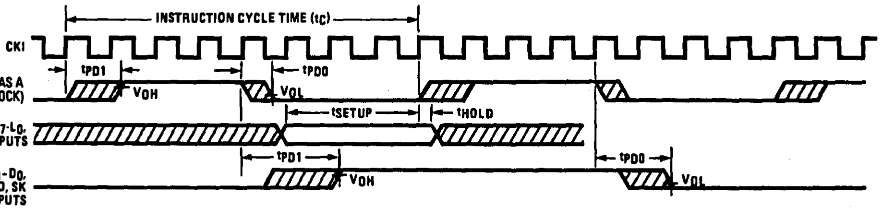
Order Number COP210C-XXX/N, See NS Molded Package Number N24A

Order Number COP210C-XXX/WM See NS Surface Mount Package Number M24B

Pin Descriptions

Pin	Description	Pin	Description
L_7–L_0	8-bit bidirectional I/O port with TRI-STATE	SK	Logic-controlled clock (or general purpose output)
G_3–G_0	4-bit bidirectional I/O port (G_2–G_0 for 20-pin package)	CKI	System oscillator input
D_3–D_0	4-bit general purpose output port (D_1–D_0 for 20-pin package)	CKO	Crystal oscillator output, or HALT mode I/O port (24-pin package only)
SI	Serial input (or counter input)	$\overline{RESET}$	System reset input
SO	Serial output (or general purpose output)	V_{CC}	System power supply
		GND	System Ground

FIGURE 2



INSTRUCTION CYCLE TIME (tc)
CKI
SK (AS A CLOCK)
tPD1
tPD0
VOH
VOL
tSETUP
tHOLD
G3-G0, L7-L0, & SI INPUTS
tPD1
tPD0
VOH
VOL
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS

TL/DD/8444-4

FIGURE 3. Input/Output Timing Diagrams (Divide-by-8 Mode)

Functional Description

A block diagram of the COP210C is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP210C/211C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8 x 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

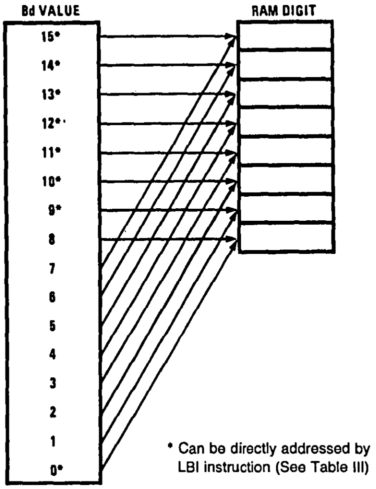
The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4.* The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP210C/211C is designed to ensure fully static operation of the device.

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP210C/211C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)



* Can be directly addressed by LBI instruction (See Table III)

TL/DD/8444-5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP210C/211C is MICROWIRE compatible.

The D register provides four general purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.

Functional Description (Continued)

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

2. EN1 is not used, it has *no* effect on the COP210C/211C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

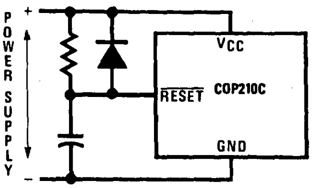
INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

When V_{CC} power is applied, the internal reset logic will keep the chip in initialization mode for up to 2500 instruction cycles. If the CKI clock is running at a low frequency, this could take a long time, therefore, the internal logic should be disabled by a mask option with initialization controlled solely by $\overline{\text{RESET}}$ pin.

Note: If CKI clock is less than 32 kHz, the internal reset logic (Option 25 = 1) *must* be disabled and the external RC network *must* be present.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/8444-6

RC > 5 × Power Supply Rise Time and RC > 100 × CKI Period

FIGURE 5. Power-Up Clear Circuit

COP211C

If the COP210C is bonded as a 20-pin package, it becomes the COP211C, illustrated in *Figure 2,* COP210C/211C Connection Diagrams. Note that the COP211C does not contain D2, D3, G3, or CKO. Use of this option, of course, precludes use of D2, D3, G3, and CKO options. All other options are available for the COP211C.

HALT MODE

The COP210C/211C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip also may be halted by the HALT instruction or by forcing CKO high when it is used as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode has slight differences depending upon the type of oscillator used.

a. 1-pin oscillator—RC or external

The HALT mode may be entered into by either program control (HALT instruction) or by forcing CKO to a logic "1" state.

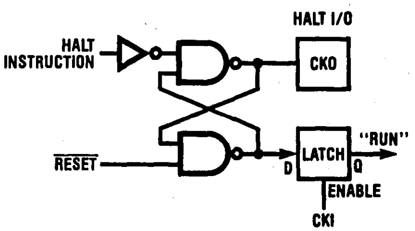
The circuit may be awakened by one of two different methods:

1) Continue function. By forcing CKO to a logic "0", the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.

2) Restart. Forcing the $\overline{\text{RESET}}$ pin to a logic "0" will restart the chip regardless of HALT or CKO (see initialization).

b. 2-pin oscillator—crystal

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.



TL/DD/8444-7

Halt I/O Port

CKO PIN OPTIONS

In a crystal-controlled oscillator system, CKO is used as an output to the crystal network. CKO will be forced high during the execution of a HALT instruction, thus inhibiting the crystal network. If a 1-pin oscillator system is chosen (RC or

Functional Description (Continued)

external), CKO will be selected as HALT and is an I/O flip-flop which is an indicator of the HALT status. An external signal can override this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and the CKO output will go high to keep the chip stopped. By forcing a low level to CKO, the chip will continue and CKO output will go low.

All features associated with the CKO I/O pin are available with the 24-pin package only.

OSCILLATOR OPTIONS

There are three options available that define the use of CKI and CKO.

a. Crystal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optionally by 8 or 4).

b. External Oscillator. CKI is configured as LSTTL-compatible input accepting an external clock signal. The external frequency is divided by 16 (optionally by 8 or 4) to give the instruction cycle time. CKO is the HALT I/O port.

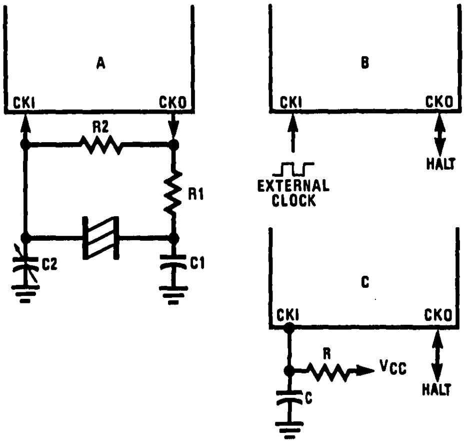
c. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.

COP210C/COP211C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP210C/211C instruction set.



TL/DD/8444-8

FIGURE 6. COP210C Oscillator

Crystal or Resonator					RC-Controller Oscillator		
Crystal Value	Component Values						Cycle Time
	R1	R2	C1pF	C2pF	R	C	
32 kHz	220k	20M	30	5–36	47k	100 pF	17–25 μs
455 kHz	5k	10M	80	40	30k	82 pF	6–18 μs
3.58 MHz	1k	1M	30	6–36	Note: 15k≤R≤150k, 50 pF≤C≤150 pF		

TABLE II. COP210C/211C Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\bar{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

Instruction Set (Continued)

TABLE III. COP210C/211C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	0101 y	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– –	0110 000 a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	–	1 $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 1)
			or			
		–	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	10 $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 2)
JSR	a	6– –	0110 100 a_8 $a_{7:0}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	0011 0011 0011 1000		None	Halt processor

Instruction Set (Continued)

TABLE III. COP210C/211C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 | 0011 0011 | 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 | 0011 0010 | 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	−5	00 | r | 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	1011 | 1111	ROM(PC_8,A,M) → Q SA → SB	None	Load Q Indirect
RMB	0	4C	0100 | 1100	0 → RAM$(B)_0$	None	Reset RAM Bit
	1	45	0100 | 0101	0 → RAM$(B)_1$		
	2	42	0100 | 0010	0 → RAM$(B)_2$		
	3	43	0100 | 0011	0 → RAM$(B)_3$		
SMB	0	4D	0100 | 1101	1 → RAM$(B)_0$	None	Set RAM Bit
	1	47	0100 | 0111	1 → RAM$(B)_1$		
	2	46	0100 | 0110	1 → RAM$(B)_2$		
	3	4B	0100 | 1011	1 → RAM$(B)_3$		
STII	y	7−	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	−6	00 | r | 0110	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	0010 | 0011 1011 | 1111	RAM(3,15) ←→ A	None	Exchange A with RAM (3,15)
XDS	r	−7	00 | r | 0111	RAM(B) ←→ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	−4	00 | r | 0100	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	−	00 | r | (d - 1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6−	0011 | 0011 0110 | y	y → EN	None	Load EN Immediate

Instruction Set (Continued)

TABLE III. COP210C/211C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		$G_{3:0}$ = 0	Skip if G is Zero
		21	0010 0001			(all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	G_0 = 0	
	1	11	0001 0001		G_1 = 0	
	2	03	0000 0011		G_2 = 0	
	3	13	0010 0011		G_3 = 0	
SKMBZ	0	01	0000 0001		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1$ = 0	
	2	03	0000 0011		$RAM(B)_2$ = 0	
	3	13	0001 0011		$RAM(B)_3$ = 0	
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 1110	$L_{3:0}$ → A		
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP210C/211C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register). If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

Description of Selected Instructions (Continued)

INSTRUCTION SET NOTES

a. The first word of a COP210C/211C program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).

c. The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw additional current. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP210C current drain.

$$Ic = Iq + (V \times 35 \times Fi) + (V \times 2195 \times Fi/Dv)$$

where Ic = chip current drain in microamps

Iq = quiescent leakage current (from curve)

Fi = CKI frequency in megahertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 4),

$$Ic = 10 + (5 \times 35 \times 0.4) + (5 \times 2195 \times 0.4/4)$$

$$Ic = 10 + 50 + 1097.5 = 1157.5\ \mu A$$

I/O OPTIONS

COP210C/211C outputs have the following optional configurations, illustrated in *Figure 7*:

a. Standard. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

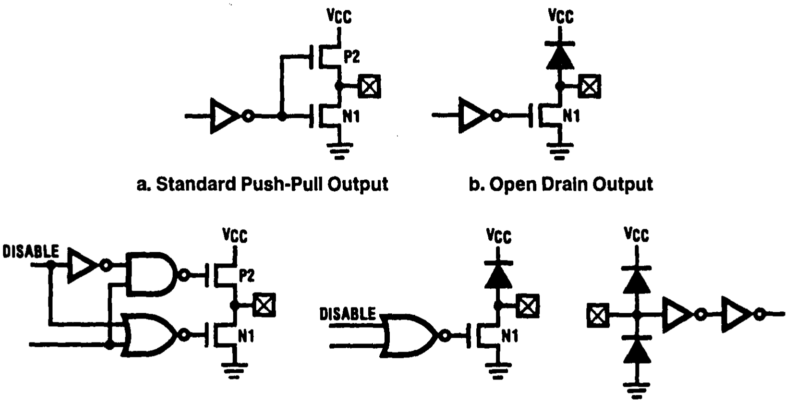
b. Open Drain. An N-channel device to ground only, allowing external pull-up as required by the user's application.

c. Standard TRI-STATE L Output. A CMOS output buffer similar to (a) which may be disabled by program control.

d. Open-Drain TRI-STATE L Output. This has the N-channel device to ground only.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (*Figure 7e*).

When using either the G or L I/O ports as inputs, an external pull-up device is necessary.



a. Standard Push-Pull Output

b. Open Drain Output

c. Standard TRI-STATE "L" Output

d. Open Drain TRI-STATE "L" Output

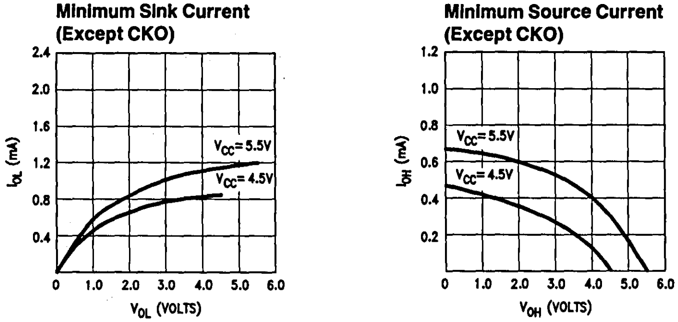
e. Hi-Z Input

TL/DD/8444-9

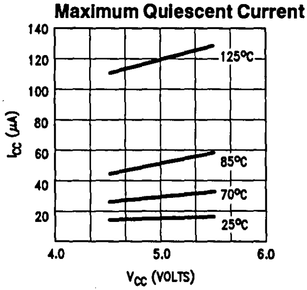
FIGURE 7. I/O Configurations

All output drivers uses one or two common devices numbered 1 to 2. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.

Typical Performance Characteristics



TL/DD/8444-10



TL/DD/8444-11

FIGURE 8

Option List

The COP210C/211C mask-programmable options are assigned numbers which correspond with the COP210C pins.

The following is a list of COP210C options. When specifying a COP211 chip, options 20, 21, and 22 must be set to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: 0 = Ground Pin. No options available.

Option 2: CKO I/O Port Determined by Option 3. = 0 no option (a. is crystal oscillator output for two pin oscillator b. is HALT I/O for one pin oscillator)

Option 3: CKI Input.
= 0: Crystal-controlled oscillator input (÷ 4).
= 1: Single-pin RC-controlled oscillator (÷ 4).
= 2: External oscillator input (÷ 4).
= 3: Crystal oscillator input (÷ 8).
= 4: External oscillator input (÷ 8).
= 5: Crystal oscillator input (÷ 16).
= 6: External oscillator input (÷ 16).

Option 4: $\overline{\text{RESET}}$ Input = 1: Hi-Z input. No option available.

Option 5: L_7 Driver
= 0: Standard TRI-STATE push-pull output.
= 2: Open-drain TRI-STATE output.

Option 6: L_6 Driver. (Same as Option 5.)

Option 7: L_5 Driver. (Same as Option 5.)

Option 8: L_4 Driver. (Same as Option 5.)

Option 9: V_{CC} Pin = 0 no option.

Option 10: L_3 Driver. (Same as Option 5.)

Option 11: L_2 Driver. (Same as Option 5.)

Option 12: L_1 Driver. (Same as Option 5.)

Option 13: L_0 Driver. (Same as Option 5.)

Option 14: SI Input.
No option available.
= 1: Hi-Z input.

Option 15: SO Output.
= 0: Standard push-pull output.
= 2: Open-drain output.

Option 16: SK Driver. (Same as Option 15.)

Option 17: G_0 I/O Port. (Same as Option 15.)

Option 18: G_1 I/O Port. (Same as Option 15.)

Option 19: G_2 I/O Port. (Same as Option 15.)

Option 20: G_3 I/O Port. (Same as Option 15.)

Option 21: D_3 Output. (Same as Option 15.)

Option 22: D_2 Output. (Same as Option 15.)

Option 23: D_1 Output. (Same as Option 15.)

Option 24: D_0 Output. (Same as Option 15.)

Option 25: Internal Initialization Logic.
= 0: Normal operation.
= 1: No internal initialization logic.

Option 26: No option available.

Option 27: COP Bonding
= 0: COP210C (24-pin device).
= 1: COP211C (20-pin device). See Note.
= 2: COP210C and COP211C. See Note.

Note: If option 27 = 1 or 2 then option 20 must = 0.

Option Table

Please fill out a photocopy of the Option Table and send along with your EPROM.

Option Table

Option 1 Value = ___0___ is: Ground Pin
Option 2 Value = ___0___ is: CKO Pin
Option 3 Value = _______ is: CKI Input
Option 4 Value = ___1___ is: $\overline{\text{RESET}}$ Input
Option 5 Value = _______ is: L_7 Driver
Option 6 Value = _______ is: L_6 Driver
Option 7 Value = _______ is: L_5 Driver
Option 8 Value = _______ is: L_4 Driver
Option 9 Value = ___0___ is: V_{CC} Pin
Option 10 Value = _______ is: L_3 Driver
Option 11 Value = _______ is: L_2 Driver
Option 12 Value = _______ is: L_1 Driver
Option 13 Value = _______ is: L_0 Driver
Option 14 Value = ___1___ is: SI Input

Option 15 Value = _______ is: SO Output
Option 16 Value = _______ is: SK Driver
Option 17 Value = _______ is: G_0 I/O Port
Option 18 Value = _______ is: G_1 I/O Port
Option 19 Value = _______ is: G_2 I/O Port
Option 20 Value = _______ is: G_3 I/O Port
Option 21 Value = _______ is: D_3 Output
Option 22 Value = _______ is: D_2 Output
Option 23 Value = _______ is: D_1 Output
Option 24 Value = _______ is: D_0 Output
Option 25 Value = _______ is: Internal Initialization Logic
Option 26 Value = ___0___ is: No Option
Option 27 Value = _______ is: COPS Bonding

National Semiconductor

COP224C/COP225C/COP226C/COP244C/COP245C Single-Chip 1k and 2k CMOS Microcontrollers

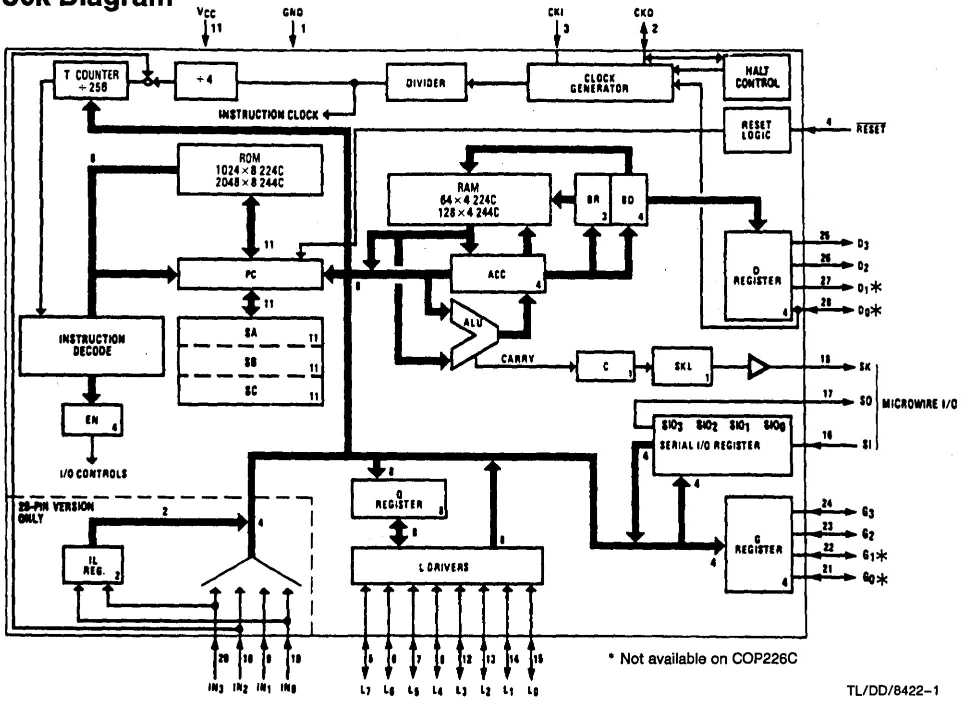
General Description

The COP224C, COP225C, COP226C, COP244C and COP245C fully static, Single-Chip CMOS Microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon gate microCMOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP224C and COP244C are 28 pin chips. The COP225C and COP245C are 24-pin versions (4 inputs removed) and COP226C is 20-pin version with 15 I/O lines. Standard test procedures and reliable high-density techniques provide the medium to large volume customers with a customized microcontroller at a low end-product cost. These microcontrollers are appropriate choices in many demanding control environments especially those with human interface.

Features

- Lowest power dissipation (600 μW typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4.4 μs instruction time
- 2k x 8 ROM, 128 x 4 RAM (COP244C/COP245C)
- 1k x 8 ROM, 64 x 4 RAM (COP224C/COP225C/COP226C)
- 23 I/O lines (COP244C and COP224C)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (4.5V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS output compatible
- Software/hardware compatible with COP400 family
- Military temperature (−55°C to +125°C) operation

Block Diagram



* Not available on COP226C

TL/DD/8422-1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at any Pin	−0.3V to V_{CC} +0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Total Allowable Power Dissipation	150 mW
Operating Temperature Range	−55°C to +125°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-55°C \le T_A \le +125°C$, $+4.5V \le V_{CC} \le +5.5V$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Power Supply Ripple (Note 5)	Peak to Peak		0.25 V_{CC}	V
Supply Current (Note 1)	V_{CC}=5.0V, tc=4.4 μs (tc is instruction cycle time)		5	mA
HALT Mode Current (Note 2)	V_{CC}=5.0V, F_{IN}=0 kHz		200	μA
Input Voltage Levels				
$\overline{\text{RESET}}$, CKI, D_0 (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		−10	+10	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels (except CKO)	Standard Outputs			
LSTTL Operation	V_{CC}=5.0V±10%			
Logic High	I_{OH}=−100 μA	2.7		V
Logic Low	I_{OL}=400 μA		0.6	V
CMOS Operation				
Logic High	I_{OH}=−10 μA	V_{CC}−0.2		V
Logic Low	I_{OL}=10 μA		0.2	V
CKO Current Levels (As Clock Out)				
Sink ÷4	CKI=V_{CC}, V_{OUT}=V_{CC}	0.2		mA
÷8		0.4		mA
÷16		0.8		mA
Source ÷4	CKI=0V, V_{OUT}=0V	−0.2		mA
÷8		−0.4		mA
÷16		−0.8		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			50	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	V_{IN}=0.2 V_{CC}		2.0	mA
To Halt	V_{IN}=0.7 V_{CC}		3.0	mA
TRI-STATE or Open Drain Leakage Current		−10	+10	μA

AC Electrical Characteristics $-55°C \leq T_A \leq +125°C$, $+4.5V \leq V_{CC} \leq +5.5V$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (tc)		4.4	DC	μs
Operating CKI Frequency ÷4 mode ÷8 mode ÷16 mode		DC DC DC	0.9 1.8 3.6	MHz MHz MHz
Duty Cycle (Note 4)	$f_1 = 3.6$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 3.6$ MHz External Clock		60	ns
Fall Time (Note 4)	$f_1 = 3.6$ MHz External Clock		40	ns

Instruction Cycle Time RC Oscillator (Note 4)	R=30k ±5%
	C=82 pF ±5% (÷4 Mode)
		6	18	μs

Inputs: (See *Figure 3*) (Note 4)				
t_{SETUP}	G Inputs	tc/4+0.8		μs
	SI Input	0.33		μs
	All Others	1.9		μs
t_{HOLD}		0.4		μs
Output Propagation Delay t_{PD1}, t_{PD0}	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5k$		1.4	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}, L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

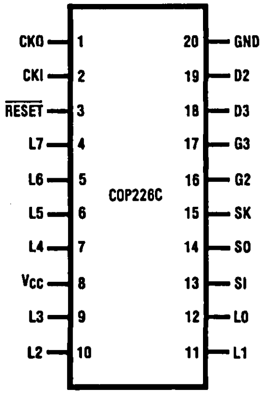
Note 4: This parameter is not tested but guaranteed by design. Variation due to the device included.

Note 5: Voltage change must be less than 0.25 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Connection Diagrams

S.O. Wide and DIP



Top View

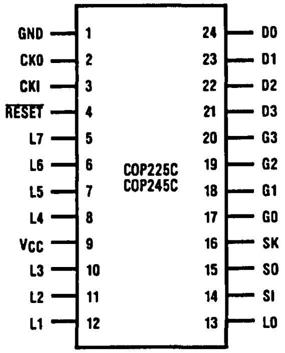
TL/DD/8422-2

Order Number COP226C-XXX/N
See NS Molded Package Number N20A

Order Number COP226C-XXX/D
See NS Hermetic Package Number D20A

Order Number COP226C-XXX/WM
See NS Surface Mount Package Number M20B

S.O. Wide and DIP



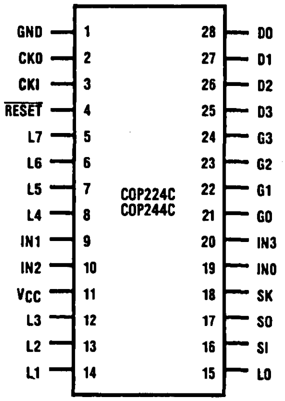
Top View

TL/DD/8422-3

Order Number COP225C-XXX/N
or COP245C-XXX/N
See NS Molded Package Number N24A

Order Number COP225C-XXX/D
or COP245C-XXX/D
See NS Hermetic Package Number D24C

DIP



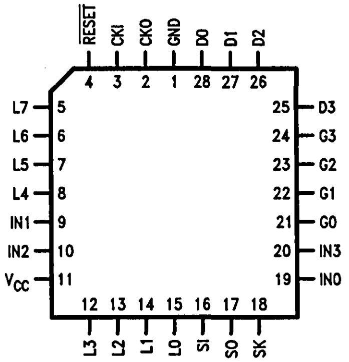
Top View

TL/DD/8422-4

Order Number COP224C-XXX/N
or COP244C-XXX/N
See NS Molded Package Number N28B

Order Number COP224C-XXX/D
or COP244C-XXX/D
See NS Hermetic Package Number D28C

28 PLCC



TL/DD/8422-13

Order Number COP224C-XXX/V
or COP244C-XXX/V
See NS PLCC Package Number V28A

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L7–L0	8-bit bidirectional port with TRI-STATE	SK	Logic controlled clock output
G3–G0	4-bit bidirectional I/O port	CKI	Chip oscillator input
		CKO	Oscillator output, HALT I/O port or general purpose input
D3–D0	4-bit output port		
IN3–IN0	4-bit input port (28 pin package only)	$\overline{\text{RESET}}$	Reset input
SI	Serial input or counter input	V_{CC}	Most positive power supply
SO	Serial or general purpose output	GND	Ground

Functional Description

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

Caution:

The output options available on the COP224C/225C/226C and COP244C/245C are not the same as those available on the COP324C/325C/326C, COP344C/345C, COP424C/425C/426C and COP444C/445C. Options not available on the COP224C/225C/226C and COP244C/245C are: Option 2 value 2; Option 4 value 0; Option 5 value 1; Option 9 value 0; Option 17 value 1; Option 30, Dual Clock, all values; Option 32, Microbus™, all values; Option 33 values 2, 4, and 6; Option 34 all values; and Option 35 all values.

PROGRAM MEMORY

Program Memory consists of ROM, 1024 bytes for the COP224C/225C/226C and 2048 bytes for the COP244C/245C. These bytes of ROM may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by an 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

Data memory consists of a 512-bit RAM for the COP244C/245C, organized as 8 data registers of 16 × 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register.

Data memory consists of a 256-bit RAM for the COP224C/225C/226C, organized as 4 data registers of 16 × 4-bits digits. The B register is 6 bits long. Upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions.

The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, to input 4 bits of L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes depending on a mask-programmable option: as a timer or as an external event counter. When the T counter overflows, an

Functional Description (Continued)

overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 7*.

Four general-purpose inputs, IN3–IN0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

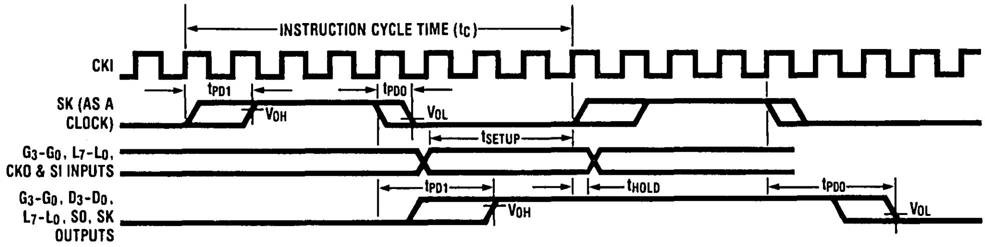
0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.
3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.

b. An interrupt will be recognized only on the following conditions:
   1. EN1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN_1 input.
   3. A currently executing instruction has been completed.


INSTRUCTION CYCLE TIME (tC)
CKI
SK (AS A CLOCK)
tPD1
tPD0
VOH
VOL
tSETUP
G3-G0, L7-L0, CKO & SI INPUTS
tPD1
tHOLD
tPD0
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
VOH
VOL

TL/DD/8422-5

FIGURE 3. Input/Output Timing Diagrams (divide by 8 mode)

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1,SK=clock If SKL=0,SK=0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL=1,SK=clock If SKL=0,SK=0
1	0	Binary Counter	Input to Counter	0	SK=SKL
1	1	Binary Counter	Input to Counter	1	SK=SKL

Functional Description (Continued)

4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

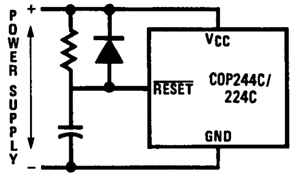
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 4* must be connected to the RESET pin (the conditions in *Figure 4* must be met). The RESET pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the RESET input, providing it stays low for at least three instruction cycle times.

Note: If CKI clock is less than 32 kHz, the internal reset logic (option #29 = 1) MUST be disabled and the external RC circuit must be used.



RC≥5X POWER SUPPLY RISE TIME AND RC≥100X CKI PERIOD.

TL/DD/8422-6

FIGURE 4. Power-Up Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).

TIMER

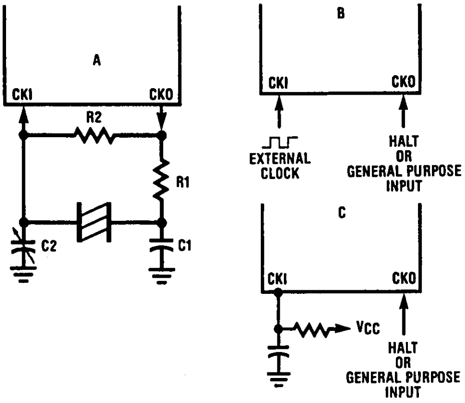
There are two modes selected by mask option:

a. Time-base counter. In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset.

For example, using a 3.58 MHz crystal with a divide-by-16 option, the instruction cycle frequency of 223.70 kHz increments the 10-bit timer every 4.47 μs. By presetting the counter and detecting overflow, accurate timeouts between 17.88 μs (4 counts) and 4.577 ms (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

b. External event counter. In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

Note: The IT instruction is not allowed in this mode.



TL/DD/8422-7

Crystal or Resonator

Crystal Value	Component Values			
	R1	R2	C1(pF)	C2(pF)
32 kHz	220k	20M	30	6–36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6–36
3.6 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R	C	Cycle Time	V_{CC}
30k	82 pF	6–18 μs	≥4.5V

Note: 15k≤R≤150k
50 pF≤C≤150 pF

FIGURE 5. Oscillator Component Values

Functional Description (Continued)

HALT MODE

The COP244C/245C/224C/225C/226C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by the HALT instruction or by forcing CKO high when it is mask-programmed as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The chip may be awakened by one of two different methods:

- Continue function: by forcing CKO low, if it mask-programmed as a HALT I/O port, the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization).

The HALT mode is the minimum power dissipation state.

CKO PIN OPTIONS

a. Two-pin oscillator—(Crystal). See *Figure 6a.*

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. The HALT mode may be entered by program control (HALT instruction) which forces CKO high, thus inhibiting the crystal network. The circuit can be awakened only by forcing the $\overline{\text{RESET}}$ pin to a logic "0" (restart).

b. One-pin oscillator—(RC or external). See *Figure 6b.*

If a one-pin oscillator system is chosen, two options are available for CKO:

- CKO can be selected as the HALT I/O port. In that case, it is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and CKO output will stay high to keep the chip stopped if the external driver returns to high impedance state. By forcing a low level to CKO, the chip will continue and CKO will stay low.
- As another option, CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction.

OSCILLATOR OPTIONS

There are three basic clock oscillator configurations available as shown by *Figure 5.*

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency optionally divided by 4, 8 or 16.

b. External Oscillator. The external frequency is optionally divided by 4, 8 or 16 to give the instruction cycle time. CKO is the HALT I/O port or a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port or a general purpose input.

Figure 7 shows the clock and timer diagram.

COP245C AND COP225C 24-PIN PACKAGE OPTION

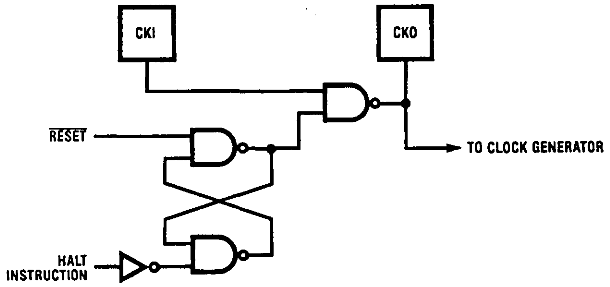
If the COP244C/224C is bonded in a 24-pin package, it becomes the COP245C/225C, illustrated in *Figure 2,* Connection diagrams. Note that the COP245C/225C does not contain the four general purpose IN inputs (IN3–IN0). Use of this option precludes, of course, use of the IN options, interrupt feature, external event counter feature.

Note: If user selects the 24-pin package, options 9, 10, 19 and 20 must be selected as a "2". See option list.

COP226C 20-PIN PACKAGE OPTION

If the COP225C is bonded as 20-pin device it becomes the COP226C. Note that the COP226C contains all the COP225C pins except D_0, D_1, G_0, and G_1.

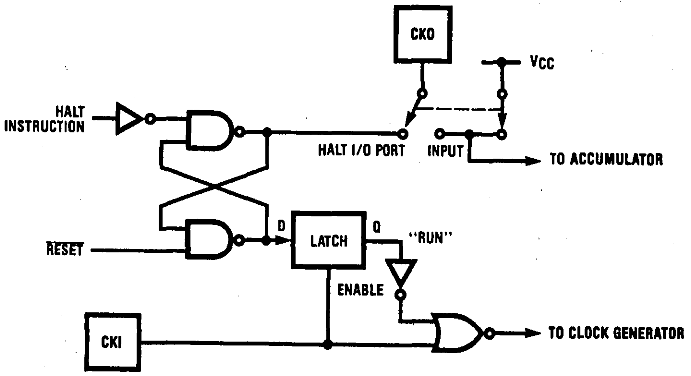
Block Diagram



TL/DD/8422-8

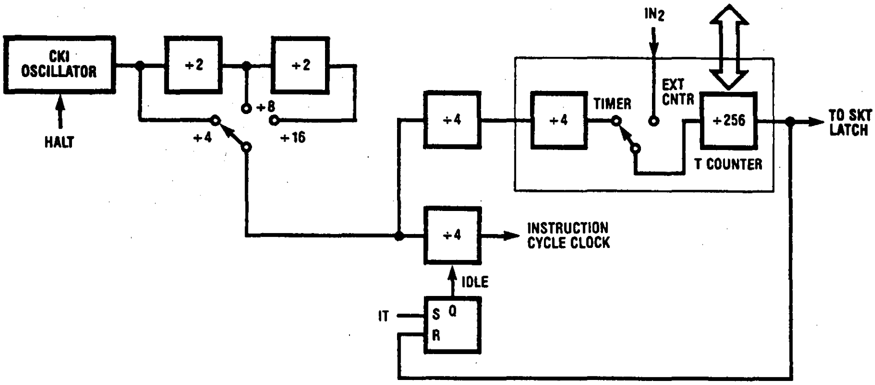
FIGURE 6a. Halt Mode—Two-Pin Oscillator

Block Diagrams (Continued)



TL/DD/8422-9

FIGURE 6b. Halt Mode—One-Pin Oscillator



TL/DD/8422-10

FIGURE 7. Clock and Timer

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

TABLE II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit accumulator
B	7-bit RAM address register (6-bit for COP224C)
Br	Upper 3 bits of B (register address) (2-bit for COP224C)
Bd	Lower 4 bits of B (digit address)
C	1-bit carry register
D	4-bit data output port
EN	4-bit enable register
G	4-bit general purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA,SB,SC	11-bit 3-level subroutine stack
SIO	4-bit shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Instruction Operand Symbols	
d	4-bit operand field, 0–15 binary (RAM digit select)
r	3(2)-bit operand field, 0–7(3) binary (RAM register select)
a	11-bit operand field, 0–2047 (1023)
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x

Operational Symbols	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	One's complement of A
⊕	Exclusive-or
:	Range of values

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

TABLE III. COP244C/245C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011|0000	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011|0001	A+RAM(B) → A	None	Add RAM to A
ADT		4A	0100|1010	$A+10_{10}$ → A	None	Add Ten to A
AISC	y	5–	0101| y	A+y → A	Carry	Add Immediate. Skip on Carry (y ≠ 0)
CASC		10	0001|0000	$\overline{A}$+RAM(B)+C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000|0000	0 → A	None	Clear A
COMP		40	0100|0000	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	0100|0100	None	None	No Operation
RC		32	0011|0010	"0" → C	None	Reset C
SC		22	0010|0010	"1" → C	None	Set C
XOR		02	0000|0010	A⊕RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{10:8}$ A,M) → $PC_{7:0}$	None	Jump Indirect (Notes 1, 3)
JMP	a	6– ––	0110 0 $a_{10:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	––	1 $a_{6:0}$ (pages 2, 3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 4)
		––	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	––	10 $a_{5:0}$	PC+1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– ––	0110 1 $a_{10:8}$ $a_{7:0}$	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	0011 0011 0011 1000		None	HALT Processor
IT		33 39	0011 0011 0011 1001		None	IDLE till Timer Overflows then Continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33 3F	0011 0011 0011 1111	A → $T_{7:4}$ RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CTMA		33 2F	0011 0011 0010 1111	$T_{7:4}$ → RAM(B) $T_{3:0}$ → A	None	Copy T to RAM, A
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101 (r=0:3)	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 ––	0010 0011 0 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate 1 and Increment Bd
X	r	–6	00 r 0110 (r=0:3)	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 – –	0010 0011 1 r d	RAM(r,d) ←→ A	None	Exchange A with RAM Pointed to Directly by r,d
XDS	r	–7	00 r 0111 (r=0:3)	RAM(B) ←→ A Bd–1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	–4	00 r 0100 (r=0:3)	RAM(B) ←→ A Bd+1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	– –	00 r (d–1) (r=0:3: d=0,9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33 – –	0011 0011 1 r d (any r, any d)			
LEI	y	33 6–	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A ←→ Br	None	Exchange A with Br (Note 8)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C="1"	Skip if C is True
SKE		21	0010 0001		A=RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0=0$	
	1	11	0001 0001		$G_1=0$	
	2	03	0000 0011		$G_2=0$	
	3	13	0001 0011		$G_3=0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1=0$	
	2	03	0000 0011		$RAM(B)_2=0$	
	3	13	0001 0011		$RAM(B)_3=0$	
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP244C/245C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN → A	None	Input IN Inputs to A (Note 2)
INIL		33 29	0011 0011 0010 1001	IL_3, CKO,"0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM,A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5–	0011 0011 0101 y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin packages since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B(Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: For 2K ROM devices, A ←→ Br (0 → A3). For 1K ROM devices, A ←→ Br (0,0 → A3, A2).

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10:PC8,A,M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC+1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC7:4, RAM(B) → PC3:0, leaving PC10, PC9 and PC8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10:8,A,M. PC10,PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT  ; load T counter
SKT   ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is mask-programmed as an external event counter (option #31 = 1).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKO and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A0 is input into A1. IL latches are cleared on reset. IL latches are not available on the COP245C/225C, and COP226C.

INSTRUCTION SET NOTES

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.

c. The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Note: The COP224C/225C/226C needs only 10 bits to address its ROM. Therefore, the eleventh bit (P10) is ignored.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal or resonator generated clock input will draw additional current. For example, a 500 kHz crystal input will typically draw 100 μA more than a squarewave input. An R/C oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate operating current drain.

$I_{CO} = I_Q + V \times 70 \times Fi + V \times 2400 \times Fi/Dv$ where:

I_{CO} = chip operating current drain in microamps

I_Q = quiescent leakage current (from curve)

Fi = CKI frequency in MegaHertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example at 5 volts V_{CC} and 400 kHz (divide by 4)

$I_{CO} = 120 + 5 \times 70 \times 0.4 + 5 \times 2400 \times 0.4/4$

$I_{CO} = 120 + 140 + 1200 = 1460\ \mu A$

Power Dissipation (Continued)

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$Ici = I_Q + V \times 70 \times Fi$$

For example, at 5 volts V_{CC} and 400 kHz

$$Ici = 120 + 5 \times 70 \times 0.4 = 260\ \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$Ita = I_{CO} \times \frac{To}{To+Ti} + Ici \times \frac{Ti}{To+Ti}$$

where: Ita = total average current
I_{CO} = operating current
Ici = idle current
To = operating time
Ti = idle time

I/O OPTIONS

Outputs have the following optional configurations, illustrated in *Figure 8*:

a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

b. Open Drain — An N-channel device to ground only, allowing external pull-up as required by the user's application.

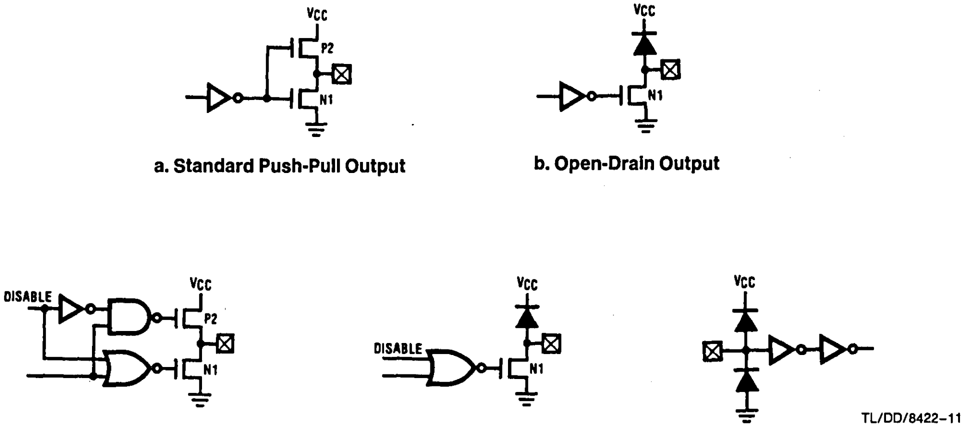
c. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control.

d. Open-Drain TRI-STATE L Output — This has the N-channel device to ground only.

All inputs have the following option:

e. Hi-Z input which must be driven by the users logic.

All output drivers use two common devices numbered 1 to 2. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 9* for each of these devices to allow the designer to effectively use these I/O configurations.



a. Standard Push-Pull Output

b. Open-Drain Output

c. Standard TRI-STATE "L" Output

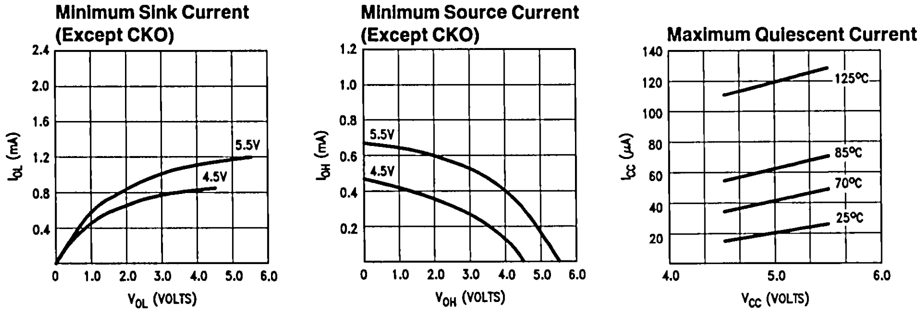
d. Open Drain TRI-STATE "L" Output

e. HI-Z Input

TL/DD/8422–11

FIGURE 8. Input/Output Configurations

Power Dissipation (Continued)



TL/DD/8422-12

FIGURE 9. Input/Output Characteristics

Option List

The COP244C/245C/224C/225C/COP226C mask-programmable options are assigned numbers which correspond with the COP244C/224C pins.

The following is a list of options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Caution:
The output options available on the COP224C/225C/226C and COP244C/245C are not the same as those available on the COP324C/325C/326C, COP344C/345C, COP424C/425C/426C and COP444C/445C. Options not available on the COP224C/225C/226C and COP244C/245C are: Option 2 value 2; Option 4 value 0; Option 5 value 1; Option 9 value 0; Option 17 value 1; Option 30, Dual Clock, all values; Option 32, Microbus, all values; Option 33 values 2 4, and 6; Option 34 all values; and Option 35 all values.

PLEASE FILL OUT THE OPTION TABLE on the next page. Photocopy the option data and send it in with your disk or EPROM.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Pin
 = 0: clock generator output to crystal/resonator
 = 1: HALT I/O port
 = 3: general purpose input, high-Z

Option 3: CKI input
 = 0: Crystal controlled oscillator input divide by 4
 = 1: Crystal controlled oscillator input divide by 8
 = 2: Crystal controlled oscillator input divide by 16
 = 4: Single-pin RC controlled oscillator (divide by 4)
 = 5: External oscillator input divide by 4
 = 6: External oscillator input divide by 8
 = 7: External oscillator input divide by 16

Option 4: $\overline{\text{RESET}}$ input
 = 1: Hi-Z input

Option 5: L7 Driver
 = 0: Standard TRI-STATE push-pull output
 = 2: Open-drain TRI-STATE output

Option 6: L6 Driver — (same as option 5)

Option 7: L5 Driver — (same as option 5)

Option 8: L4 Driver — (same as option 5)

Option 9: IN1 input
 = 1: Hi-Z input, mandatory for 28 Pin Package
 = 2: Mandatory for 20 and 24 Pin Packages

Option 10: IN2 input — (same as option 9)

Option 11 = 0: V_{CC} Pin — no option available

Option 12: L3 Driver — (same as option 5)

Option 13: L2 Driver — (same as option 5)

Option 14: L1 Driver — (same as option 5)

Option 15: L0 Driver — (same as option 5)

Option 16: SI input — (same as option 4)

Option 17: SO Driver
 = 0: Standard push-pull output
 = 2: Open-drain output

Option 18: SK Driver — (same as option 17)

Option 19: IN0 Input — (same as option 9)

Option 20: IN3 Input — (same as option 9)

Option 21: G0 I/O Port — (same as option 17)

Option 22: G1 I/O Port — (same as option 17)

Option 23: G2 I/O Port — (same as option 17)

Option 24: G3 I/O Port — (same as option 17)

Option 25: D3 Output — (same as option 17)

Option 26: D2 Output — (same as option 17)

Option 27: D1 Output — (same as option 17)

Option List (Continued)

Option 28: D0 Output — (same as option 17)

Option 29: Internal Initialization Logic
  =0: Normal operation
  =1: No internal initialization logic

Option 30=0: No Option Available

Option 31: Timer
  =0: Time-base counter
  =1: External event counter

Option 32=0: No Option Available

Option 33: COP bonding. See note.
  (1k and 2k Microcontroller)
  =0: 28-pin package
  =1: 24-pin package
  (1k Microcontroller only)
  =3: 20-pin package
  =5: 24- and 20-pin package

Note:—If opt. #33=0 then opt. #9, 10, 19, and 20 must=1.

If opt. #33=1 then opt. #9, 10, 19 and 20 must=2, and option #31 must=0.

If opt. #33=3 or 5 then opt. #9, 10, 19, 20 must=2 and opt. #21, 22, 31 must=0.

Option 34=0: No Option Available

Option 35=0: No Option Available

Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE = 0 IS: GROUND PIN
OPTION 2 VALUE = ________ IS: CKO PIN
OPTION 3 VALUE = ________ IS: CKI INPUT
OPTION 4 VALUE = 1 IS: $\overline{\text{RESET}}$ INPUT
OPTION 5 VALUE = ________ IS: L7 DRIVER
OPTION 6 VALUE = ________ IS: L6 DRIVER
OPTION 7 VALUE = ________ IS: L5 DRIVER
OPTION 8 VALUE = ________ IS: L4 DRIVER
OPTION 9 VALUE = ________ IS: IN1 INPUT
OPTION 10 VALUE = ________ IS: IN2 INPUT
OPTION 11 VALUE = 0 IS: VCC PIN
OPTION 12 VALUE = ________ IS: L3 DRIVER
OPTION 13 VALUE = ________ IS: L2 DRIVER
OPTION 14 VALUE = ________ IS: L1 DRIVER
OPTION 15 VALUE = ________ IS: L0 DRIVER
OPTION 16 VALUE = 1 IS: SI INPUT
OPTION 17 VALUE = ________ IS: SO DRIVER
OPTION 18 VALUE = ________ IS: SK DRIVER

OPTION DATA

OPTION 19 VALUE = ________ IS: IN0 INPUT
OPTION 20 VALUE = ________ IS: IN3 INPUT
OPTION 21 VALUE = ________ IS: G0 I/O PORT
OPTION 22 VALUE = ________ IS: G1 I/O PORT
OPTION 23 VALUE = ________ IS: G2 I/O PORT
OPTION 24 VALUE = ________ IS: G3 I/O PORT
OPTION 25 VALUE = ________ IS: D3 OUTPUT
OPTION 26 VALUE = ________ IS: D2 OUTPUT
OPTION 27 VALUE = ________ IS: D1 OUTPUT
OPTION 28 VALUE = ________ IS: D0 OUTPUT
OPTION 29 VALUE = ________ IS: INT INIT LOGIC
OPTION 30 VALUE = 0 IS: N/A
OPTION 31 VALUE = ________ IS: TIMER
OPTION 32 VALUE = 0 IS: N/A
OPTION 33 VALUE = ________ IS: COP BONDING
OPTION 34 VALUE = 0 IS: N/A
OPTION 35 VALUE = 0 IS: N/A

National Semiconductor

COP410C/COP411C/COP310C/COP311C Single-Chip CMOS Microcontrollers

General Description

The COP410C, COP411C, COP310C, and COP311C fully static, single-chip CMOS microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP411C is identical to the COP410C but with 16 I/O lines instead of 20. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller-oriented processor at a low end-product cost.

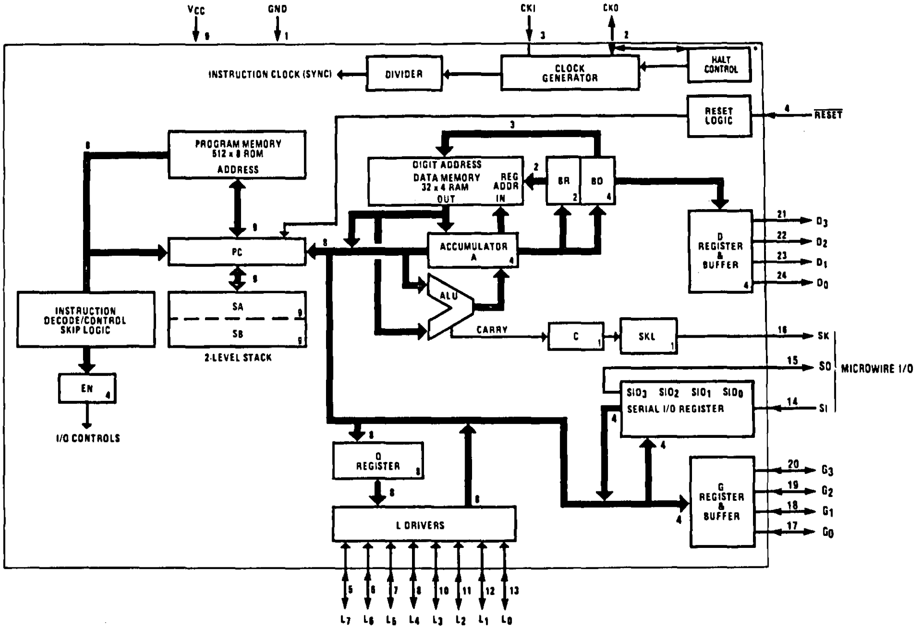
The COP310C/COP311C is the extended temperature range version of the COP410C/COP411C.

The COP404C should be used for exact emulation.

Features

- Lowest power dissipation (40 μW typical)
- Low cost
- Power-saving HALT Mode with Continue function
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 20 I/O lines (COP410C)
- Two-level subroutine stack
- DC to 4 μs instruction time
- Single supply operation (2.4V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of the COP400 family
- Extended temperature (−40°C to +85°C) devices available
- The military temperature range devices (−55°C to +125°C) are specified on COP210C/211C data sheet.

Block Diagram



TL/DD/5015-1

FIGURE 1. COP410C

COP410C/COP411C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	$-0.3V$ to $V_{CC}+0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \leq T_A \leq 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple[5]			$0.1\ V_{CC}$	V
Supply Current	$V_{CC} = 2.4V$, $t_c = 125\ \mu s$ $V_{CC} = 5.0V$, $t_c = 16\ \mu s$ $V_{CC} = 5.0V$, $t_c = 4\ \mu s$ (t_c is instruction cycle time)		80 500 2000	μA μA μA
HALT Mode Current[2]	$V_{CC} = 5.0V$, $F_{IN} = 0$ kHz $V_{CC} = 2.4V$, $F_{IN} = 0$ kHz		30 10	μA μA
Input Voltage Levels $\overline{\text{RESET}}$, CKI				
Logic High		$0.9\ V_{CC}$		V
Logic Low			$0.1\ V_{CC}$	V
All Other Inputs Logic High		$0.7\ V_{CC}$		V
Logic Low			$0.2\ V_{CC}$	V
Hi-Z Input Leakage		−1	+1	μA
Input Capacitance			7	pF
Output Voltage Levels LSTTL Operation	Standard Outputs $V_{CC} = 5.0V \pm 10\%$			
Logic High	$I_{OH} = -25\ \mu A$	2.7		V
Logic Low	$I_{OL} = 400\ \mu A$		0.4	V
CMOS Operation Logic High	$I_{OH} = -10\ \mu A$	$V_{CC}-0.2$		V
Logic Low	$I_{OL} = 10\ \mu A$		0.2	V
Output Current Levels[4] (Except CKO) Sink	$V_{CC} = 4.5V$, $V_{OUT} = V_{CC}$	1.2		mA
	$V_{CC} = 2.4V$, $V_{OUT} = V_{CC}$	0.2		mA
Source (Standard	$V_{CC} = 4.5V$, $V_{OUT} = 0V$	−0.5		mA
Option)	$V_{CC} = 2.4V$, $V_{OUT} = 0V$	−0.1		mA
Source (Low	$V_{CC} = 4.5V$, $V_{OUT} = 0V$	−30	−330	μA
Current Option)	$V_{CC} = 2.4V$, $V_{OUT} = 0V$	−6	−80	μA
CKO Current Levels (As Clock Out) Sink ÷4	$V_{CC} = 4.5V$, CKI = V_{CC}, $V_{OUT} = V_{CC}$	0.3		mA
÷8		0.6		mA
÷16		1.2		mA
Source ÷4	$V_{CC} = 4.5V$, CKI = 0V, $V_{OUT} = 0V$	−0.3		mA
÷8		−0.6		mA
÷16		−1.2		mA
Allowable Sink/Source Current Per Pin[4]			5	mA

COP410C/COP411C

DC Electrical Characteristics (Continued)

Parameter	Conditions	Min	Max	Units
Allowable Loading on CKO (as HALT I/O pin)			100	pF
Current Needed to Override HALT[3]				
To Continue	V_{CC} = 4.5V, V_{IN} = 0.2 V_{CC}		0.6	mA
To Halt	V_{CC} = 4.5V, V_{IN} = 0.7 V_{CC}		1.6	mA
TRI-STATE or Open Drain Leakage Current		−2	+2	μA

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating in the RC and crystal configurations.

Note 3: When forcing HALT, current is only needed for a short time (approximately 200 ns) to flip the HALT flip-flop.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 5: Voltage change must be less than 0.5V in a 1 ms period.

Note 6: This parameter is only sampled and not 100% tested.

Note 7: Variation due to the device included.

COP410C/COP411C

AC Electrical Characteristics 0°C ≤ T_A ≤ 70°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)	V_{CC} ≥ 4.5V	4	DC	μs
	4.5V > V_{CC} ≥ 2.4V	16	DC	μs
Operating CKI Frequency ÷4 mode	V_{CC} ≥ 4.5V	DC	1.0	MHz
÷8 mode		DC	2.0	MHz
÷16 mode		DC	4.0	MHz
÷4 mode	4.5V > V_{CC} ≥ 2.4V	DC	250	kHz
÷8 mode		DC	500	kHz
÷16 mode		DC	1.0	MHz

Instruction Cycle Time RC Oscillator[7]	R = 30k ± 5%, V_{CC} = 5V
	C = 82 pF ± 5% (÷4 Mode)
		8	16	μs

Duty Cycle[6]	f_I = 4 MHz	40	60	%
Rise Time[6]	f_I = 4 MHz External Clock		60	ns
Fall Time[6]	f_I = 4 MHz External Clock		40	ns
Inputs (See *Figure 3*)				
t_{SETUP}	G Inputs, V_{CC} ≥ 4.5V	tc/4+0.7		μs
	SI Input, V_{CC} ≥ 4.5V	0.3		μs
	All Others, V_{CC} ≥ 4.5V	1.7		μs
t_{HOLD}	V_{CC} ≥ 4.5V	0.25		μs
	V_{CC} ≥ 2.4V	1.0		μs
Output Propagation Delay	V_{OUT} = 1.5V, C_L = 100 pF, R_L = 5k			
t_{PD1}, t_{PD0}	V_{CC} ≤ 4.5V		1.0	μs
t_{PD1}, t_{PD0}	V_{CC} ≤ 2.4V		4.0	μs

COP310C/COP311C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	−0.3V to V_{CC}+0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics −40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		3.0	5.5V	V
Power Supply Ripple[5]			0.1 V_{CC}	V
Supply Current	V_{CC} = 3.0V, t_c = 125 μs		100	μA
	V_{CC} = 5.0V, t_c = 16 μs		600	μA
	V_{CC} = 5.0V, t_c = 4 μs		2500	μA
	(t_c is instruction cycle time)			
HALT Mode Current[2]	V_{CC} = 5.0V, F_{IN} = 0 kHz		50	μA
	V_{CC} = 3.0V, F_{IN} = 0 kHz		20	μA
Input Voltage Levels				
$\overline{\text{RESET}}$, CKI				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Hi-Z Input Leakage		−2	+2	μA
Input Capacitance			7	pF
Output Voltage Levels	Standard Outputs			
LSTTL Operation	V_{CC} = 5.0V ±10%			
Logic High	I_{OH} = −25 μA	2.7		V
Logic Low	I_{OL} = 400 μA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC}−0.2		V
Logic Low	I_{OL} = 10 μA		0.2	V
Output Current Levels[4]				
(Except CKO)				
Sink	V_{CC} = 4.5V, V_{OUT} = V_{CC}	1.2		mA
	V_{CC} = 3.0V, V_{OUT} = V_{CC}	0.2		mA
Source (Standard	V_{CC} = 4.5V, V_{OUT} = 0V	−0.5		mA
Option)	V_{CC} = 3.0V, V_{OUT} = 0V	−0.1		mA
Source (Low	V_{CC} = 4.5V, V_{OUT} = 0V	−30	−440	μA
Current Option)	V_{CC} = 3.0V, V_{OUT} = 0V	−8	−200	μA
CKO Current Levels				
(As Clock Out)				
Sink ÷4	V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC}	0.3		mA
÷8		0.6		mA
÷16		1.2		mA
Source ÷4	V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V	−0.3		mA
÷8		−0.6		mA
÷16		−1.2		mA
Allowable Sink/Source Current Per Pin[4]			5	mA

COP310C/COP311C

DC Electrical Characteristics (Continued)

Parameter	Conditions	Min	Max	Units
Allowable Loading on CKO (as HALT I/O pin)			100	pF
Current Needed to Override HALT[3]				
To Continue	$V_{CC} = 4.5V, V_{IN} = 0.2\,V_{CC}$		0.8	mA
To Halt	$V_{CC} = 4.5V, V_{IN} = 0.7\,V_{CC}$		2.0	mA
TRI-STATE or Open Drain Leakage Current		−4	+4	μA

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating in the RC and crystal configurations.

Note 3: When forcing HALT, current is only needed for a short time (approximately 200 ns) to flip the HALT flip-flop.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 5: Voltage change must be less than 0.5V in a 1 ms period.

Note 6: This parameter is only sampled and not 100% tested.

Note 7: Variation due to the device included.

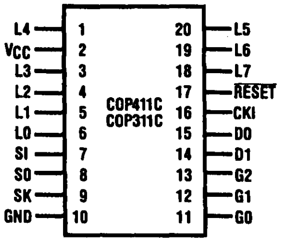
COP310C/COP311C

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_c)	$V_{CC} \ge 4.5V$	4	DC	μs
	$4.5V > V_{CC} \ge 3.0V$	16	DC	μs
Operating CKI Frequency ÷4 mode	$V_{CC} \ge 4.5V$	DC	1.0	MHz
÷8 mode		DC	2.0	MHz
÷16 mode		DC	4.0	MHz
÷4 mode	$4.5V > V_{CC} \ge 3.0V$	DC	250	kHz
÷8 mode		DC	500	kHz
÷16 mode		DC	1.0	MHz
Instruction Cycle Time RC Oscillator[7]	$R = 30k \pm 5\%, V_{CC} = 5V$ $C = 82\,pF \pm 5\%$ (÷4 Mode)	8	16	μs
Duty Cycle[6]	$f_I = 4$ MHz	40	60	%
Rise Time[6]	$f_I = 4$ MHz External Clock		60	ns
Fall Time[6]	$f_I = 4$ MHz External Clock		40	ns
Inputs (See *Figure 3*)				
t_{SETUP}	G Inputs, $V_{CC} \ge 4.5V$	tc/4+0.7		μs
	SI Input	0.3		μs
	All Others	1.7		μs
t_{HOLD}	$V_{CC} \ge 4.5V$	0.25		μs
	$V_{CC} \ge 3.0V$	1.0		μs
Output Propagation Delay	$V_{OUT} = 1.5V, C_L = 100\,pF, R_L = 5k$			
t_{PD1}, t_{PD0}	$V_{CC} \le 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$V_{CC} \le 3.0V$		4.0	μs

Connection Diagrams

S.O. Wide and DIP



TL/DD/5015-2

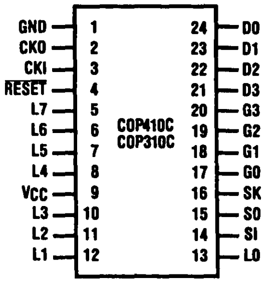
Top View

Order Number COP311C-XXX/D or COP411C-XXX/D
See NS Hermetic Package Number D20A

Order Number COP311C-XXX/N or COP411C-XXX/N
See NS Molded Package Number N20A

Order Number COP311C-XXX/WM or
COP411C-XXX/WM
See NS Surface Mount Package Number M20B

S.O. Wide and DIP



TL/DD/5015-3

Top View

Order Number COP310C-XXX/D or COP410C-XXX/D
See NS Hermetic Package Number D24C

Order Number COP310C-XXX/N or COP410C-XXX/N
See NS Molded Package Number N24A

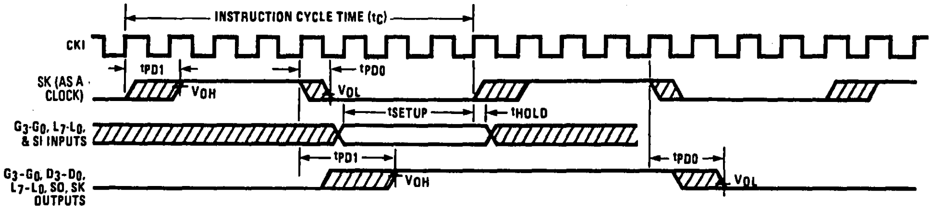
Order Number COP310C-XXX/WM or
COP410C-XXX/WM
See NS Surface Mount Package Number M24B

FIGURE 2

Pin Descriptions

Pin	Description	Pin	Description
L_7–L_0	8-bit bidirectional I/O port with TRI-STATE	SK	Logic-controlled clock (or general purpose output)
G_3–G_0	4-bit bidirectional I/O port (G_2–G_0 for 20-pin package)	CKI	System oscillator input
D_3–D_0	4-bit general purpose output port (D_1–D_0 for 20-pin package)	CKO	Crystal oscillator output, or HALT mode I/O port (24-pin package only)
SI	Serial input (or counter input)	$\overline{\text{RESET}}$	System reset input
SO	Serial output (or general purpose output)	V_{CC}	System power supply
		GND	System Ground

Timing Diagram


INSTRUCTION CYCLE TIME (tC)
CKI
tPD1
tPD0
SK (AS A CLOCK)
VOH
VOL
tSETUP
tHOLD
G3-G0, L7-L0, & SI INPUTS
tPD1
tPD0
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
VOH
VOL

TL/DD/5015-4

FIGURE 3. Input/Output (Divide-by-8 Mode)

Functional Description

To ease reading of this description, only COP410C and/or COP411C are referenced; however, all such references apply equally to COP310C and/or COP311C, respectively.

A block diagram of the COP410C is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP410C/411C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8 × 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4.* The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP410C/411C is designed to ensure fully static operation of the device.

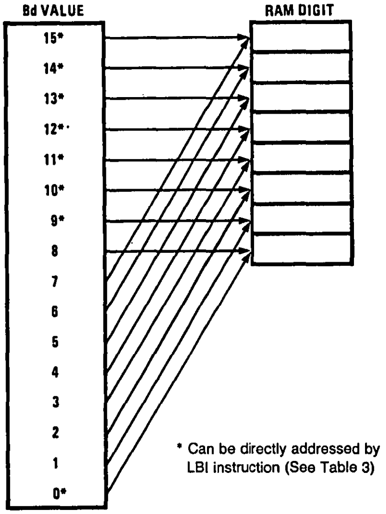
The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP410C/411C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.



* Can be directly addressed by LBI instruction (See Table 3)

TL/DD/5015-5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP410C/411C is MICROWIRE compatible.

The D register provides four general purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.
2. EN 1 is not used, it has no effect on the COP410C/411C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

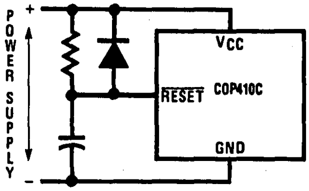
INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

When V_{CC} power is applied, the internal reset logic will keep the chip in initialization mode for up to 2500 instruction cycles. If the CKI clock is running at a low frequency, this could take a long time, therefore, the internal logic should be disabled by a mask option with initialization controlled solely by $\overline{\text{RESET}}$ pin.

Note: If CKI clock is less than 32 kHz, the internal reset logic (Option 25 = 1) *must* be disabled and the external RC network *must* be present.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/5015-6

RC > 5 × Power Supply Rise Time
and RC > 100 × CKI Period

FIGURE 5. Power-Up Clear Circuit

COP411C

If the COP410C is bonded as a 20-pin package, it becomes the COP411C, illustrated in *Figure 2,* COP410C/411C Connection Diagrams. Note that the COP411C does not contain D2, D3, G3, or CKO. Use of this option, of course, precludes use of D2, D3, G3, and CKO options. All other options are available for the COP411C.

TABLE I. Enable Register Modes — Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

HALT MODE

The COP410C/411C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip also may be halted by the HALT instruction or by forcing CKO high when it is used as a HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode has slight differences depending upon the type of oscillator used.

a. 1-pin oscillator—RC or external

The HALT mode may be entered into by either program control (HALT instruction) or by forcing CKO to a logic "1" state.

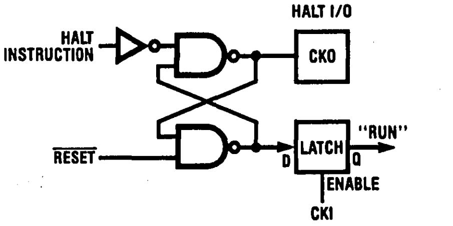
The circuit may be awakened by one of two different methods:

1) Continue function. By forcing CKO to a logic "0", the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.

2) Restart. Forcing the $\overline{\text{RESET}}$ pin to a logic "0" will restart the chip regardless of HALT or CKO (see initialization).

b. 2-pin oscillator—crystal

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.



Halt I/O Port

CKO Pin Options

In a crystal-controlled oscillator system, CKO is used as an output to the crystal network. CKO will be forced high during the execution of a HALT instruction, thus inhibiting the crystal network. If a 1-pin oscillator system is chosen (RC or external), CKO will be selected as HALT and is an I/O flip-flop which is an indicator of the HALT status. An external signal can override this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and the CKO output will go high to keep the chip stopped. By forcing a low level to CKO, the chip will continue and CKO output will go low.

All features associated with the CKO I/O pin are available with the 24-pin package only.

OSCILLATOR OPTIONS

There are three options available that define the use of CKI and CKO.

a. Crystal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optionally by 8 or 4).

b. External Oscillator. CKI is configured as LSTTL-compatible input accepting an external clock signal. The external frequency is divided by 16 (optionally by 8 or 4) to give the instruction cycle time. CKO is the HALT I/O port.

c. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 μA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.

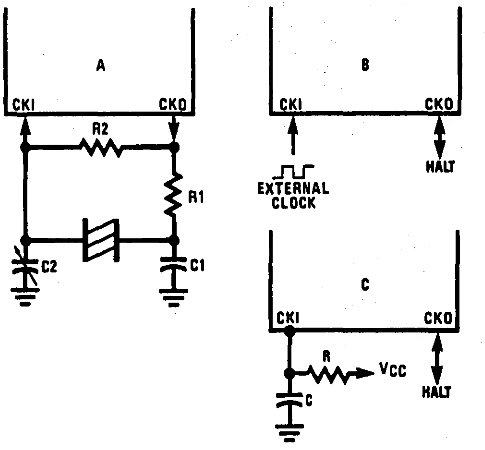


FIGURE 6. COP410C Oscillator

Crystal or Resonator

RC-Controlled Oscillator

Crystal		Component Value					Cycle	
Value	R1	R2	C1 pF	C2 pF	R	C	Time	V_{CC}
32 kHz	220k	20M	30	5-36	15k	82 pF	4-9 μs	≥4.5V
455 kHz	5k	10M	80	40	30k	82 pF	8-16 μs	≥4.5V
2.096 MHz	2k	1M	30	6-36	47k	100 pF	16-32 μs	2.4 to 4.5
4.0 MHz	1k	1M	30	6-36	Note: 15k ≤ R ≤ 150k,			
					50 pf ≤ C ≤ 150 pF			

COP410C/COP411C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410C/411C instruction set.

TABLE II. COP410C/411C Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP410C/411C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	0101 y	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP410C/411C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– –	0110 000 a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	–	1 $a_{6:0}$ (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 1)
		–	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	–	10 $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 2)
JSR	a	6– –	0110 100 a_8 $a_{7:0}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 10011	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	0011 0011 0011 1000		None	Halt processor
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	1011 1111	ROM(PC_8,A,M) → Q SA → SB	None	Load Q Indirect
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	0010 0011 1011 1111	RAM(3,15) ←→ A	None	Exchange A with RAM (3,15)

Instruction Set (Continued)

TABLE III. COP410C/411C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	−7	00 r 0111	RAM(B) ←→ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	−4	00 r 0100	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	-	00 r (d - 1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6−	0011 0011 0010 y	y → EN	None	Load EN Immediate
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0 = 0$	
	1	11	0001 0001		$G_1 = 0$	
	2	03	0000 0011		$G_2 = 0$	
	3	13	0010 0011		$G_3 = 0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1 = 0$	
	2	03	0000 0011		$RAM(B)_2 = 0$	
	3	13	0001 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP410C/411C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register). If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

INSTRUCTION SET NOTES

a. The first word of a COP410C/411C program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).

c. The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw additional current. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP410C current drain.

$$Ic = Iq + (V \times 20 \times Fi) + (V \times 1280 \times FI/Dv)$$

where Ic = chip current drain in microamps

Iq = quiescent leakage current (from curve)

FI = CKI frequency in megahertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 4),

$$Ic = 10 + (5 \times 20 \times 0.4) + (5 \times 1280 \times 0.4/4)$$

$$Ic = 10 + 40 + 640 = 690\ \mu A$$

I/O OPTIONS

COP410C/411C outputs have the following optional configurations, illustrated in *Figure 7*:

a. Standard. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

b. Low Current. This is the same configuration as (a) above except that the sourcing current is much less.

c. Open Drain. An N-channel device to ground only, allowing external pull-up as required by the user's application.

d. Standard TRI-STATE L Output. A CMOS output buffer similar to (a) which may be disabled by program control.

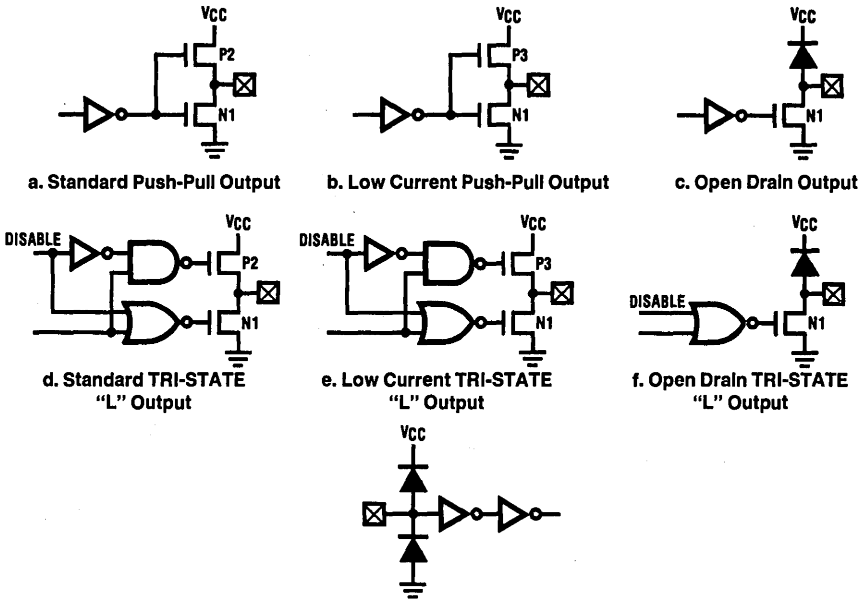
e. Low-Current TRI-STATE L Output. This is the same as (d) above except that the sourcing current is much less.

f. Open-Drain TRI-STATE L Output. This has the N-channel device to ground only.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (*Figure 7g*).

When using either the G or L I/O ports as inputs, a pull-up device is necessary. This can be an external device or the following alternative is available: Select the low-current output option. Now, by setting the output registers to a logic "1" level, the P-channel devices will act as the pull-up load. Note that when using the L ports in this fashion, the Q registers must be set to a logic "1" level and the L drivers *must be enabled* by an LEI instruction.

Functional Description (Continued)



a. Standard Push-Pull Output

b. Low Current Push-Pull Output

c. Open Drain Output

d. Standard TRI-STATE "L" Output

e. Low Current TRI-STATE "L" Output

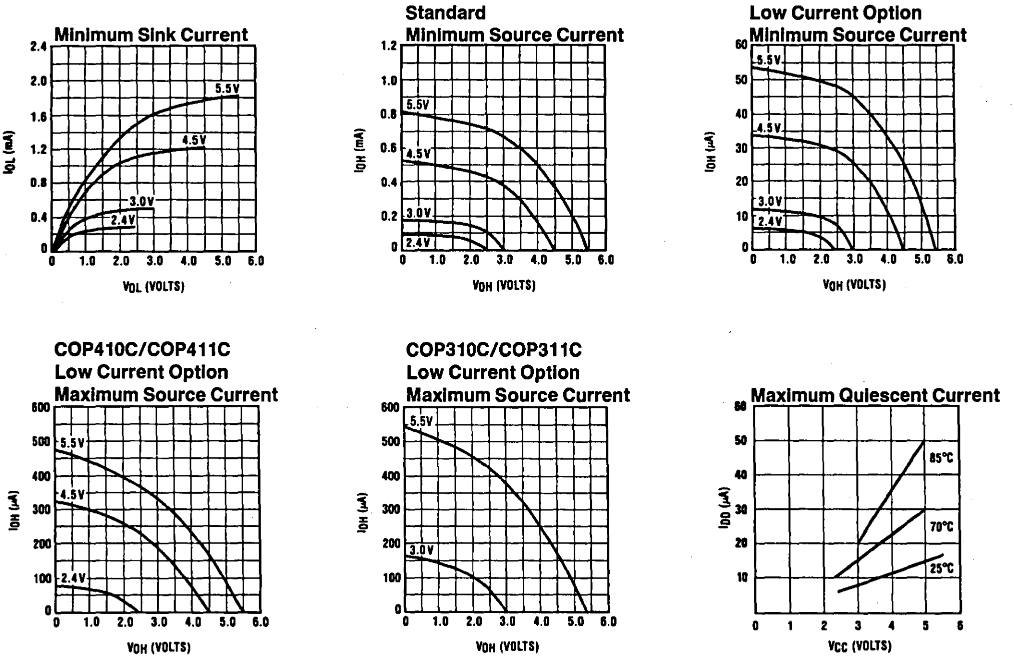
f. Open Drain TRI-STATE "L" Output

g. HI-Z Input

TL/DD/5015-9

FIGURE 7. I/O Configurations

Typical Performance Characteristics


Minimum Sink Current

Standard Minimum Source Current

Low Current Option Minimum Source Current

COP410C/COP411C Low Current Option Maximum Source Current

COP310C/COP311C Low Current Option Maximum Source Current

Maximum Quiescent Current

TL/DD/5015-10

FIGURE 8

All output drivers uses one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.

Option List

The COP410C/411C mask-programmable options are assigned numbers which correspond with the COP410C pins.

The following is a list of COP410C options. When specifying a COP411 chip, options 20, 21, and 22 must be set to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: 0 = Ground Pin. No options available.

Option 2: CKO I/O Port. (Determined by Option 3.)
= 0: No option.
(a. is crystal oscillator output for two pin oscillator.
b. is HALT I/O for one pin oscillator.)

Option 3: CKI Input.
= 0: Crystal-controlled oscillator input (÷ 4).
= 1: Single-pin RC-controlled oscillator (÷ 4).
= 2: External oscillator input (÷ 4).
= 3: Crystal oscillator input (÷ 8).
= 4: External oscillator input (÷ 8).
= 5: Crystal oscillator input (÷ 16).
= 6: External oscillator input (÷ 16).

Option 4: $\overline{\text{RESET}}$ Input = 1: Hi-Z input. No option available.

Option 5: L_7 Driver
= 0: Standard TRI-STATE push-pull output.
= 1: Low-current TRI-STATE push-pull output.
= 2: Open-drain TRI-STATE output.

Option 6: L_6 Driver. (Same as Option 5.)

Option 7: L_5 Driver. (Same as Option 5.)

Option 8: L_4 Driver. (Same as Option 5.)

Option 9: V_{CC} Pin = 0 no option.

Option 10: L_3 Driver. (Same as Option 5.)

Option 11: L_2 Driver. (Same as Option 5.)

Option 12: L_1 Driver. (Same as Option 5.)

Option 13: L_0 Driver. (Same as Option 5.)

Option 14: SI Input.
No option available.
= 1: Hi-Z input.

Option 15: SO Output.
= 0: Standard push-pull output.
= 1: Low-current push-pull output.
= 2: Open-drain output.

Option 16: SK Driver. (Same as Option 15.)

Option 17: G_0 I/O Port. (Same as Option 15.)

Option 18: G_1 I/O Port. (Same as Option 15.)

Option 19: G_2 I/O Port. (Same as Option 15.)

Option 20: G_3 I/O Port. (Same as Option 15.)

Option 21: D_3 Output. (Same as Option 15.)

Option 22: D_2 Output. (Same as Option 15.)

Option 23: D_1 Output. (Same as Option 15.)

Option 24: D_0 Output. (Same as Option 15.)

Option 25: Internal Initialization Logic.
= 0: Normal operation.
= 1: No internal initialization logic.

Option 26: No option available.

Option 27: COP Bonding
= 0: COP410C (24-pin device).
= 1: COP411C (20-pin device). See note.
= 2: COP410C and COP411C. See note.

Note: If opt. #27 = 1 or 2 then opt #20 must = 0.

Option Table

Please fill out a photocopy of the option table and send it along with your EPROM.

Option Table

Option 1 Value =	0	is: Ground Pin	Option 15 Value =		is: SO Output
Option 2 Value =	0	is: CKO Pin	Option 16 Value =		is: SK Driver
Option 3 Value =		is: CKI Input	Option 17 Value =		is: G_0 I/O Port
Option 4 Value =	1	is: $\overline{\text{RESET}}$ Input	Option 18 Value =		is: G_1 I/O Port
Option 5 Value =		is: L_7 Driver	Option 19 Value =		is: G_2 I/O Port
Option 6 Value =		is: L_6 Driver	Option 20 Value =		is: G_3 I/O Port
Option 7 Value =		is: L_5 Driver	Option 21 Value =		is: D_3 Output
Option 8 Value =		is: L_4 Driver	Option 22 Value =		is: D_2 Output
Option 9 Value =	0	is: V_{CC} Pin	Option 23 Value =		is: D_1 Output
Option 10 Value =		is: L_3 Driver	Option 24 Value =		is: D_0 Output
Option 11 Value =		is: L_2 Driver	Option 25 Value =		is: Internal
Option 12 Value =		is: L_1 Driver			Initialization
Option 13 Value =		is: L_0 Driver			Logic
Option 14 Value =	1	is: SI Input	Option 26 Value =	0	is: N/A
			Option 27 Value =		is: COP Bonding

National Semiconductor

COP410L/COP411L/COP310L/COP311L Single-Chip N-Channel Microcontrollers

General Description

The COP410L and COP411L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP411L is identical to the COP410L, but with 16 I/O lines instead of 19. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

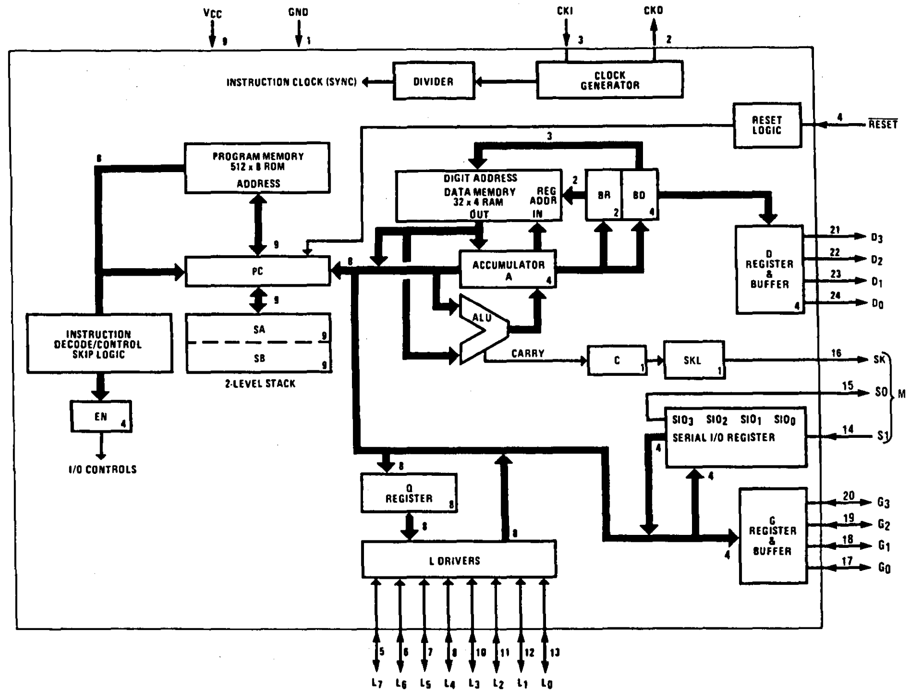
The COP310L and COP311L are exact functional equivalents but extended temperature versions of COP410L and COP411L respectively.

The COP401L should be used for exact emulation.

Features

- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 19 I/O lines (COP410L)
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device
  — COP310L/COP311L (−40°C to +85°C)

Block Diagram



TL/DD/6919-1

FIGURE 1. COP410L

COP410L/COP411L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Power Dissipation	
COP410L	0.75W at 25°C 0.4W at 70°C
COP411L	0.65W at 25°C 0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{RESET}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 3)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

Note 3: TRI-STATE® and LED configurations are excluded.

COP410L/COP411L

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 6.3V$, $V_{OL} = 0.4V$	1.2		mA
	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	0.9		mA
L_0–L_7 Outputs, G_0–G_3 and LSTTL D_0–D_3 Outputs (I_{OL})	$V_{CC} = 6.3V$, $V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	0.4		mA
D_0–D_3 Outputs with High Current Options (I_{OL})	$V_{CC} = 6.3V$, $V_{OL} = 1.0V$	11		mA
	$V_{CC} = 4.5V$, $V_{OL} = 1.0V$	7.5		mA
D_0–D_3 Outputs with Very High Current Options (I_{OL})	$V_{CC} = 6.3V$, $V_{OL} = 1.0V$	22		mA
	$V_{CC} = 4.5V$, $V_{OL} = 1.0V$	15		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V$, $V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	$V_{CC} = 6.3V$, $V_{OH} = 2.0V$	−75	−480	μA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−30	−250	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	$V_{CC} = 6.3V$, $V_{OH} = 2.4V$	−1.4		mA
	$V_{CC} = 4.5V$, $V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 6.0V$, $V_{OH} = 2.0V$	−1.5	−13	mA
LED Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 6.0V$, $V_{OH} = 2.0V$	−3.0	−25	mA
TRI-STATE Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 6.3V$, $V_{OH} = 3.2V$	−0.8		mA
	$V_{CC} = 4.5V$, $V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 6.3V$, $V_{OH} = 3.2V$	−1.6		mA
	$V_{CC} = 4.5V$, $V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V$, $V_{IL} = 0V$	−10	−140	μA
CKO Output				
RAM Power Supply Option Power Requirement	$V_R = 3.3V$		1.5	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pin			1.5	mA

COP310L/COP311L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Power Dissipation	
COP310L	0.75W at 25°C 0.25W at 85°C
COP311L	0.65W at 25°C 0.20W at 85°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Crystal Input				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 3)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

Note 3: TRI-STATE and LED configurations are excluded.

COP310L/COP311L

DC Electrical Characteristics (Continued)

$-40°C \leq T_A \leq +85°C$, $4.5V \leq V_{CC} \leq 5.5V$ unless othewise noted

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0–L_7 Outputs, G_0–G_3 and	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
LSTTL D_0–D_3 Outputs (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
D_0–D_3 Outputs with High	$V_{CC} = 5.5V, V_{OL} = 1.0V$	9		mA
Current Options (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7		mA
D_0–D_3 Outputs with Very	$V_{CC} = 5.5V, V_{OL} = 1.0V$	18		mA
High Current Options (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 1.0V$	14		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	1.5		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	μA
All Outputs (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	μA
Push-Pull Configuration	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−0.7	−15	μA
LED Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.4	−30	μA
TRI-STATE Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−0.6		mA
L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−1.2		mA
L_0–L_7 Outputs, High Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−200	μA
CKO Output				
RAM Power Supply Option Power Requirement	$V_R = 3.3V$		2.0	mA
TRI-STATE Output Leakage Current		−5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pins			1.5	mA

AC Electrical Characteristics

COP410L/411L: 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted
COP310L/311L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time — t_C		16	40	μs
CKI				
Input Frequency — f_I	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 0.5 MHz		500	ns
Fall Time			200	ns

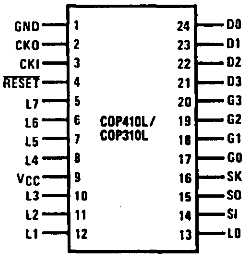
CKI Using RC (÷4)
(Note 1)
	R = 56 kΩ ±5%
	C = 100 pF ±10%			

Instruction Cycle Time		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
INPUTS				
G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

Note 1: Variation due to the device included.

Connection Diagrams

SO Wide and DIP



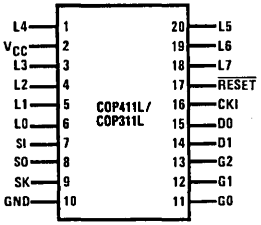
TL/DD/6919-2

Top View

Order Number COP310L-XXX/D or COP410L-XXX/D
See NS Hermetic Package Number D24C

Order Number COP310L-XXX/N or COP410L-XXX/N
See NS Molded Package Number N24A

SO Wide and DIP



TL/DD/6919-3

Top View

Order Number COP311L-XXX/D or COP411L-XXX/D
See NS Hermetic Package Number D24C

Order Number COP311L-XXX/N or COP411L-XXX/N
See NS Molded Package Number N20A

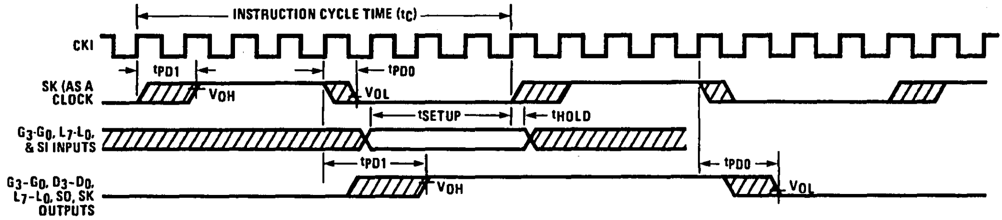
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports (G_2–G_0 for COP411L)
D_3–D_0	4 general purpose outputs (D_1–D_0 for COP411L)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)

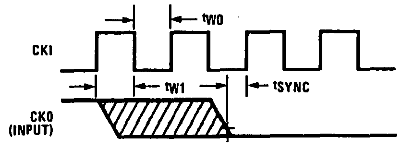
Pin	Description
CKI	System oscillator input
CKO	System oscillator output (or RAM power supply or SYNC input) (COP410L only)
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams


INSTRUCTION CYCLE TIME (tC)
CKI
tPD1
tPD0
SK (AS A CLOCK)
VOH
VOL
tSETUP
tHOLD
G3-G0, L7-L0, & SI INPUTS
tPD1
tPD0
VOH
VOL
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS

TL/DD/6919-4

FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)



TL/DD/6919-5

FIGURE 3a. Synchronization Timing

Functional Description

A block diagram of the COP410L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP410L/COP411L also apply to the COP310L/COP311L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP410L/411L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

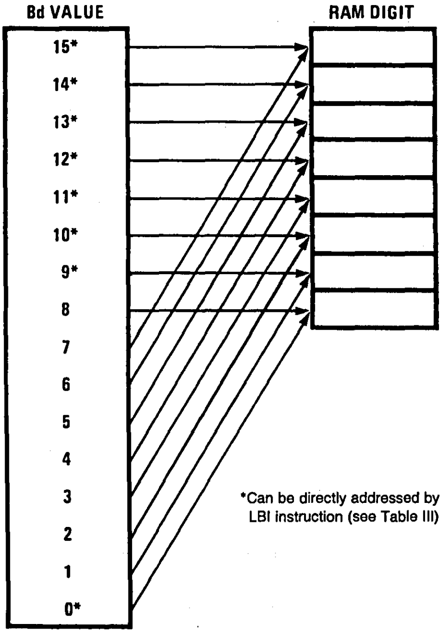
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).



*Can be directly addressed by LBI instruction (see Table III)

TL/DD/6919-6

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP410L/411L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

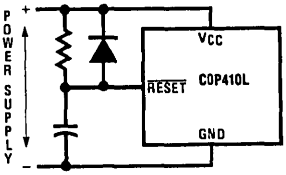
The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP410L/COP411L operation.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below *(Figure 5)*. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



RC ≥ 5 × Power Supply Rise Time TL/DD/6919-7

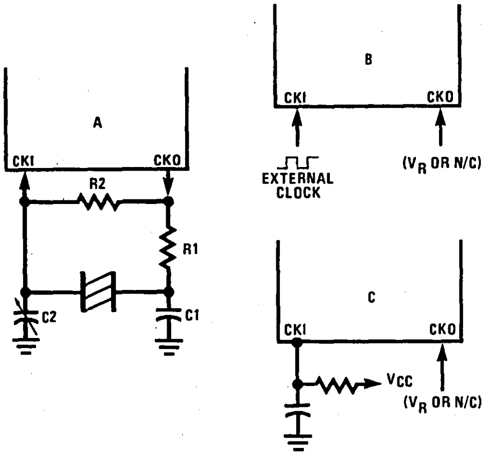
FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/6919-8

Ceramic Resonator Oscillator

Resonator Value	Components Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time in μs
51	100	19 ±15%
82	56	19 ±13%

Note: 200 kΩ ≥ R ≥ 25 kΩ. 360 pF ≥ C ≥ 50 pF. Does not include tolerances.

FIGURE 6. COP410L/411L Oscillator

OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 6.*

a. **Resonator Controlled Oscillator.** CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8. This is not available in the COP411L.

b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 4 to give the instruction frequency time. CKO is now available to be used as the RAM power supply (V_R), or no connection.

Note: No CKO on COP411L.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or no connection.

CKO PIN OPTIONS

In a resonator controlled oscillator system, CKO is used as an output to the resonator network. As an option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using no connection option is appropriate in applications where the COP410L system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. During normal operation, V_R must be within the operating range of the chip with $(V_{CC} - 1) \leq V_R \leq V_{CC}$.
3. V_R must be ≥ 3.3V with V_{CC} off.

I/O OPTIONS

COP410L/411L inputs and outputs have the following optional configurations, illustrated in *Figure 7*:

a. **Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. **Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. **Standard L**—same as **a.**, but may be disabled. Available on L outputs only.

e. **Open Drain L**—same as **b.**, but may be disabled. Available on L outputs only.

f. **LED Direct Drive**—an enhancement mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note: Series current limiting resistors must be used if LEDs are driven directly and higher operating voltage option is selected.

g. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

Functional Description (Continued)

h. An on-chip depletion load device to V_{CC}.

I. A Hi-Z input which must be driven to a "1" or "0" by external components.

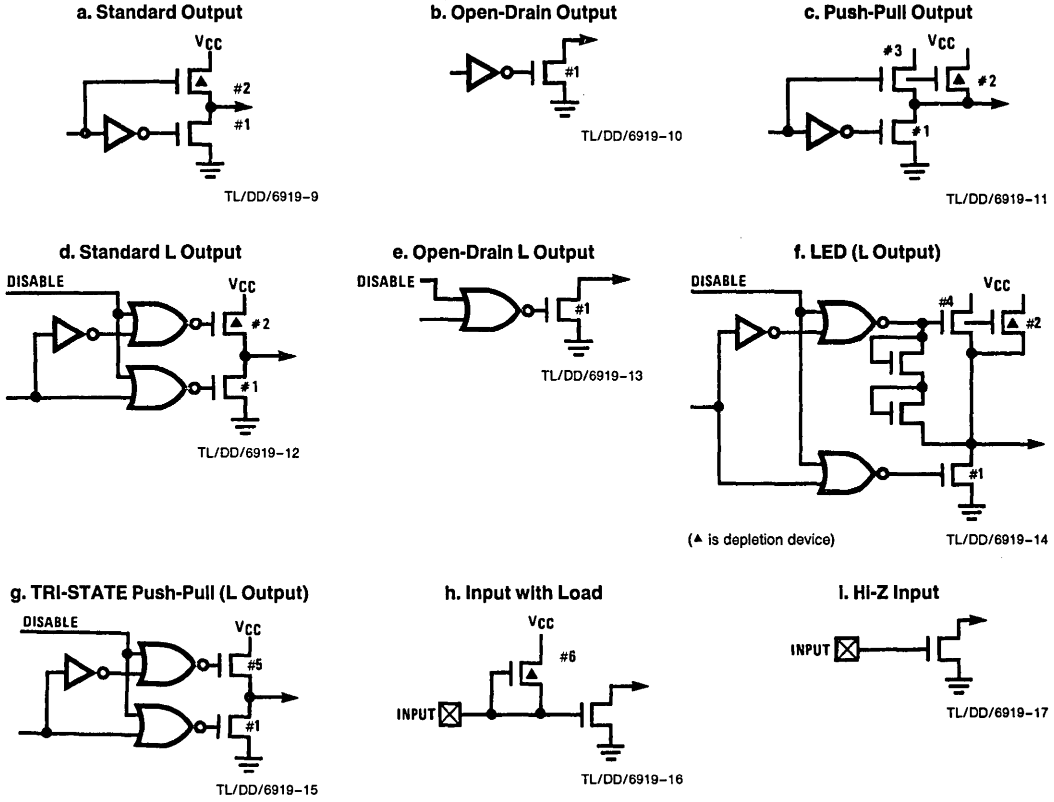
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP410L/411L system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in **d.**, **e.**, **f.**, or **g.**

An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 8*, device 2.) However, when the L port is used as input, the disabled depletion device CANNOT be relied on to source sufficient current to pull an input to a logic "1".

COP411L

If the COP410L is bonded as a 20-pin device, it becomes the COP411L, illustrated in *Figure 2,* COP410L/411L Connection Diagrams. Note that the COP411L does not contain D2, D3, G3, or CKO. Use of this option of course precludes use of D2, D3, G3, and CKO options. All other options are available for the COP411L.


a. Standard Output

TL/DD/6919-9

b. Open-Drain Output

TL/DD/6919-10

c. Push-Pull Output

TL/DD/6919-11

d. Standard L Output

TL/DD/6919-12

e. Open-Drain L Output

TL/DD/6919-13

f. LED (L Output)

(▲ is depletion device) TL/DD/6919-14

g. TRI-STATE Push-Pull (L Output)

TL/DD/6919-15

h. Input with Load

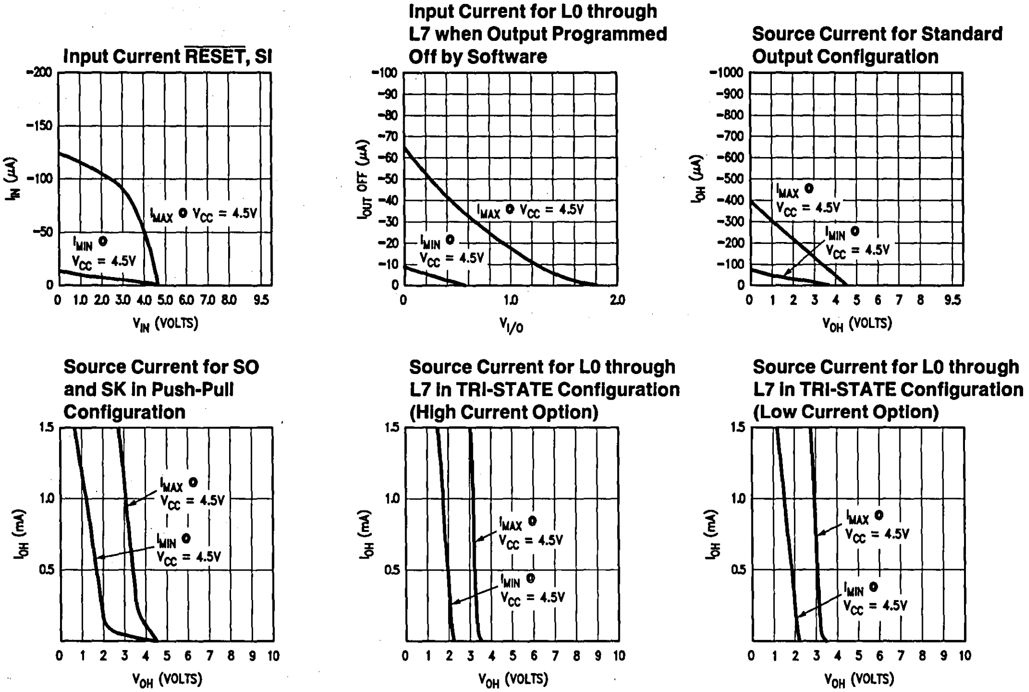
TL/DD/6919-16

I. Hi-Z Input

TL/DD/6919-17

FIGURE 7. Input and Output Configurations

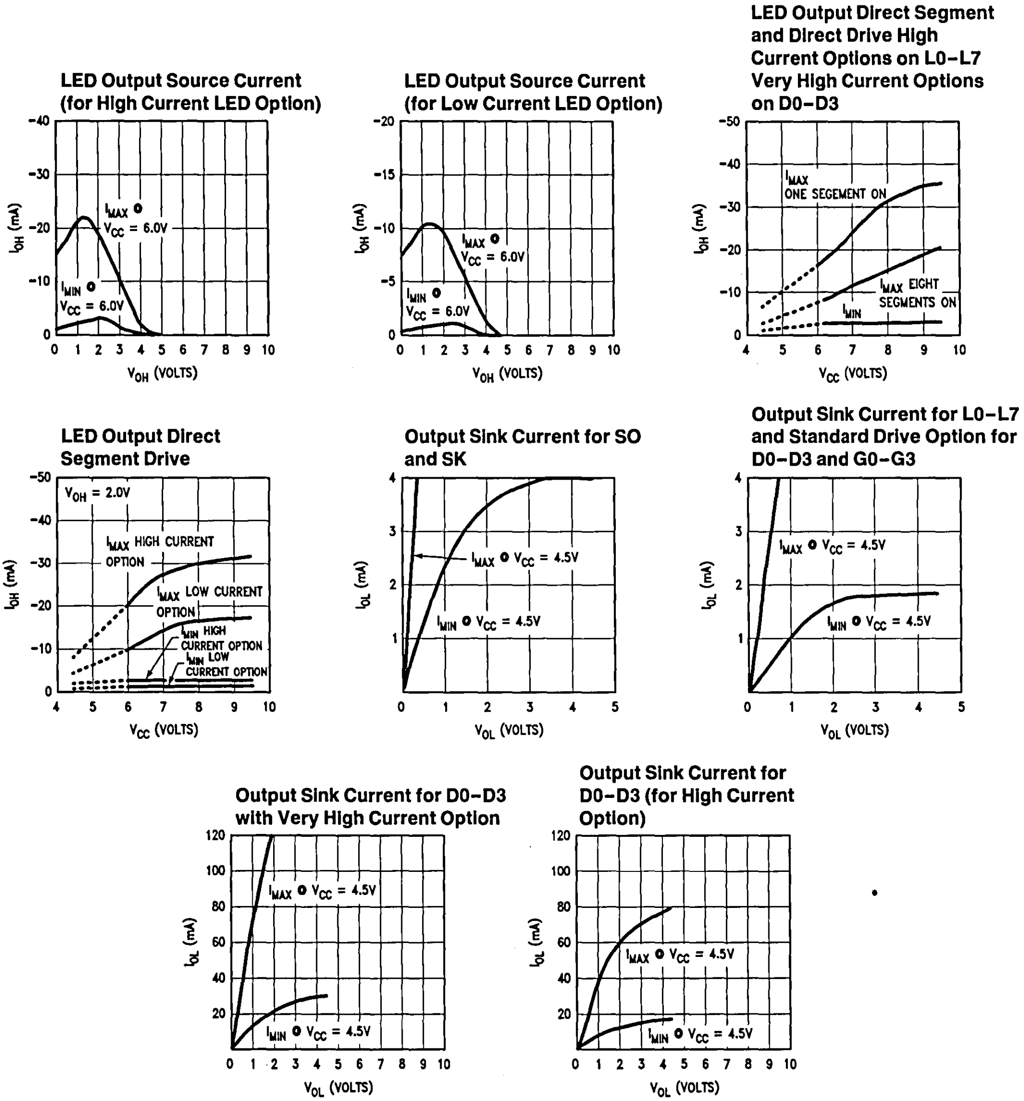
Typical Performance Characteristics



TL/DD/6919-18

FIGURE 8a. COP410L/COP411L I/O DC Current Characteristics

Typical Performance Characteristics (Continued)



LED Output Source Current (for High Current LED Option)

LED Output Source Current (for Low Current LED Option)

LED Output Direct Segment and Direct Drive High Current Options on L0–L7 Very High Current Options on D0–D3

LED Output Direct Segment Drive

Output Sink Current for SO and SK

Output Sink Current for L0–L7 and Standard Drive Option for D0–D3 and G0–G3

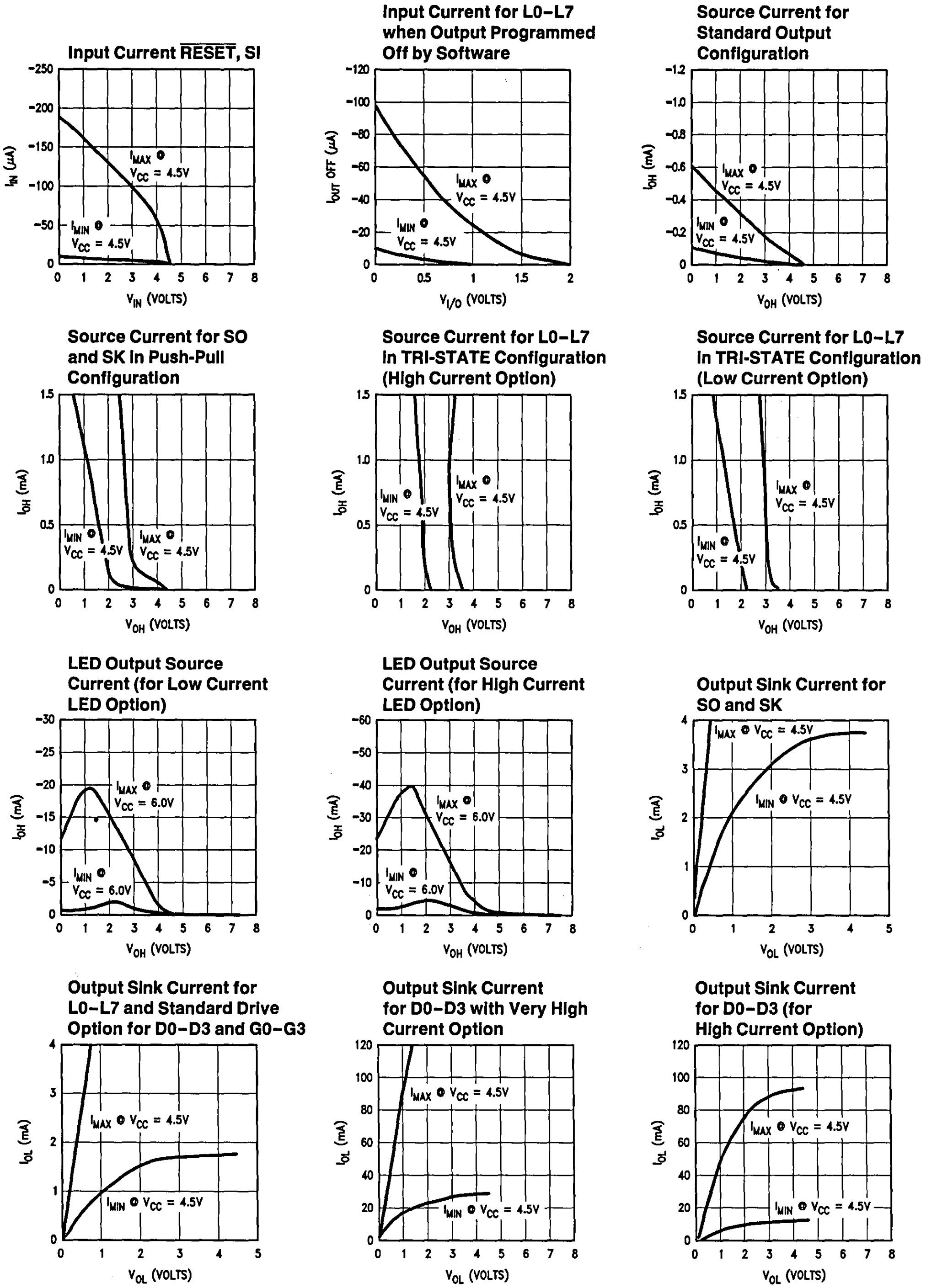
Output Sink Current for D0–D3 with Very High Current Option

Output Sink Current for D0–D3 (for High Current Option)

TL/DD/6919–19

FIGURE 8a. COP410L/COP411L I/O DC Current Characteristics (Continued)

Typical Performance Characteristics (Continued)



Input Current RESET, SI

Input Current for L0–L7 when Output Programmed Off by Software

Source Current for Standard Output Configuration

Source Current for SO and SK in Push-Pull Configuration

Source Current for L0–L7 in TRI-STATE Configuration (High Current Option)

Source Current for L0–L7 in TRI-STATE Configuration (Low Current Option)

LED Output Source Current (for Low Current LED Option)

LED Output Source Current (for High Current LED Option)

Output Sink Current for SO and SK

Output Sink Current for L0–L7 and Standard Drive Option for D0–D3 and G0–G3

Output Sink Current for D0–D3 with Very High Current Option

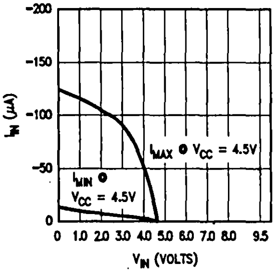
Output Sink Current for D0–D3 (for High Current Option)

TL/DD/6919-20

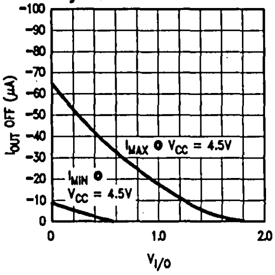
FIGURE 8b. COP310L/COP311L Input/Output Characteristics

Typical Performance Characteristics

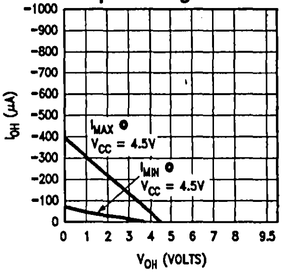
Current for Inputs with Load Device



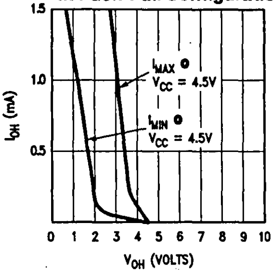
Input Current for L_0 through L_7 when Output Programmed Off by Software



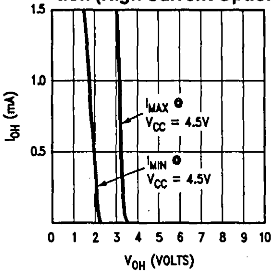
Source Current for Standard Output Configuration



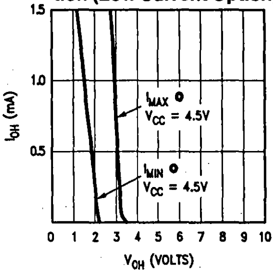
Source Current for SO and SK in Push-Pull Configuration



Source Current for L_0 through L_7 in TRI-STATE Configuration (High Current Option)

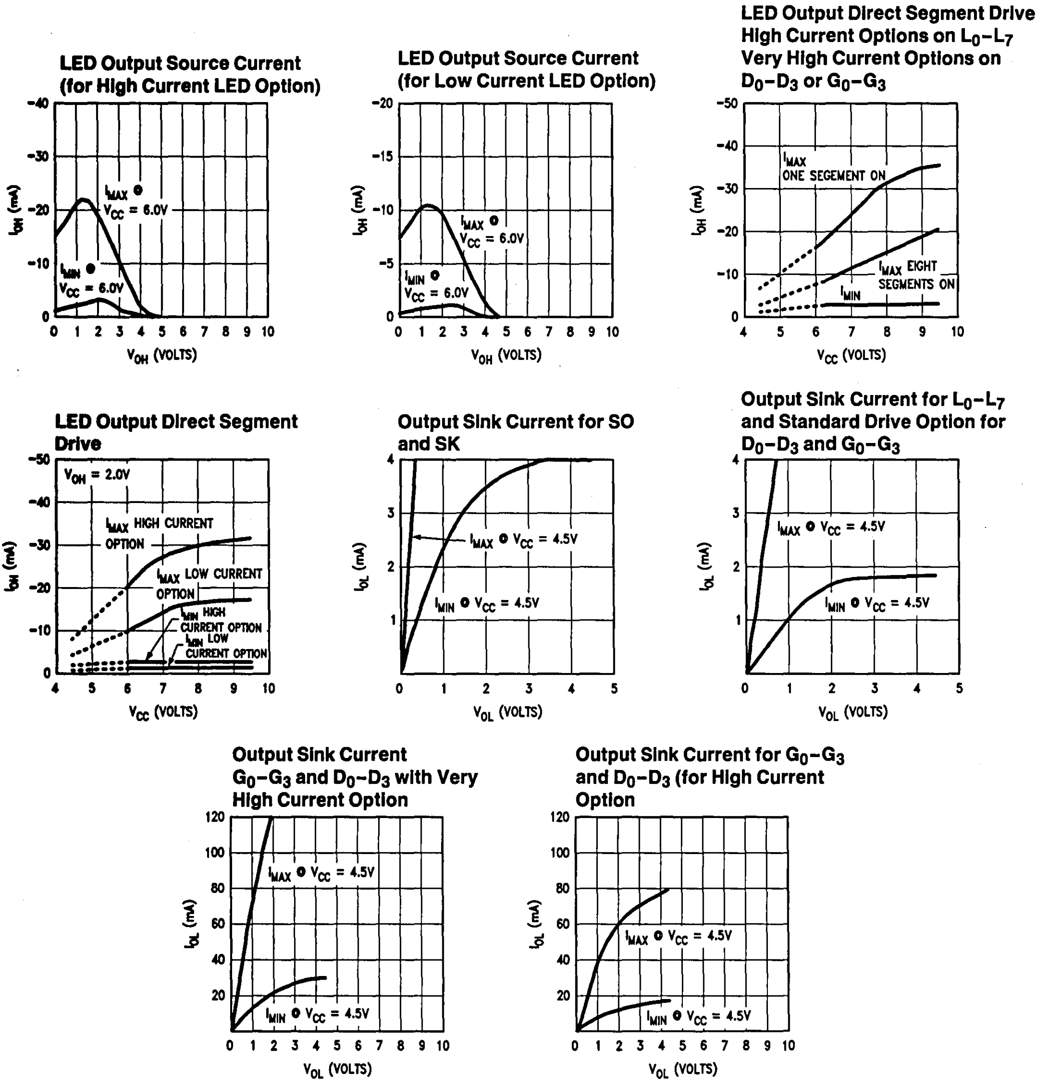


Source Current for L_0 through L_7 in TRI-STATE configuration (Low Current Option)



TL/DD/6919-18

Typical Performance Characteristics (Continued)



LED Output Source Current (for High Current LED Option)

LED Output Source Current (for Low Current LED Option)

LED Output Direct Segment Drive High Current Options on L_0–L_7 Very High Current Options on D_0–D_3 or G_0–G_3

LED Output Direct Segment Drive

Output Sink Current for SO and SK

Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3

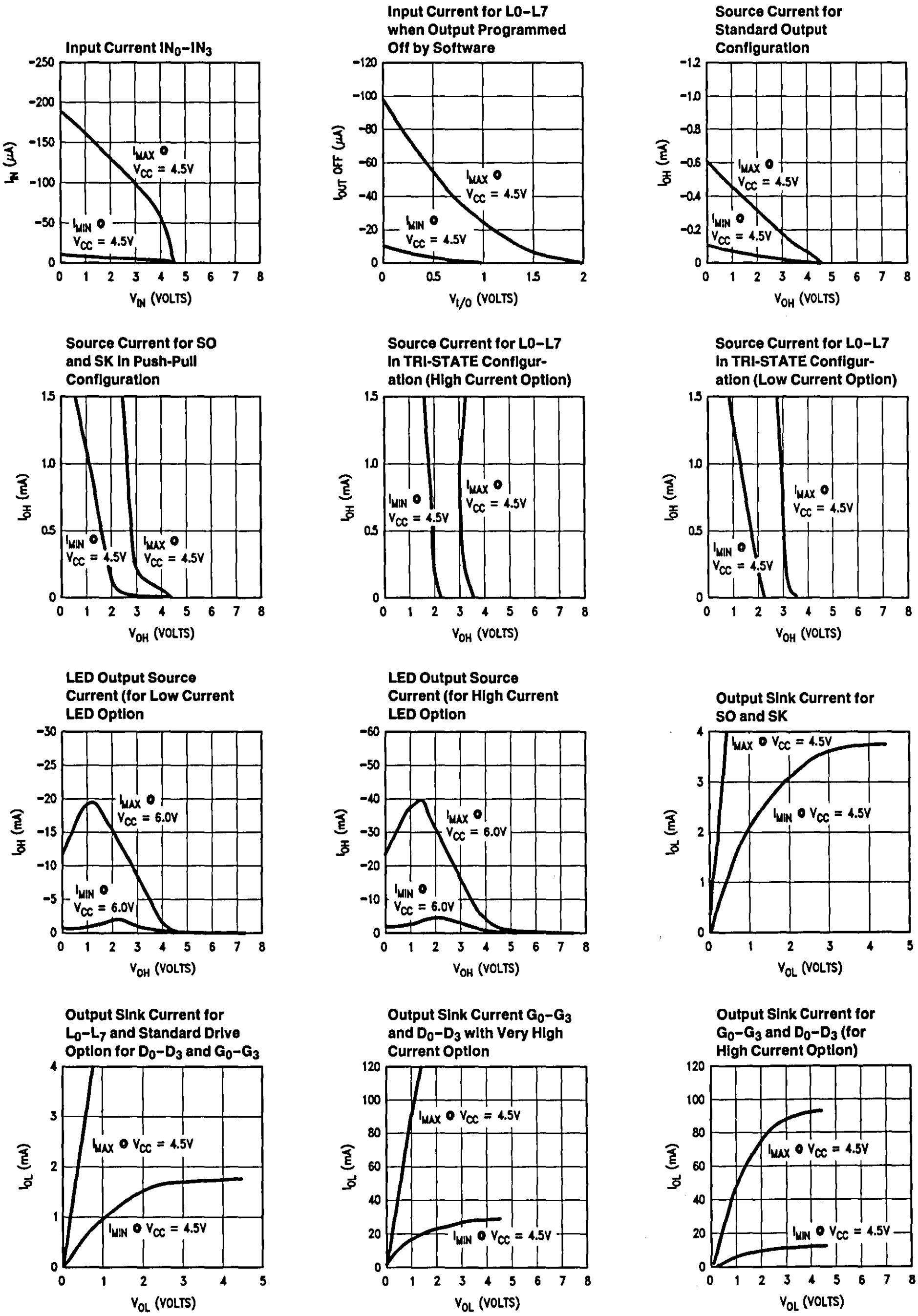
Output Sink Current G_0–G_3 and D_0–D_3 with Very High Current Option

Output Sink Current for G_0–G_3 and D_0–D_3 (for High Current Option

TL/DD/6919-19

FIGURE 8a. COP410L/COP411L Input/Output Characteristics

Typical Performance Characteristics (Continued)



Input Current IN₀–IN₃

Input Current for L0–L7 when Output Programmed Off by Software

Source Current for Standard Output Configuration

Source Current for SO and SK in Push-Pull Configuration

Source Current for L0–L7 in TRI-STATE Configur-ation (High Current Option)

Source Current for L0–L7 in TRI-STATE Configur-ation (Low Current Option)

LED Output Source Current (for Low Current LED Option

LED Output Source Current (for High Current LED Option

Output Sink Current for SO and SK

Output Sink Current for L₀–L₇ and Standard Drive Option for D₀–D₃ and G₀–G₃

Output Sink Current G₀–G₃ and D₀–D₃ with Very High Current Option

Output Sink Current for G₀–G₃ and D₀–D₃ (for High Current Option)

TL/DD/6919-20

FIGURE 8b. COP310L/COP311L Input/Output Characteristics

COP410L/411L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410L/411L instruction set.

TABLE II. COP410L/411L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP410L/411L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	0101 y	A + y → A	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP410L/411L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM (PC_8,A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– – –	|0110|000|a_8| |$a_{7:0}$|	a → PC	None	Jump
JP	a	– – – –	|1| $a_{6:0}$ | (pages 2,3 only) or |11| $a_{5:0}$ | (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 3)
JSRP	a	– –	|10| $a_{5:0}$ |	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– – –	|0110|100|a_8| |$a_{7:0}$|	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	|0011|0011| |0011|1100|	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
LD	r	-5	|00|r|0101|	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	|1011|1111|	ROM(PC_8,A,M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0 1 2 3	4C 45 42 43	|0100|1100| |0100|0101| |0100|0010| |0100|0011|	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	|0100|1101| |0100|0111| |0100|0110| |0100|1011|	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	|0111| y |	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	|00|r|0110|	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	|0010|0011| |1011|1111|	RAM(3,15) ←→ A	None	Exchange A with RAM (3,15)
XDS	r	-7	|00|r|0111|	RAM(B) ←→ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	|00|r|0100|	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r

Instruction Set (Continued)

TABLE III. COP410L/411L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	- -	00 r (d − 1) (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6-	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0 = 0$	
	1	11	0001 0001		$G_1 = 0$	
	2	03	0000 0011		$G_2 = 0$	
	3	13	0001 0011		$G_3 = 0$	
SKMBZ	0	01	0000 0001		RAM(B)$_0$ = 0	Skip if RAM Bit is Zero
	1	11	0001 0001		RAM(B)$_1$ = 0	
	2	03	0000 0011		RAM(B)$_2$ = 0	
	3	13	0001 0011		RAM(B)$_3$ = 0	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP410L/411L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP410L/411L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or 7 will access data in the next group of 4 pages.

Option List

The COP410L/411L mask-programmable options are assigned numbers which correspond with the COP410L pins.

The following is a list of COP410L options. The LED Direct Drive option on the L Lines cannot be used if higher V_{CC} option is selected. When specifying a COP411L chip, Option 2 must be set to 3, Options 20, 21, and 22 to 0. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Output (no option available for COP411L)
= 0: Clock output to ceramic resonator
= 1: Pin is RAM power supply (V_R) input
= 3: No connection

Option 3: CKI Input
= 0: Oscillator input divided by 8 (500 kHz max)
= 1: Single-pin RC controlled oscillator divided by 4
= 2: External Schmitt trigger level clock divided by 4

Option 4: $\overline{\text{RESET}}$ Input
= 0: Load device to V_{CC}
= 1: Hi-Z input

Option 5: L_7 Driver
= 0: Standard output
= 1: Open-drain output
= 2: High current LED direct segment drive output
= 3: High current TRI-STATE push-pull output
= 4: Low-current LED direct segment drive output
= 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: Operating voltage

	COP41XL	COP31XL
= 0:	+4.5V to +6.3V	+4.5V to +5.5V

Option 10: L_3 Driver
same as Option 5

Option 11: L_2 Driver
same as Option 5

Option 12: L_1 Driver
same as Option 5

Option 13: L_0 Driver
same as Option 5

Option 14: SI Input
= 0: load device to V_{CC}
= 1: Hi-Z input

Option 15: SO Driver
= 0: Standard Output
= 1: Open-drain output
= 2: Push-pull output

Option 16: SK Driver
same as Option 15

Option List (Continued)

Option 17: G_0 I/O Port
= 0: Standard output
= 1: Open-drain output

Option 18: G_1 I/O Port
same as Option 17

Option 19: G_2 I/O Port
same as Option 17

Option 20: G_3 I/O Port (no option available for COP411L)
same as Option 17

Option 21: D_3 Output (no option available for COP411L)
= 0: Very-high sink current standard output
= 1: Very-high sink current open-drain output
= 2: High sink current standard output
= 3: High sink current open-drain output
= 4: Standard LSTTL output (fanout = 1)
= 5: Open-drain LSTTL output (fanout = 1)

Option 22: D_2 Output (no option available for COP411L)
same as Option 21

Option 23: D_1 Output
same as Option 21

Option 24: D_0 Output
same as Option 21

Option 25: L Input Levels
= 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
= 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 26: G Input Levels
same as Option 25

Option 27: SI Input Levels
same as Option 25

Option 28: COP Bonding
= 0: COP410L (24-pin device)
= 1: COP411L (20-pin device)
= 2: Both 24- and 20-pin versions

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP410L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

Option Table

The following option information is to be sent to National along with the EPROM.

Option Data

OPTION 1 VALUE = ___0___ IS: GROUND PIN
OPTION 2 VALUE = ________ IS: CKO PIN
OPTION 3 VALUE = ________ IS: CKI INPUT
OPTION 4 VALUE = ________ IS: RESET INPUT
OPTION 5 VALUE = ________ IS: L(7) DRIVER
OPTION 6 VALUE = ________ IS: L(6) DRIVER
OPTION 7 VALUE = ________ IS: L(5) DRIVER
OPTION 8 VALUE = ________ IS: L(4) DRIVER
OPTION 9 VALUE = ___0___ IS: V_{CC} PIN
OPTION 10 VALUE = ________ IS: L(3) DRIVER
OPTION 11 VALUE = ________ IS: L(2) DRIVER
OPTION 12 VALUE = ________ IS: L(1) DRIVER
OPTION 13 VALUE = ________ IS: L(0) DRIVER
OPTION 14 VALUE = ________ IS: SI INPUT

Option Data

OPTION 15 VALUE = ________ IS: SO DRIVER
OPTION 16 VALUE = ________ IS: SK DRIVER
OPTION 17 VALUE = ________ IS: G_0 I/O PORT
OPTION 18 VALUE = ________ IS: G_1 I/O PORT
OPTION 19 VALUE = ________ IS: G_2 I/O PORT
OPTION 20 VALUE = ________ IS: G_3 I/O PORT
OPTION 21 VALUE = ________ IS: D_3 OUTPUT
OPTION 22 VALUE = ________ IS: D_2 OUTPUT
OPTION 23 VALUE = ________ IS: D_1 OUTPUT
OPTION 24 VALUE = ________ IS: D_0 OUTPUT
OPTION 25 VALUE = ________ IS: L INPUT LEVELS
OPTION 26 VALUE = ________ IS: G INPUT LEVELS
OPTION 27 VALUE = ________ IS: SI INPUT LEVELS
OPTION 28 VALUE = ________ IS: COPS BONDING

National Semiconductor

COP413L/COP313L Single Chip Microcontrollers

General Description

The COP413L and COP313L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These Control Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, 15 I/O lines with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Control Oriented Processor at a very low end-product cost.

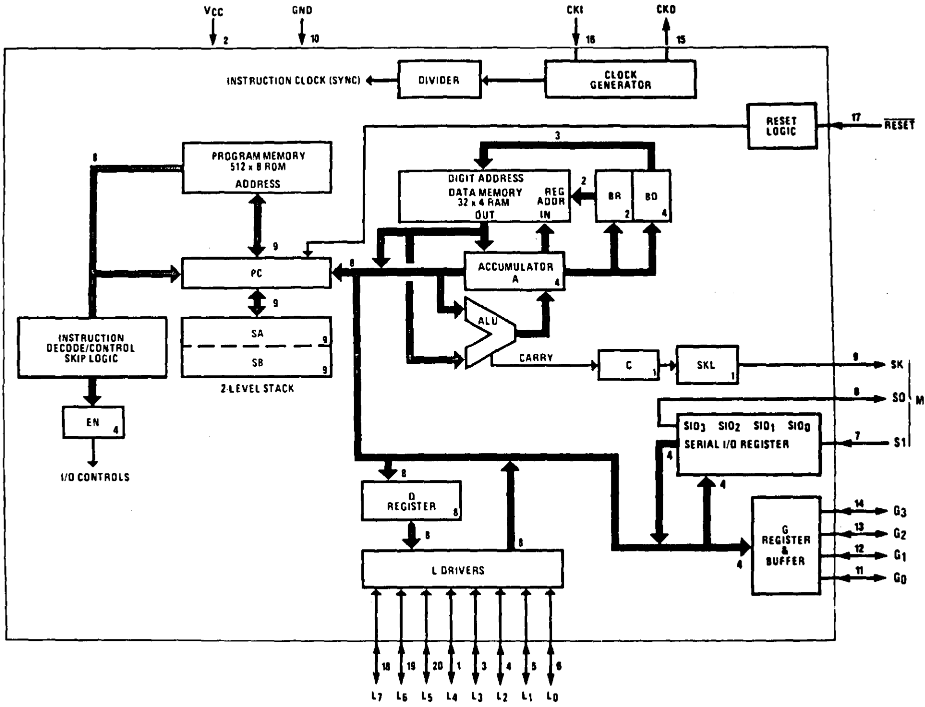
The COP313L is an exact functional equivalent but extended temperature version of the COP413L.

The COP401L-R13 and COP410L-X13 should be used for exact emulation.

Features

- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-Level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V-6.3V)
- Low current drain (6 mA max.)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose outputs
- High noise immunity inputs ($V_{IL}=1.2V$, $V_{IH}=3.6V$)
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device COP313L (−40°C to +85°C)

Block Diagram



TL/DD/8371-1

FIGURE 1

COP413L Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.3 to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation COP413L	0.3 Watt at 70°C
Total Source Current	25 mA
Total Sink Current	25 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C, 4.5V \le V_{CC} \le 6.3V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Power Supply Ripple	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})		3.0		V
Logic Low (V_{IL})			0.4	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low			0.6	V
SO Input Level (Test Mode)	(Note 2)	2.5		V
SI Input Level				
Logic High	(TTL Level)	2.0		V
Logic Low			0.8	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance			7	pF
Reset Input Leakage		−1	+1	μA
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{OL}=0.4V$	0.9		mA
L0–L7 Outputs, G0–G3	$V_{OL}=0.4V$	0.4		mA
CKO (I_{OL})	$V_{OL}=0.4V$	0.2		mA
Output Source Current				
L0–L7 and G0–G3	$V_{OH}=2.4V$	−25		μA
SO and SK Outputs (I_{OH})	$V_{OH}=1.0V$	−1.2		mA
Push-Pull	$V_{OH}=2.4V$	−25		μA
SI Input Load Source Current	$V_{IL}=0V$	−10	−140	μA
Total Sink Current Allowed				
L7–L4, G Port			4	mA
L3–L0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

COP413L/COP313L

COP313L Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.3 to +7V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation COP313L	**0.20 Watt at 85°C**
Total Source Current	**25 mA**
Total Sink Current	**25 mA**

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})		3.0		V
Logic Low (V_{IL})			0.3	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low			0.4	V
SO Input (Test Mode)	(Note 2)	2.5		V
SI Input Level				
Logic High	(TTL Level)	2.2		V
Logic Low			0.6	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance			7	pF
Reset Input Leakage		−2	+2	μA
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{OL}=0.4V$	0.8		mA
L0–L7 Outputs, G0–G3 (I_{OL})	$V_{OL}=0.4V$	0.4		mA
CKO (I_{OL})	$V_{OL}=0.4V$	0.2		mA
Output Source Current				
L0–L7 and G0–G3	$V_{OH}=2.4V$	−23		μA
SO and SK Outputs (I_{OH})	$V_{OH}=1.0V$	−1.0		mA
(Push-Pull)	$V_{OH}=2.4V$	−23		μA
SI Input Load Source Current	$V_{IL}=0V$	−10	−200	μA
Total Sink Current Allowed				
L7–L4, G Port			4	mA
L3–L0			4	mA
Any Other Pin			1.5	mA
Total Source Current Allowed				
Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

AC Electrical Characteristics COP413L: 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 6.3V COP313L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time - t_c		16	40	μs
CKI				
Input Frequency - fi	÷8 Mode	0.2	0.5	MHz
Duty Cycle		30	60	%
Rise Time	fi = 0.5 MHz		500	ns
Fall Time			200	ns

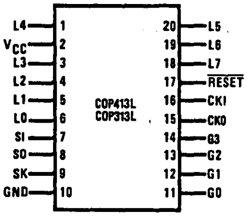
CKI Using RC (÷4)	R = 56 kΩ ±5%
	C = 100 pF ±10%			

Instruction Cycle Time (Note 1)		16	28	μs
Inputs:				
G3–G0, L7–L0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3	1.3	μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs tpd1, tpd0			4.0	μs
All Other Outputs tpd1, tpd0			5.6	μs

Note 1: Variation due to the device included.

Connection Diagram

S.O. Wide and DIP



TL/DD/8371-2

FIGURE 2

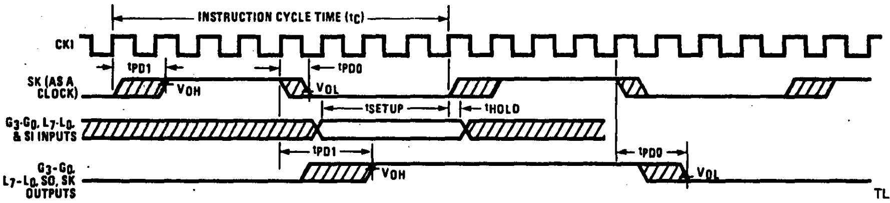
Order Number COP313L-XXX/D or COP413L-XXX/D See NS Hermetic Package Number D20A

Order Number COP313L-XXX/WM or COP413L-XXX/WM See NS Surface Mount Package Number M20B

Order Number COP313L-XXX/N or COP413L-XXX/N See NS Molded Package Number N20A

Pin Descriptions

Pin	Description
L7–L0	8-bit bidirectional I/O port
G3–G0	4-bit bidirectional I/O port
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output or NC
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power Supply
GND	Ground


INSTRUCTION CYCLE TIME (t_C)
CKI
SK (AS A CLOCK)
tPD1
VOH
tPD0
VOL
tSETUP
tHOLD
G3-G0, L7-L0, & SI INPUTS
G3-G0, L7-L0, SO, SK OUTPUTS
tPD1
VOH
tPD0
VOL

TL/DD/8371-3

FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)

Functional Description

A block diagram of the COP413L is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP413L also apply to the COP313L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP413L instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction.

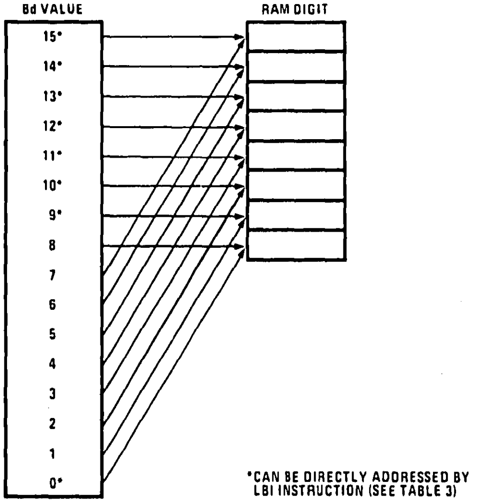
The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP413L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general purpose bidirectional I/O ports.



TL/DD/8371-4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0 selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting with each instruction cycle time. The data present at SO goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP413L operation.

Functional Description (Continued)

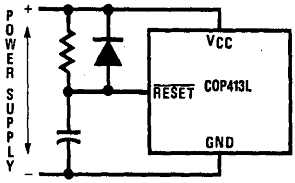
TABLE I. Enable Register Modes - Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below (*Figure 5*). The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



RC > 5 x POWER SUPPLY RISE TIME

TL/DD/8371-5

FIGURE 5. Power-Up Clear Circuit

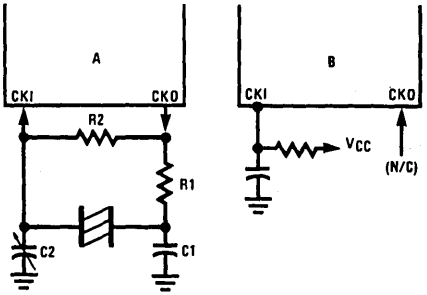
Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

OSCILLATOR

There are two basic clock oscillator configurations available as shown by *Figure 6.*

a. Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.

b. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO becomes no connection.



TL/DD/8371-6

FIGURE 6. COP413L Oscillator

Ceramic Resonator Oscillator

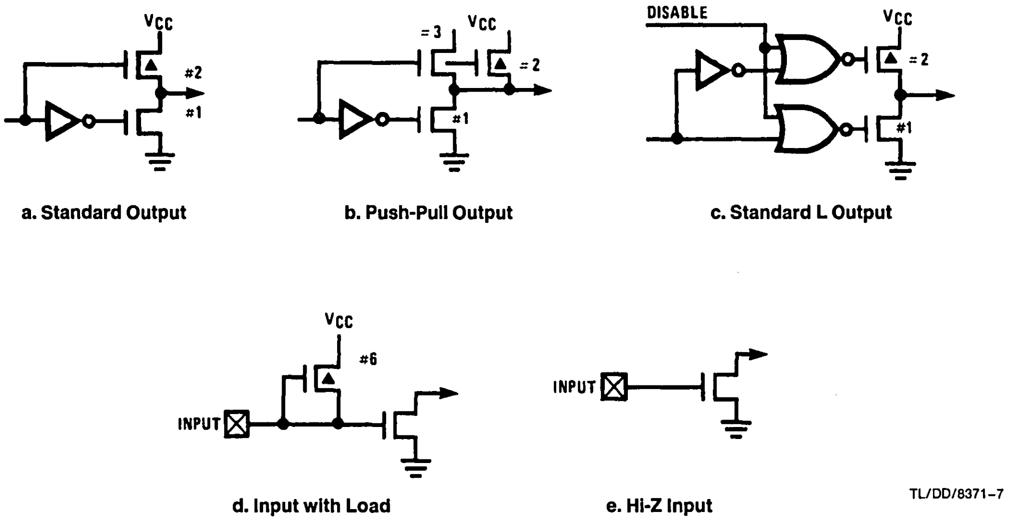
Resonator Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (in μs)
51	100	19 ± 15%
82	56	19 ± 13%

Note: 200 kΩ ≥ R ≥ 25 kΩ
220 pF ≥ C ≥ 50 pF

Functional Description (Continued)



a. Standard Output

b. Push-Pull Output

c. Standard L Output

d. Input with Load

e. Hi-Z Input

FIGURE 7. Input and Output Configurations

I/O CONFIGURATIONS

COP413L inputs and outputs have the following configurations, illustrated in *Figure 7*:

a. G0–G3—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}.

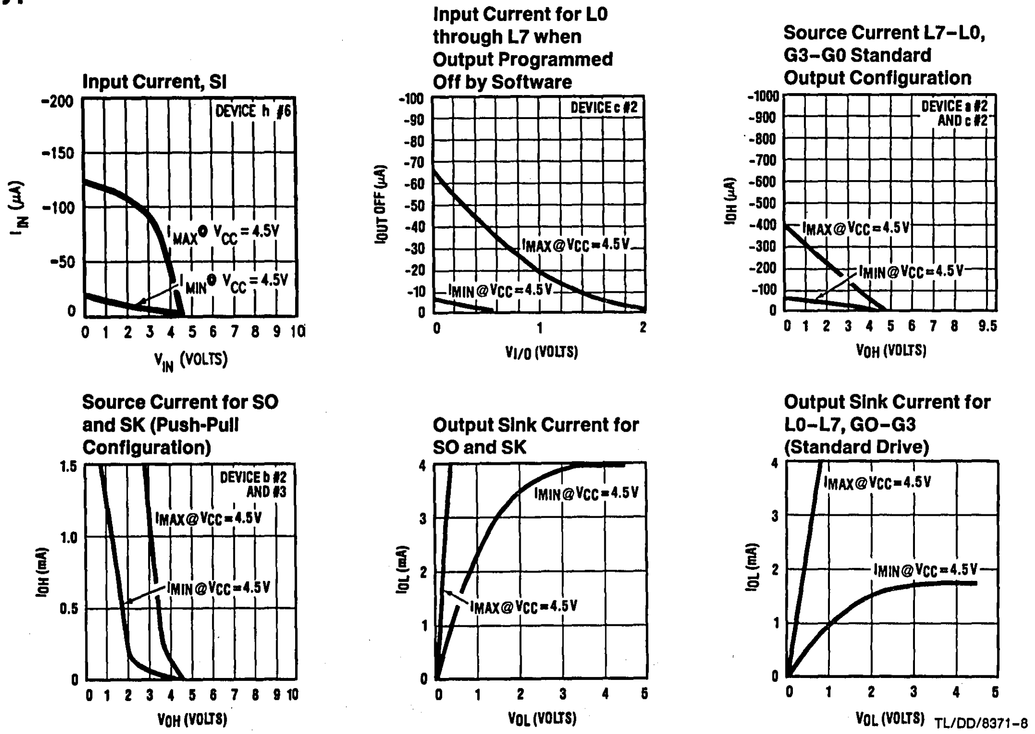
b. SO, SK—an enhancement mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

c. L0–L7—same as a., but may be disabled.

d. SI has on-chip depletion load device to V_{CC}.

e. $\overline{\text{RESET}}$ has a Hi-Z input which must be driven to a "1" or "0" by external components.

Typical Performance Characteristics


Input Current, SI

Input Current for L0 through L7 when Output Programmed Off by Software

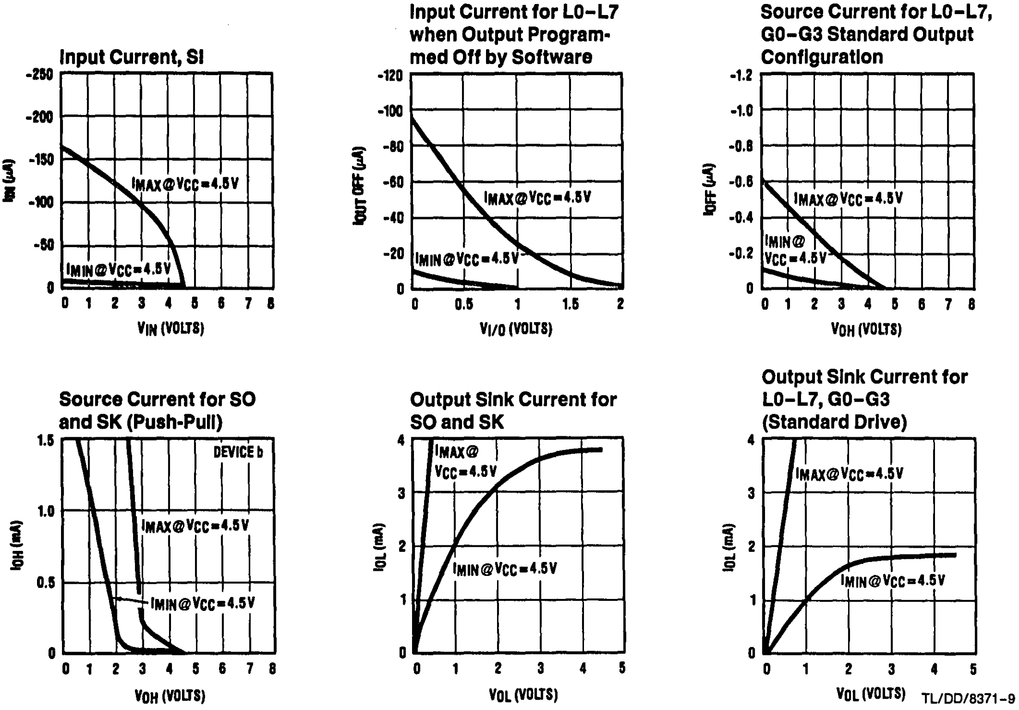
Source Current L7–L0, G3–G0 Standard Output Configuration

Source Current for SO and SK (Push-Pull Configuration)

Output Sink Current for SO and SK

Output Sink Current for L0–L7, GO–G3 (Standard Drive)

FIGURE 8a. COP413L I/O DC Current Characteristics


Input Current, SI

Input Current for L0–L7 when Output Programmed Off by Software

Source Current for L0–L7, G0–G3 Standard Output Configuration

Source Current for SO and SK (Push-Pull)

Output Sink Current for SO and SK

Output Sink Current for L0–L7, G0–G3 (Standard Drive)

FIGURE 8b. COP313L I/O DC Current Characteristics

COP413L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table. Table III provides the mnemonic, operand, machine code data flow, skip conditions and description associated with each instruction in the COP413L instruction set.

TABLE II. COP413L Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE® I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic Controlled Clock Output
Instruction Operand Symbols	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
Operational Symbols	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011|0000	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011|0001	A+RAM(B) → A	None	Add RAM to A
AISC	y	5–	0101| y	A+y → A	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	0000|0000	0 → A	None	Clear A
COMP		40	0100|0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100|0100	None	None	No Operation
RC		32	0011|0010	"0" → C	None	Reset C
SC		22	0010|0010	"1" → C	None	Set C
XOR		02	0000|0010	A⊕RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111|1111	ROM(PC_8,A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– –	0110|000|a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	– –	1| $a_{6:0}$ (pages 2, 3 only) or 11| $a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within-Page (Note 3)
JSRP	a	–	10| $a_{5:0}$	PC+1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– –	0110|100|a_8 $a_{7:0}$	PC+1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100|1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100|1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011|0011 0011|1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
LD	r	–5	00|r|0101	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011|1111	ROM(PC_8, A,M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	0100|1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100|0101	0 → $RAM(B)_1$		
	2	42	0100|0010	0 → $RAM(B)_2$		
	3	43	0100|0011	0 → $RAM(B)_3$		
SMB	0	4D	0100|1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100|0111	1 → $RAM(B)_1$		
	2	46	0100|0110	1 → $RAM(B)_2$		
	3	4B	0100|1011	1 → $RAM(B)_3$		

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7–	0111 y	y → RAM(B) Bd+1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110	RAM(B) ←→ A Br⊕r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	0010 0011 1011 1111	RAM(3,15) ←→ A	None	Exchange A with RAM (3,15)
XDS	r	–7	00 r 0111	RAM(B) ←→ A Bd–1 → Bd Br⊕r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	–4	00 r 0100	RAM(B) ←→ A Bd+1 → Bd Br⊕r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	–	00 r (d–1) (d=0,9:15)	r,d → B	Skip until not a LBI	Load B immediate with r,d (Note 5)
LEI	y	33 6–	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C="1"	Skip if C is True
SKE		21	0010 0001		A=RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0=0$	
	1	11	0001 0001		$G_1=0$	
	2	03	0000 0011		$G_2=0$	
	3	13	0001 0011		$G_3=0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1=0$	
	2	03	0000 0011		$RAM(B)_2=0$	
	3	13	0001 0011		$RAM(B)_3=0$	

COP413L Instruction Set (Continued)

TABLE III. COP413L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 1110	$L_{3:0}$ → A		
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicity defined (e.g., Br and Bd are explicitly defined) Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1* e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP413L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM (B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP413L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or will access data in the next group of 4 pages.

Description of Selected Instructions (Continued)

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmable COP413L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

Option List

The option selected must be sent in with the EPROM of ROM Code for a Mask order of 413L. Make xerox copy of the table, select the appropriate option, and send it in with the EPROM.

COP 413L/COP 313L

Option 1: Oscillator Selection

=0 Ceramic Resonator or external input frequency divided by 8. CKO is oscillator output.

=1 Single pin RC controlled oscillator divided by 4. CKO is no connection.

NOTE:

The following option information is to be sent to National along with the EPROM

Option 1: Value = _______ is: Oscillator Selection

National Semiconductor

COP413C/COP413CH/COP313C/COP313CH Single-Chip CMOS Microcontrollers

General Description

The COP413C, COP413CH, COP313C, and COP313CH fully static, single-chip CMOS microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon-gate CMOS technology. These controller-oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP413CH is identical to the COP413C except for operating voltage and frequency. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide a customized controller-oriented processor at a low end-product cost.

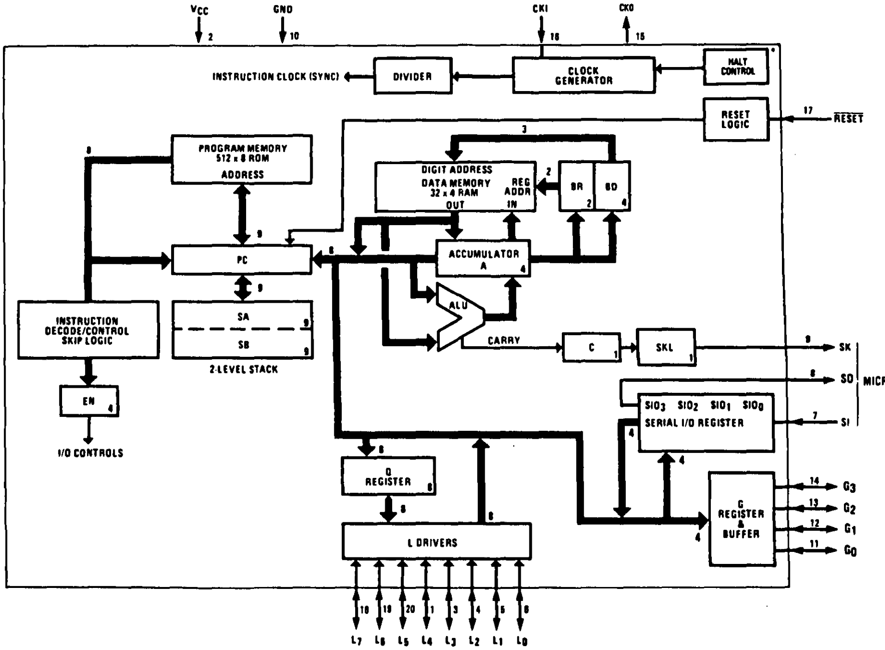
The COP313C/COP313CH is the extended temperature range version of the COP413C/COP413CH.

For emulation use the ROMless COP404C.

Features

- Lowest power dissipation (40 μW typical)
- Low cost
- Power-saving HALT Mode
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-level subroutine stack
- DC to 4 μs instruction time
- Single supply operation (3V to 5.5V)
- General purpose and TRI-STATE® outputs
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- Software/hardware compatible with other members of the COP400 family
- Extended temperature (−40°C to +85°C) devices available

Block Diagram



TL/DD/8537–1

FIGURE 1. COP413C/413CH

COP413C/COP413CH

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Conditions	COP413C		COP413CH		Units
		Min	Max	Min	Max	
Operating Voltage		3.0	5.5	4.5	5.5	V
Power Supply Ripple (Note 4)			0.1 V_{CC}		0.1 V_{CC}	V
Supply Current (Note 1)	V_{CC} = 5.0V, t_c = Min V_{CC} = 3.0V, t_c = Min (t_c is inst. cycle)		500 300		2000	μA μA
HALT Mode Current (Note 2)	V_{CC} = 5.0V, F_I = 0 kHz V_{CC} = 3.0V, F_I = 0 kHz		30 10		30	μA μA
Input Voltage Levels						
$\overline{\text{RESET}}$, CKI						
Logic High		0.9 V_{CC}		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}		0.1 V_{CC}	V
All Other Inputs						
Logic High		0.7 V_{CC}		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}		0.2 V_{CC}	V
$\overline{\text{RESET}}$, SI Input Leakage		−1	+1	−1	+1	μA
Input Capacitance			7		7	pF
Output Voltage Levels						
(SO, SK, L Port)						
Logic High	I_{OH} = −10 μA	V_{CC} − 0.2		V_{CC} − 0.2		V
Logic Low	I_{OL} = 10 μA		0.2		0.2	V
Output Current Levels						
Sink (Note 3)	V_{CC} = Min, V_{OUT} = V_{CC}	0.2		1.2		mA
Source (SO, SK, L Port)	V_{CC} = Min, V_{OUT} = 0V	−0.1		−0.5		mA
Source (G Port)	V_{CC} = Min, V_{OUT} = 0V	−8	−150	−30	−330	μA
Allowable Sink/Source Current Per Pin (Note 3)			5		5	mA
TRI-STATE Leakage Current		−2	+2	−2	+2	μA

COP413C/COP413CH

AC Electrical Characteristics 0°C ≤ T_A ≤ 70°C unless otherwise specified

Parameter	Conditions	COP413C		COP413CH		Units
		Min	Max	Min	Max	
Instruction Cycle Time		16	DC	4	DC	μs
Operating CKI Frequency	÷8 Mode	DC	500	DC	2000	kHz
Instruction Cycle Time RC Oscillator ÷ 4	R = 30k ±5%, V_{CC} = 5V C = 82 pF ± 5%			8	16	μs
Instruction Cycle Time RC Oscillator ÷ 4 (Note 6)	R = 56k ±5%, V_{CC} = 5V C = 100 pF ± 5%	16	32	16	32	μs
Duty Cycle (Note 5)	Fi = Max freq ext clk	40	60	40	60	%
Rise Time (Note 5)	Fi = Max freq ext clk		60		60	ns
Fall Time (Note 5)	Fi = Max freq ext clk		40		40	ns
Inputs (See *Figure 3*)						
t_{SETUP}	G Inputs	tc/4 + 2.8		tc/4 + 0.7		μs
	SI Input	1.2		0.3		μs
	L Inputs	6.8		1.7		μs
t_{HOLD}		1.0		0.25		μs
Output Propagation Delay t_{PD1}, t_{PD0}	V_{OUT} = 1.5, C_L = 100 pF R_L = 5k		4.0		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating.

Note 3: SO output sink current must be limited to keep V_{OL} less tha 0.2 V_{CC} when part is running in order to prevent entering test mode.

Note 4: Voltage change must be less than 0.5V in a 1 ms period.

Note 5: This parameter is only sampled and not 100% tested.

Note 6: Variation due to the device included.

COP313C/COP313CH

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	6V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	COP313C		COP313CH		Units
		Min	Max	Min	Max	
Operating Voltage		3.0	5.5	4.5	5.5	V
Power Supply Ripple (Note 4)			$0.1\ V_{CC}$		$0.1\ V_{CC}$	V
Supply Current (Note 1)	$V_{CC} = 5.0V$, t_c = Min		600		2500	µA
	$V_{CC} = 3.0V$, t_c = Min		360			µA
	(t_c is inst. cycle)					
Halt Mode Current (Note 2)	$V_{CC} = 5.0V$, Fi = 0 kHz		50		50	µA
	$V_{CC} = 3.0V$, Fi = 0 kHz		20			µA
Input Voltage Levels						
$\overline{\text{RESET}}$, CKI						
Logic High		$0.9\ V_{CC}$		$0.9\ V_{CC}$		V
Logic Low			$0.1\ V_{CC}$		$0.1\ V_{CC}$	V
All Other Inputs						
Logic High		$0.7\ V_{CC}$		$0.7\ V_{CC}$		V
Logic Low			$0.2\ V_{CC}$		$0.2\ V_{CC}$	V
$\overline{\text{RESET}}$, SI Input Leakage		−2	+2	−2	+2	µA
Input Capacitance			7		7	pF
Output Voltage Levels						
(SO, SK, L Port)						
Logic High	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.2$		$V_{CC} - 0.2$		V
Logic Low	$I_{OL} = 10\ \mu A$		0.2		0.2	V
Output Current Levels						
Sink (Note 3)	V_{CC} = Min, $V_{OUT} = V_{CC}$	0.2		1.2		mA
Source (SO, SK, L Port)	V_{CC} = Min, V_{OUT} = 0V	−0.1		−0.5		mA
Source (G Port)	V_{CC} = Min, V_{OUT} = 0V	−8	−200	−30	−440	µA
Allowable Sink/Source Current Per Pin (Note 3)			5		5	mA
TRI-STATE Leakage Current[3]		−4	+4	−4	+4	µA

COP313C/COP313CH

AC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	COP313C		COP313CH		Units
		Min	Max	Min	Max	
Instruction Cycle Time		16	DC	4	DC	µs
Operating CKI Frequency	÷ 8 Mode	DC	500	DC	2000	kHz
Instruction Cycle Time RC Oscillator ÷ 4	$R = 30k \pm 5\%$, $V_{CC} = 5V$ $C = 82\ pF \pm 5\%$			8	16	µs
Instruction Cycle Time RC Oscillator ÷ 4 (Note 6)	$R = 56k \pm 5\%$, $V_{CC} = 5V$ $C = 100\ pF \pm 5\%$	16	32	16	32	µs
Duty Cycle (Note 5)	Fi = Max Freq Ext Clk	40	60	40	60	%
Rise Time (Note 5)	Fi = Max Freq Ext Clk		60		60	ns
Fall Time (Note 5)	Fi = Max Freq Ext Clk		40		40	ns
Inputs (See *Figure 3*) t_{SETUP}	 G Inputs SI Input L Inputs	 tc/4 + 2.8 1.2 6.8		 tc/4 + 0.7 0.3 1.7		 µs µs µs
t_{HOLD}		1.0		0.25		µs
Output Propagation Delay t_{PD1}, t_{PD0}	$V_{OUT} = 1.5V$, $C_L = 100\ pF$ $R_L = 5k$		4.0		1.0	µs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 13.

Note 2: The Halt mode will stop CKI from oscillating.

Note 3: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} when part is running in order to prevent entering test mode.

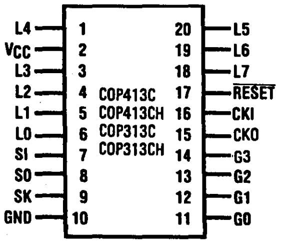
Note 4: Voltage change must be less than 0.5V in a 1 ms period.

Note 5: This parameter is only sampled and not 100% tested.

Note 6: Variation due to the device included.

Connection Diagram

DIP



TL/DD/8537-2

Top View

Pin Descriptions

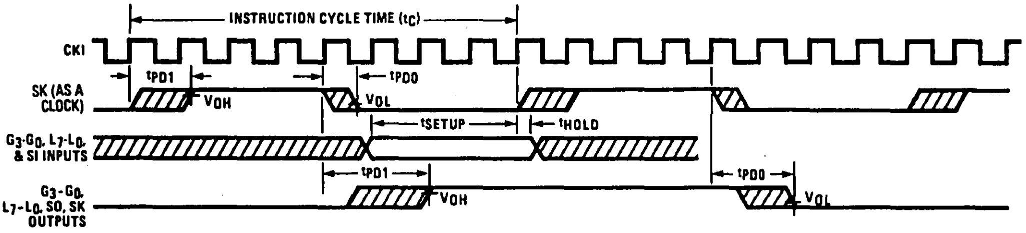
Pin	Description
L_7–L_0	8-bit bidirectional I/O port with TRI-STATE
G_3–G_0	4-bit bidirectional I/O port
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	Crystal oscillator output, or NC
$\overline{\text{RESET}}$	System reset input
V_{CC}	System power supply
GND	System Ground

FIGURE 2

Order Number COP313C-XXX/D, COP313CH-XXX/D, COP413C-XXX/D or COP413CH-XXX/D
See NS Hermetic Package Number D20A

Order Number COP313C-XXX/N, COP313CH-XXX/N, COP413C-XXX/N or COP413CH-XXX/N
See NS Molded Package Number N20A

Timing Waveform


INSTRUCTION CYCLE TIME (t_C)
CKI
SK (AS A CLOCK)
t_{PD1}
t_{PD0}
V_{OH}
V_{OL}
t_{SETUP}
t_{HOLD}
G_3-G_0, L_7-L_0, & SI INPUTS
t_{PD1}
t_{PD0}
V_{OH}
V_{OL}
G_3-G_0, L_7-L_0, SO, SK OUTPUTS

TL/DD/8537–3

FIGURE 3. Input/Output Timing Diagrams (Divide-by-8 Mode)

Development Support

The MOLE (Microcontroller On Line Emulator) is a low cost development system and real time emulator for COPS' products. They also include TMP, 8050 and the new 16 bit HPC microcontroller family. The MOLE provides effective support for the development of both software and hardware in the user's application.

The purpose of the MOLE is to provide a tool to write and assemble code, emulate code for the target microcontroller and assist in debugging of the system.

The MOLE can be connected to various hosts, IBM PC, STARPLEX™, Kaypro, Apple and Intel systems, via RS-232 port. This link facilitates the up loading/down loading of code, supports host assembly and mass storage.

The MOLE consists of three parts; brain, personality and optional host software.

The brain board is the computing engine of the system. It is a self-contained computer with its own firmware which provides for all system operation, emulation control, communication, from programming and diagnostic operation. It has three serial ports which can be connected to a terminal, host system, printer, modem or to other MOLE's in a multi-MOLE environment.

The personality board contains the necessary hardware and firmware needed to emulate the target microcontroller. The emulation cable which replaces the target controller attaches to this board. The software contains a cross assembler and a communications program for up loading and down loading code from the MOLE.

MOLE Ordering Information

P/N	Description
MOLE-BRAIN	MOLE Computer Board
MOLE-COPS-PB1	COPS Personality Board
MOLE-XXX-YYY	Optional Software

Where XXX = COPS
YYY = Host System, IBM, Apple,
KAY (Kaypro), CP/M

Functional Description

To ease reading of this description, only COP413C is referenced; however, all such references apply equally to COP413CH, COP313C, and COP313CH.

A block diagram of the COP413C is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1"; when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program memory consists of a 512-byte ROM. As can be seen by an examination of the COP413C instruction set, these words may be program instructions, program data, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words (bytes) each.

ROM ADDRESSING

ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by two 9-bit subroutine save registers, SA and SB.

ROM instruction words are fetched, decoded, and executed by the instruction decode, control and skip logic circuitry.

DATA MEMORY

Data Memory consists of a 128-bit RAM, organized as four data registers of 8 × 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper two bits (Br) selects one of four data registers and lower three bits of the 4-bit Bd select one of eight 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), they may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence, each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4.* The skip condition for XIS and XDS instructions will be true if Bd changes between 0 to 15, but *not* between 7 and 8 (see Table III).

INTERNAL LOGIC

The internal logic of the COP413C is designed to ensure fully static operation of the device.

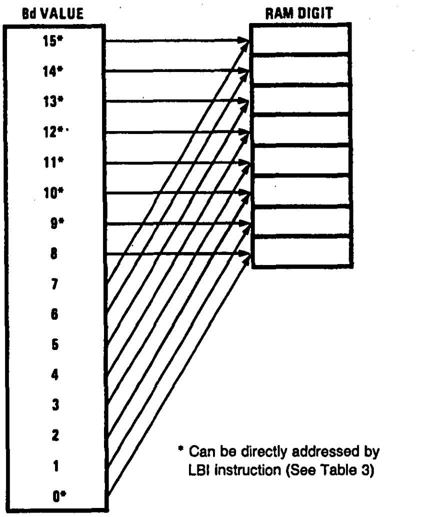
The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load four bits of the 8-bit Q latch data and to perform data exchanges with the SIO register.

The 4-bit adder performs the arithmetic and logic functions of the COP413C, storing its results in A. It also outputs the carry information to a 1-bit carry register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description below.)

The G register contents are outputs to four general purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from RAM and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The eight L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and RAM.



* Can be directly addressed by LBI instruction (See Table 3)

TL/DD/8537-4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter, depending upon the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. With SIO functioning as a serial-in/serial-out shift register and SK as a sync clock, the COP413C is MICROWIRE compatible.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK is a sync clock, inhibited when SKL is a logic "0".

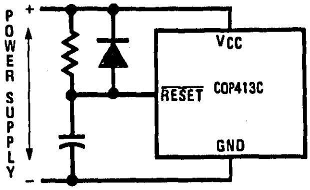
The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN3–EN0).

1. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or as a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN3. With EN0 reset, SIO is a serial shift register, shifting left each instruction cycle time. The data present at SI is shifted into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each instruction cycle time. (See 4, below.) The SK output becomes a logic-controlled clock.
2. EN 1 is not used, it has no effect on the COP413C.
3. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected), SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected, disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0".

INITIALIZATION

The external RC network shown in *Figure 5* must be connected to the $\overline{\text{RESET}}$ pin. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. Data memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



TL/DD/8537-5

RC > 5 × Power Supply Rise Time
and RC > 100 × CKI Period

FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN0 and EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = clock If SKL = 0, SK = 0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL = 1, SK = clock If SKL = 0, SK = 0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

HALT MODE

The COP413C is a *fully static* circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may be halted by the HALT instruction. Once in the HALT mode, the internal circuitry does not receive any clock signal, and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The HALT mode is the minimum power dissipation state.

The HALT mode may be entered into by program control (HALT instruction) which forces CKO to a logic "1" state. The circuit can be awakened only by the $\overline{\text{RESET}}$ function.

POWER DISSIPATION

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, to minimize power consumption, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to ensure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal- or resonator-generated clock will draw more than a square-wave input. An RC oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate the COP413C current drain.

$$Ic = Iq + (V \times 20 \times Fi) + (V \times 1280 \times FI/Dv)$$

where Ic = chip current drain in microamps

Iq = quiescent leakage current (from curve)

FI = CKI frequency in megahertz

V = chip V_{CC} in volts

Dv = divide by option selected

For example, at 5V V_{CC} and 400 kHz (divide by 8),

$$Ic = 30 + (5 \times 20 \times 0.4) + (5 \times 1280 \times 0.4/8)$$

$$Ic = 30 + 40 + 320 = 390\,\mu A$$

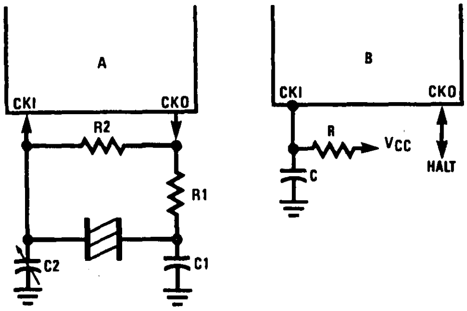
OSCILLATOR OPTIONS

There are two options available that define the use of CKI and CKO.

a. Cyrstal-Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 8.

b. RC-Controlled Oscillator. CKI is configured as a single pin RC-controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is NC.

The RC oscillator is not recommended in systems that require accurate timing or low current. The RC oscillator draws more current than an external oscillator (typically an additional 100 µA at 5V). However, when the part halts, it stops with CKI high and the halt current is at the minimum.



TL/DD/8537-6

FIGURE 6. COP413C Oscillator

Crystal or Resonator					RC-Controlled Oscillator			
Crystal	Component Value						Cycle	
Value	R1	R2	C1 pF	C2 pF	R	C	Time	V_{CC}
32 kHz	220k	20M	30	5–36	15k	82 pF	4–9 µs	≥ 4.5V COP413CH Only
455 kHz	5k	10M	80	40	30k	82 pF	8–16 µs	≥ 4.5V COP413CH Only
2.000 MHz	2k	1M	30	6–36	47k	100 pF	16–32 µs	3.0 to 4.5V COP413C Only
					56k	100 pF	16–32 µs	≥ 4.5V
					Note: 15k ≤ R ≤ 150k, 50 pF ≤ C ≤ 150 pF			

Functional Description (Continued)

I/O CONFIGURATIONS

COP413C outputs have the following configurations, illustrated in *Figure 7*:

a. Standard SO, SK Output. A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

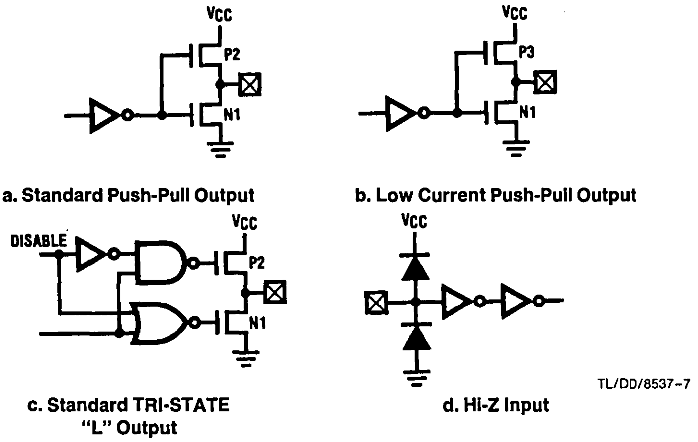
b. Low Current G Output. This is the same configuration as (a) above except that the sourcing current is much less.

c. Standard TRI-STATE L Output. L output is a CMOS output buffer similar to (a) which may be disabled by program control.

The SI and $\overline{\text{RESET}}$ inputs are Hi-Z inputs (*Figure 7d*).

When using the G I/O port as an input, set the output register to a logic "1" level. The P-channel device will act as a pull-up load. When using the L I/O port as an input, disable the L drivers with the LEI instruction. The drivers are then in TRI-STATE mode and can be driven externally.

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations.



a. Standard Push-Pull Output

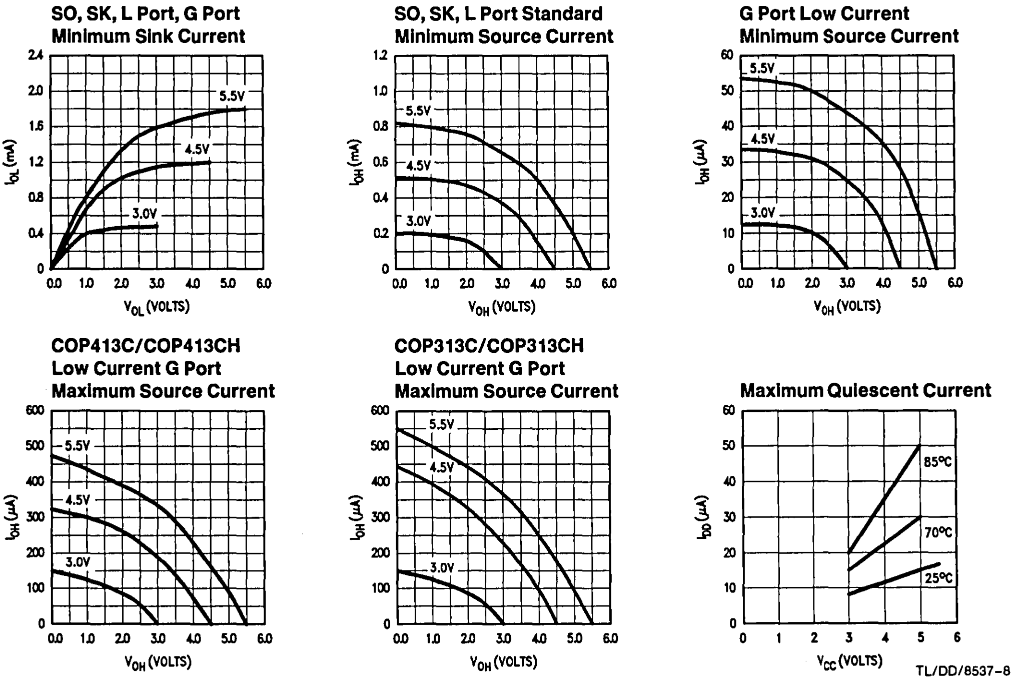
b. Low Current Push-Pull Output

c. Standard TRI-STATE "L" Output

d. Hi-Z Input

TL/DD/8537-7

FIGURE 7. I/O Configurations



TL/DD/8537-8

FIGURE 8

COP413C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP413C instruction set.

TABLE II. COP413C Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP413C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP413C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM (PC$_8$, A,M) → PC$_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– –	|0110|000|a_8| |$a_{7:0}$|	a → PC	None	Jump
JP	a	– –	|1|$a_{6:0}$| (pages 2, 3 only) or |11|$a_{5:0}$| (all other pages)	a → PC$_{6:0}$ a → PC$_{5:0}$	None	Jump within Page (Note 1)
JSRP	a	–	|10|$a_{5:0}$|	PC + 1 → SA → SB 010 → PC$_{8:6}$ a → PC$_{5:0}$	None	Jump to Subroutine Page (Note 2)
JSR	a	6– –	|0110|100|a_8| |$a_{7:0}$|	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|10011|	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	|0011|0011| |0011|1000|		None	Halt processor
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	|0011|0011| |0011|1100|	A → Q$_{7:4}$ RAM(B) → Q$_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	|0011|0011| |0010|1100|	Q$_{7:4}$ → RAM(B) Q$_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	|00|r|0101|	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LQID		BF	|1011|1111|	ROM(PC$_8$,A,M) → Q SA → SB	None	Load Q Indirect
RMB	0 1 2 3	4C 45 42 43	|0100|1100| |0100|0101| |0100|0010| |0100|0011|	0 → RAM(B)$_0$ 0 → RAM(B)$_1$ 0 → RAM(B)$_2$ 0 → RAM(B)$_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	|0100|1101| |0100|0111| |0100|0110| |0100|1011|	1 → RAM(B)$_0$ 1 → RAM(B)$_1$ 1 → RAM(B)$_2$ 1 → RAM(B)$_3$	None	Set RAM Bit
STII	y	7–	|0111|y|	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	|00|r|0110|	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	|0010|0011| |1011|1111|	RAM(3,15) ←→ A	None	Exchange A with RAM (3,15)

Instruction Set (Continued)

TABLE III. COP413C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	−7	|00|r|0111|	RAM(B) ←→ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd Exclusive-OR Br with r
XIS	r	−4	|00|r|0100|	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	-	|00|r|(d - 1)| (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d
LEI	y	33 6−	|0011|0011| |0010| y |	y → EN	None	Load EN Immediate
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	$G_0 = 0$	
	1	11	|0001|0001|		$G_1 = 0$	
	2	03	|0000|0011|		$G_2 = 0$	
	3	13	|0010|0011|		$G_3 = 0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1 = 0$	
	2	03	|0000|0011|		$RAM(B)_2 = 0$	
	3	13	|0001|0011|		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	|0011|0011| |0010|1010|	G → A	None	Input G Ports to A
INL		33 2E	|0011|0011| |0010|1110|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OMG		33 3A	|0011|0011| |0011|1010|	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	|0100|1111|	A ←→ SIO, C → SKL	None	Exchange A with SIO

Note 1: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 2: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP413C programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every four instruction cycle times to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower eight bits of the ROM address register PC with the contents of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note: JID uses two instruction cycles if executed, one if skipped.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table look-up or code conversion such as BCD to 7-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant eight bits of the PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of the PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost.

Note: LQID uses two instruction cycles if executed, one if skipped.

INSTRUCTION SET NOTES

a. The first word of a COP413C program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed (except JID and LQID).

c. The ROM is organized into eight pages of 64 words each. The program counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: A JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word in page 3 or 7 will access data in the next group of four pages.

COPS Programming Manual

For detailed information on writing. COPS programs, the COPS Programming Manual 424410284-001 provides an in-depth discussion of the COPS architecture, instruction set and general techniques of COPS programming. This manual is written with the programmer in mind.

OPTION LIST—OSCILLATOR SELECTION

The oscillator option selected must be sent in with the EPROM of ROM Code for masking into the COP413C. Select the appropriate option, make a photocopy of the table and send it with the EPROM.

COP413C/COP313C

Option 1: Oscillator selection

= 0 Ceramic Resonator input frequency divided by 8. CKO is oscillator output.

= 1 Single pin RC controlled oscillator divided by 4. CKO is no connection.

Note: The following option information is to be sent to National along with the EPROM.

Option 1: Value = ____ is Oscillator Selected.

National Semiconductor

PRELIMINARY

COP414L/COP314L Single-Chip N-Channel Microcontrollers

General Description

The COP414L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. This Controller Oriented Processor is a complete microcomputer containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP414L is an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

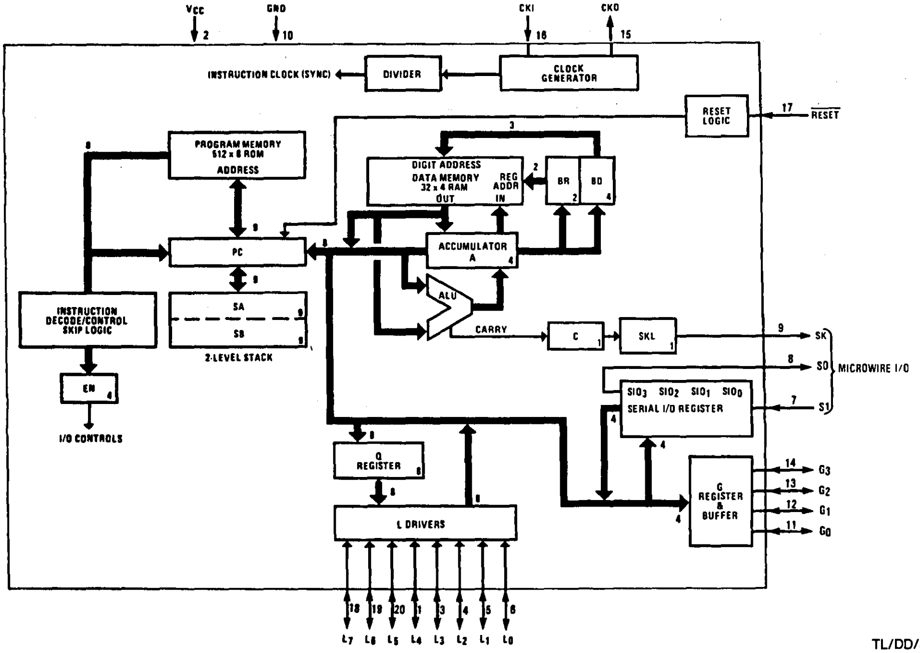
The COP314L is an exact functional equivalent but extended temperature version of COP414L.

The COP414L can be emulated by the COP404C. The COP401L should be used for exact emulation.

Features

- Late waferfab programming of ROM and I/O for fast delivery of units
- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- 15 I/O lines
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (6 mA max)
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device — COP314L (−40°C to +85°C)
- Wider supply range (4.5V–9.5V) optionally available

Block Diagram



TL/DD/8814–1

FIGURE 1. COP414L

COP414L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	
COP414L	0.65W at 25°C 0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Optional Operating Voltage (V_{CC})		4.5	9.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		6	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

COP414L

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 9.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Ouputs (I_{OL})	$V_{CC} = 9.5V, V_{OL} = 0.4V$	1.8		mA
	$V_{CC} = 6.3V, V_{OL} = 0.4V$	1.2		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.9		mA
L_0–L_7 Outputs, G_0–G_3 and	$V_{CC} = 9.5V, V_{OL} = 0.4V$	0.4		mA
LSTTL D_0–D_3 Outputs (I_{OL})	$V_{CC} = 6.3V, V_{OL} = 0.4V$	0.4		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
CKI (Single-pin RC Oscillator)	$V_{CC} = 4.5, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−140	−800	μA
All Outputs (I_{OH})	$V_{CC} = 6.3V, V_{OH} = 2.0V$	−75	−480	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−30	−250	μA
Push-Pull Configuration	$V_{CC} = 9.5V, V_{OH} = 4.75V$	−1.4		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 6.3V, V_{OH} = 2.4V$	−1.4		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
Input Load Source Current	$V_{CC} = 5.0V, V_{IL} = 0V$	−10	−140	μA
Open Drain Output Leakage		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pin			1.5	mA

COP314L

Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Power Dissipation COP314L	0.65W at 25°C 0.20W at 85°C
Total Source Current	120 mA
Total Sink Current	100 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

COP314L: $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 7.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Optional Operating Voltage (V_{CC})		4.5	7.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
Ceramic Resonator Input (÷8)				
Crystal Input				
Logic High (V_{IH})	V_{CC} = Max	3.0		
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

COP314L

DC Electrical Characteristics (Continued)

COP314L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs(I_{OL})	V_{CC} = 7.5V, V_{OL} = 0.4V	1.4		mA
	V_{CC} = 5.5V, V_{OL} = 0.4V	1.0		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.8		mA
L_0–L_7 Outputs, G_0–G_3 and LSTTL, D_0–D_3 Outputs (I_{OL})	V_{CC} = 7.5V, V_{OL} = 0.4V	0.4		mA
	V_{CC} = 5.5V, V_{OL} = 0.4V	0.4		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
CKI (Single-pin RC Oscillator)	V_{CC} = 4.5V, V_{IH} = 3.5V	1.5		mA
CKO	V_{CC} = 4.5V, V_{OL} = 0.4V	0.2		mA
Output Source Current				
Standard Configuration, All Outputs (I_{OH})	V_{CC} = 7.5V, V_{OH} = 2.0V	−100	−900	μA
	V_{CC} = 5.5V, V_{OH} = 2.0V	−55	−600	μA
	V_{CC} = 4.5V, V_{OH} = 2.0V	−28	−350	μA
Push-Pull Configuration SO and SK Outputs (I_{OH})	V_{CC} = 7.5V, V_{OH} = 3.75V	−0.85		mA
	V_{CC} = 5.5V, V_{OH} = 2.0V	−1.1		mA
	V_{CC} = 4.5V, V_{OH} = 1.0V	−1.2		mA
Input Load Source Current	V_{CC} = 5.0V, V_{IL} = 0V	−10	−200	μA
Open Drain Output Leakage		−5	+5	μA
Total Sink Current Allowed				
All Outputs Combined			100	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
Any Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
Any Other Pins			1.5	mA

AC Electrical Characteristics

COP414L: 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 9.5V unless otherwise noted

COP314L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted

COP214L: −40°C ≤ T_A ≤ +110°C, 4.5V ≤ V_{CC} ≤ 7.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time — t_C		16	40	μs
CKI				
Input Frequency — f_I	÷8 Mode ÷4 Mode	0.2 0.1	0.5 0.25	MHz MHz
Duty Cycle		30	60	%
Rise Time	f_I = 0.5 MHz		500	ns
Fall Time			200	ns

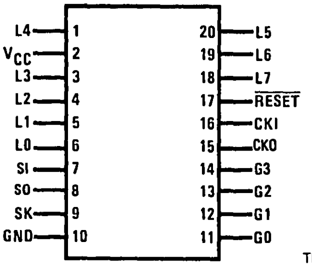
CKI Using RC (÷4)	R = 56 kΩ ±5%
	C = 100 pF ±10%			

Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
Inputs				
G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

Note 1: Variation due to the device included.

Connection Diagram

Dual-In-Line Package



TL/DD/8814-2

Top View

Order Number COP214L-XXX/D, COP314L-XXX/D or COP414L-XXX/D See NS Hermetic Package D20A

Order Number COP214L-XXX/N, COP314L-XXX/N or COP414L-XXX/N See NS Molded Package N20A

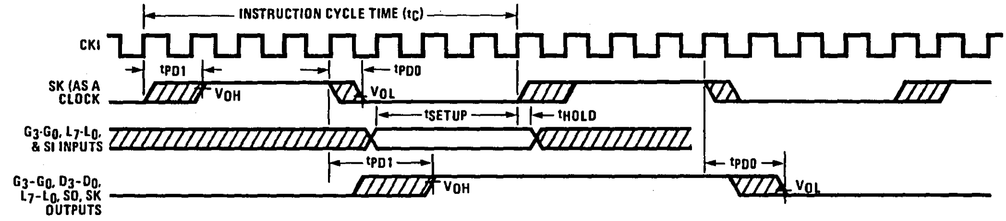
Order Number COP214L-XXX/WM, COP314L-XXX/WM or COP414L-XXX/WM See NS Surface Mount Package M20B

FIGURE 2

Pin Descriptions

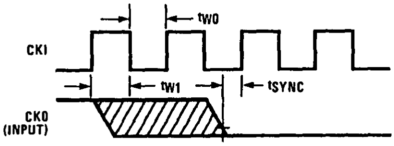
Pin	Description	Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE	CKI	System oscillator input
G_3–G_0	4 bidirectional I/O ports	CKO	System oscillator output
SI	Serial input (or counter input)	$\overline{\text{RESET}}$	System reset input
SO	Serial output (or general purpose output)	V_{CC}	Power supply
SK	Logic-controlled clock (or general purpose output)	GND	Ground

Timing Diagrams


INSTRUCTION CYCLE TIME (tC)
CKI
SK (AS A CLOCK
tPD1
tPD0
VOH
VOL
tSETUP
tHOLD
G3-G0, L7-L0, & SI INPUTS
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
tPD1
tPD0
VOH
VOL

TL/DD/8814-3

FIGURE 3. Input/Output Timing Diagrams (Ceramic Resonator Divide-by-8 Mode)



TL/DD/8814-4

FIGURE 3a. Synchronization Timing

Functional Description

A block diagram of the COP414L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

All functional references to the COP414L also apply to the COP314L, and COP214L.

PROGRAM MEMORY

Program Memory consists of a 512-byte ROM. As can be seen by an examination of the COP414L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

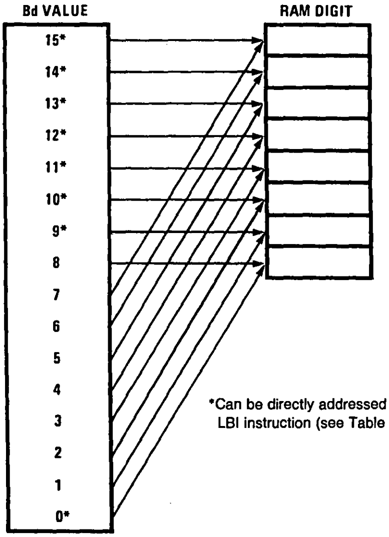
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table III).



*Can be directly addressed by LBI instruction (see Table III)

TL/DD/8814-5

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

Functional Description (Continued)

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP414L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

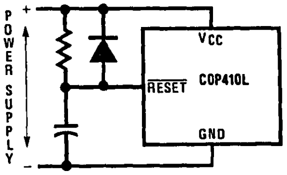
The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occuring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP414L operation.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below *(Figure 5)*. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



RC ≥ 5 × Power Supply Rise Time TL/DD/8814-6

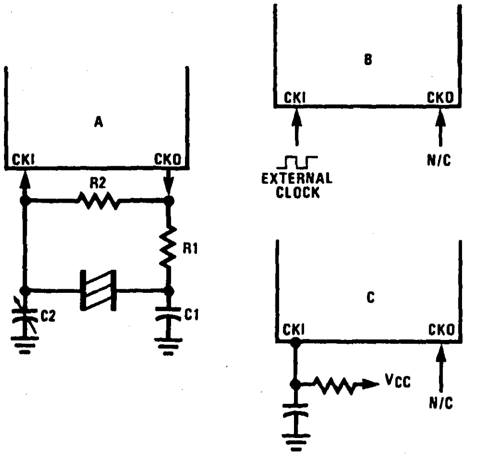
FIGURE 5. Power-Up Clear Circuit

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/8814-7

Ceramic Resonator Oscillator

Resonator Value	Components Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time in μs
51	100	19 ±15%
82	56	19 ±13%

Note: $200\,k\Omega \geq R \geq 25\,k\Omega$. $360\,pF \geq C \geq 50\,pF$. Does not include tolerances.

FIGURE 6. COP414L Oscillator

OSCILLATOR

There are four basic clock oscillator configurations available as shown by *Figure 6.*

a. **Resonator Controlled Oscillator.** CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.

b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 4 to give the instruction frequency time. CKO is no connection.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is no connection.

CKO PIN OPTIONS

In a resonator controlled oscillator system, CKO is used as an output to the resonator network. CKO is no connection for External or RC controlled oscillator.

I/O OPTIONS

COP414L inputs and outputs have the following optional configurations, illustrated in *Figure 7*:

a. **Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK and all D and G outputs.

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK and all D and G outputs.

c. **Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. **Standard L**—same as **a.**, but may be disabled. Available on L outputs only.

e. **Open Drain L**—same as **b.**, but may be disabled. Available on L outputs only.

f. An on-chip depletion load device to V_{CC}.

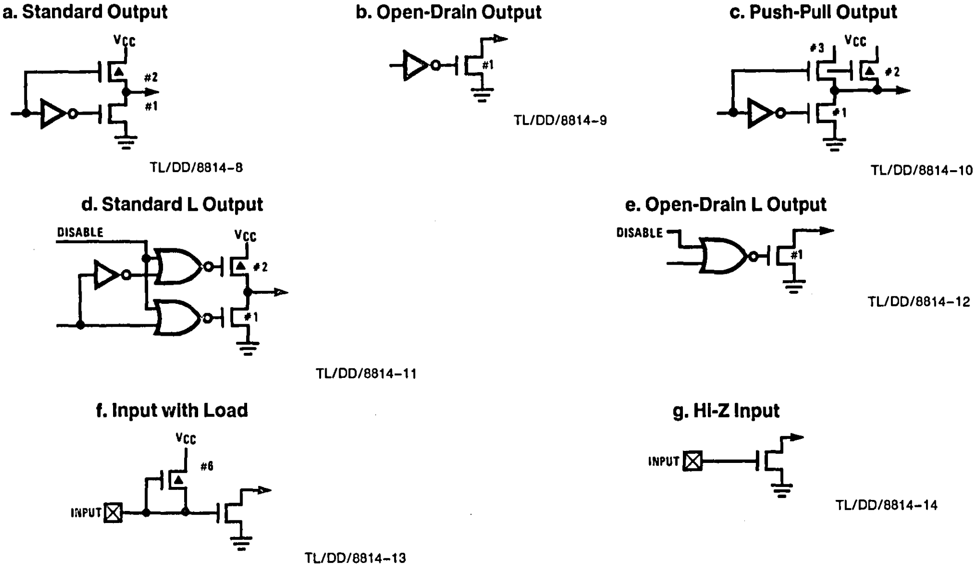
g. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 8* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP414L system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in **d.**, or **e.**

An important point to remember if using configuration **d.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 8*, device 2.) However, when the L port is used as input, the disabled depletion device CANNOT be relied on to source sufficient current to pull an input to a logic "1".

Functional Description (Continued)



a. Standard Output

b. Open-Drain Output

c. Push-Pull Output

d. Standard L Output

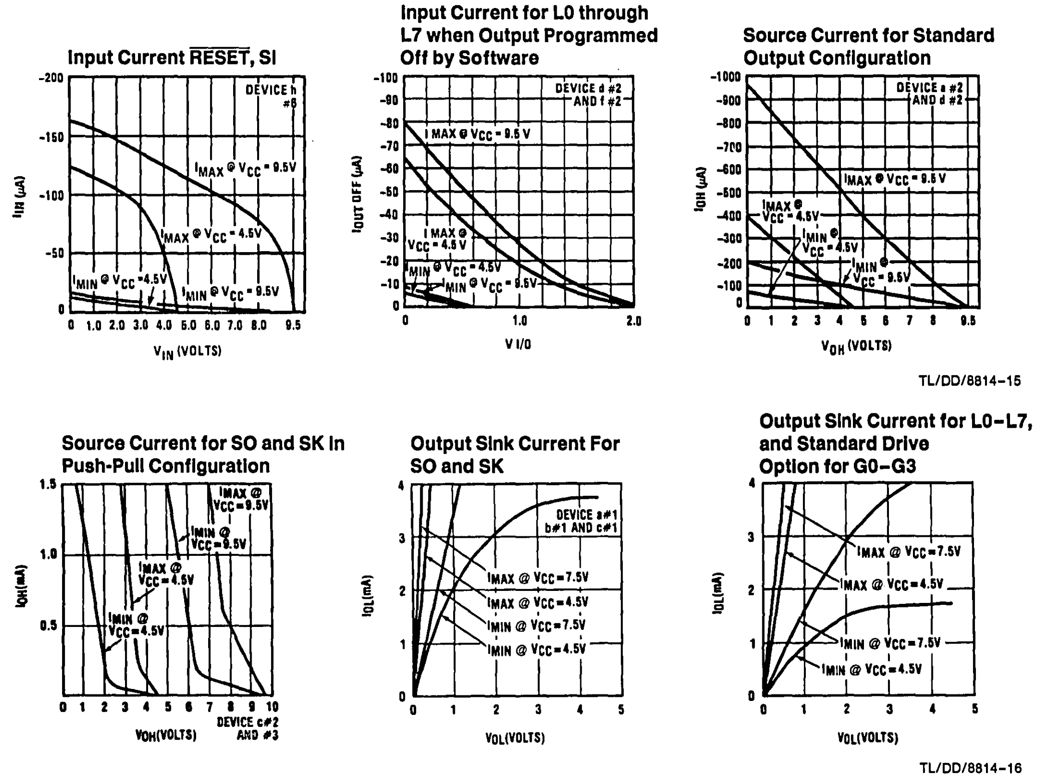
e. Open-Drain L Output

f. Input with Load

g. Hi-Z Input

FIGURE 7. Input and Output Configurations

Typical Performance Curves



Input Current $\overline{\text{RESET}}$, SI

Input Current for L0 through L7 when Output Programmed Off by Software

Source Current for Standard Output Configuration

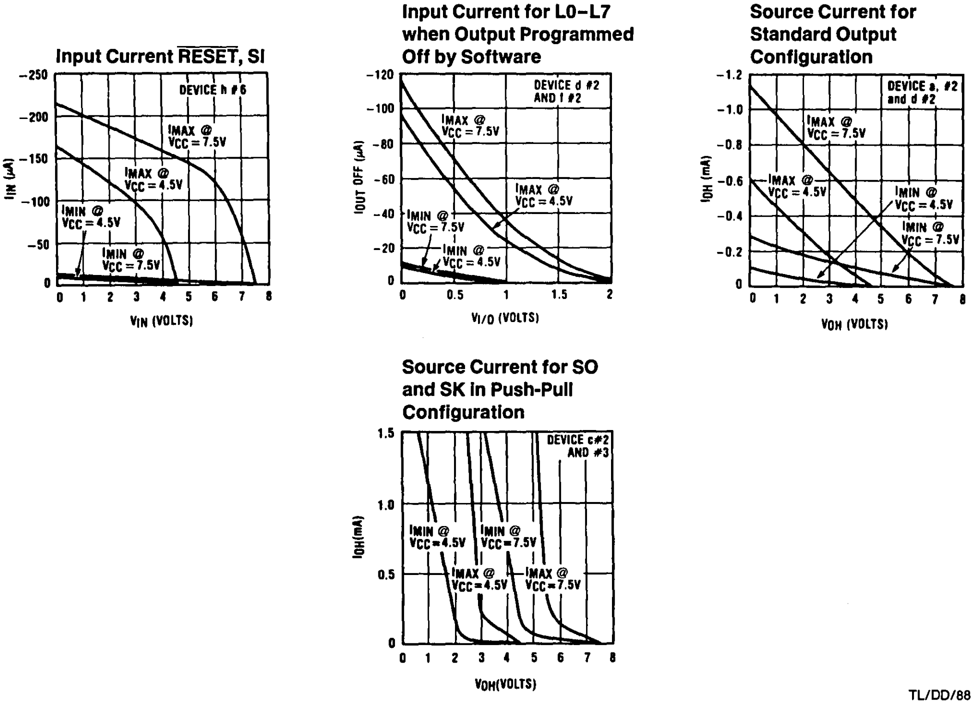
Source Current for SO and SK in Push-Pull Configuration

Output Sink Current For SO and SK

Output Sink Current for L0–L7, and Standard Drive Option for G0–G3

FIGURE 8a. COP414 I/O DC Current Characteristics

Typical Performance Curves (Continued)



TL/DD/8814–17

FIGURE 8b. COP314L Input/Output Characteristics

COP414L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP414L instruction set.

TABLE II. COP414L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP414L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
AISC	y	5-	0101 y	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

COP414L Instruction Set (Continued)

TABLE III. COP414L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM (PC_8,A,M) $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– – –	|0110|000|a_8| |$a_{7:0}$|	a → PC	None	Jump
JP	a	– – – –	|1| $a_{6:0}$ | (pages 2, 3 only) or |11| $a_{5:0}$ | (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 3)
JSRP	a	– –	|10| $a_{5:0}$ |	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– – –	|0110|100|a_8| |$a_{7:0}$|	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	|0011|0011| |0011|1100|	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
LD	r	-5	|00|r|0101|	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	|1011|1111|	ROM(PC_8, A, M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0 1 2 3	4C 45 42 43	|0100|1100| |0100|0101| |0100|0010| |0100|0011|	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	|0100|1101| |0100|0111| |0100|0110| |0100|1011|	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	|0111| y |	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	|00|r|0110|	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3, 15	23 BF	|0010|0011| |1011|1111|	RAM(3,15) ←→ A	None	Exchange A with RAM (3,15)
XDS	r	-7	|00|r|0111|	RAM(B) ←→ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	|00|r|0100|	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd Exclusive-OR Br with r

COP414L Instruction Set (Continued)

TABLE III. COP414L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	- -	|00|r|(d − 1)| (d = 0,9:15)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 5)
LEI	y	33 6−	|0011|0011| |0010| y |	y → EN	None	Load EN Immediate (Note 6)
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	$G_0 = 0$	
	1	11	|0001|0001|		$G_1 = 0$	
	2	03	|0000|0011|		$G_2 = 0$	
	3	13	|0001|0011|		$G_3 = 0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1 = 0$	
	2	03	|0000|0011|		$RAM(B)_2 = 0$	
	3	13	|0001|0011|		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	|0011|0011| |0010|1010|	G → A	None	Input G Ports to A
INL		33 2E	|0011|0011| |0010|1110|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	|0011|0011| |0011|1110|	Bd → D	None	Output Bd to D Outputs
OMG		33 3A	|0011|0011| |0011|1010|	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	|0100|1111|	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Option List

The COP414L mask-programmable options are assigned numbers which correspond with the COP414L pins.

The following is a list of COP414L options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1: L_4 Driver
= 0: Standard output
= 1: Open-drain output

Option 2: V_{CC} Pin
= 0: Standard V_{CC}
= 1: Optional higher voltage V_{CC}

Option 3: L_3 Driver
same as Option 1

Option 4: L_2 Driver
same as Option 1

Option 5: L_1 Driver
same as Option 1

Option 6: L_6 Driver
same as Option 1

Option 7: SI Input
= 0: load device to V_{CC}
= 1: Hi-Z Output

Option 8: SO Driver
= 0: Standard output
= 1: Open-drain output
= 2: Push-pull output

Option 9: SK Driver
same as Option 8

Option 10:
= 0: Ground Pin—no options available

Option 11: G_0 I/O Port
= 0: Standard output
= 1: Open-drain output

Option 12: G_1 I/O Port
same as Option 11

Option 13: G_2 I/O Port
same as Option 11

Option 14: G_3 I/O Port
same as Option 11

Option 15: CKO Output
= 0: Clock output to ceramic resonator/crystal
= 1: No connection

Option 16: CKI Input
= 0: Ocillator input divided by 8 (500 kHz max)
= 1: Single pin RC controlled oscillator divided by 4
= 2: External Schmitt trigger level clock divided by 4

Option 17: $\overline{\text{RESET}}$ Input
= 0: Load device to V_{CC}
= 1: Hi-Z Input

Option 18: L_7 Driver
same as Option 1

Option 19: L_6 Driver
same as Option 1

Option 20: L_6 Driver
same as Option 1

Option 21: L Input Levels
= 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
= 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 22: G Input Levels
same as Option 21

Option 23: SI Input Levels
same as Option 21

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP414L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing tests only.

COP414L Option List

Please fill out the Option List and send it with the EPROM.

Option Data

OPTION 1 VALUE = ______ IS: L_4 DRIVER
OPTION 2 VALUE = ______ IS: V_{CC} PIN
OPTION 3 VALUE = ______ IS: L_3 DRIVER
OPTION 4 VALUE = ______ IS: L_2 DRIVER
OPTION 5 VALUE = ______ IS: L_1 DRIVER
OPTION 6 VALUE = ______ IS: L_6 DRIVER
OPTION 7 VALUE = ______ IS: SI INPUT
OPTION 8 VALUE = ______ IS: SO DRIVER
OPTION 9 VALUE = ______ IS: SK DRIVER
OPTION 10 VALUE = __0__ IS: GROUND PIN
OPTION 11 VALUE = ______ IS: G_0 I/O PORT
OPTION 12 VALUE = ______ IS: G_1 I/O PORT
OPTION 13 VALUE = ______ IS: G_2 I/O PORT
OPTION 14 VALUE = ______ IS: G_3 I/O PORT
OPTION 15 VALUE = ______ IS: CKO OUTPUT
OPTION 16 VALUE = ______ IS: CKI INPUT
OPTION 17 VALUE = ______ IS: RESET INPUT
OPTION 18 VALUE = ______ IS: L_7 DRIVER
OPTION 19 VALUE = ______ IS: L_6 DRIVER
OPTION 20 VALUE = ______ IS: L_6 DRIVER
OPTION 21 VALUE = ______ IS: L INPUT LEVELS
OPTION 22 VALUE = ______ IS: G INPUT LEVELS
OPTION 23 VALUE = ______ IS: SI INPUT LEVELS

National Semiconductor

COP420/COP421/COP422 and COP320/COP321/COP322 Single-Chip N-Channel Microcontrollers

General Description

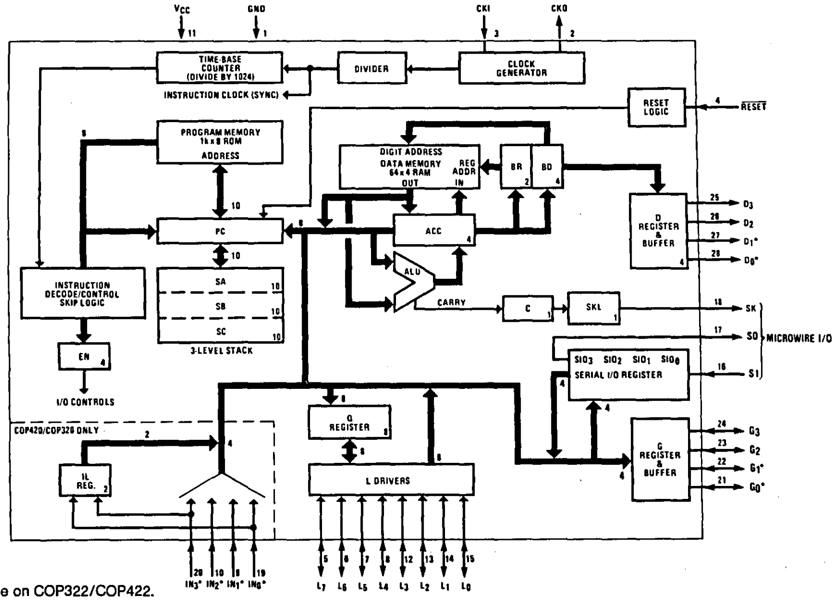
The COP420, COP421, COP422, COP320, COP321 and COP322 Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. They are complete microcomputers containing all system timing, internal logic, ROM, RAM and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP421 is identical to the COP420, except with 19 I/O lines instead of 23; the COP422 has 15 I/O lines. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized Controller Oriented Processor at a low end-product cost.

The COP320 is the extended temperature range version of the COP420 (likewise the COP321 and COP322 are the extended temperature range versions of the COP421/COP422). The COP320/321/322 are exact functional equivalents of the COP420/421/422.

Features

- Low cost
- Powerful instruction set
- 1k x 8 ROM, 64 x 4 RAM
- 23 I/O lines (COP420, COP320)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 4.0 μs instruction time
- Single supply operation
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O capacity
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- LED direct drive outputs
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device COP320/COP321/COP322 (−40°C to +85°C)

Block Diagram



*Not available on COP322/COP422.

TL/DD/6921-1

FIGURE 1

COP420/COP421/COP422 and COP320/COP321/COP322
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin	−0.3V to +7V
Operating Temperature Range	
COP420/COP421/COP422	0°C to 70°C
COP320/COP321/COP322	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Total Sink Current	75 mA
Total Source Current	95 mA

Package Power Dissipation	750 mW at 25°C
24 and 28 pin	400 mW at 70°C
	250 mW at 85°C
Package Power Dissipation	650 mW at 25°C
20 pin	300 mW at 70°C
	200 mW at 85°C
Lead Temperature (soldering, 10 sec.)	300°C

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP420/COP421/COP422
DC Electrical Characteristics $0°C \leq T_A \leq +70°C$, $4.5V \leq V_{CC} \leq 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	6.3	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	Outputs Open		38	mA
Supply Current	Outputs Open, V_{CC} = 5V, T_A = 25°C		30	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High	V_{CC} = Max.	3.0		
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.4	V
TTL Input	V_{CC} = 5V ±5%			
Logic High		2.0		V
Logic Low		−0.3	0.8	V
Schmitt Trigger Inputs				
RESET, CKI (÷4)				
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 2)	2.0	3.0	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Load Source Current	V_{CC} = 5V			
CKO		−4	−800	µA
All Others		−100	−800	µA
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	µA
Output Voltage Levels				
Standard Outputs				
TTL Operation	V_{CC} = 5V ±10%			
Logic High	I_{OH} = −100 µA	2.4		V
Logic Low	I_{OL} = 1.6 mA	−0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	I_{OH} = −10 µA	V_{CC} −1		V
Logic Low	I_{OL} = +10 µA		0.2	V

Note 1: TRI-STATE and LED configurations are excluded.

Note 2: SO output "0" level must be less than 0.8V for normal operation.

COP420/COP421/COP422

DC Electrical Characteristics $0^\circ C \leq T_A \leq +70^\circ C$, $4.5V \leq V_{CC} \leq 6.3V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
LED Direct Drive Output	$V_{CC} = 6V$			
Logic High	$V_{OH} = 2.0V$	2.5	14	mA
CKI Sink Current (R/C Option)	$V_{IN} = 3.5V$	2		mA
CKO (RAM Supply Current)	$V_R = 3.3V$		3	mA
TRI-STATE or Open Drain Leakage Current	$V_{CC} = 5V$	−2.5	+2.5	μA
Output Current Levels				
Output Sink Current (I_{OL})	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	+1.6		mA
Output Source Current (I_{OH})				
Standard Configuration				
All Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.0V$	−200	−900	μA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−100	−500	μA
Push-Pull Configuration				
SO, SK Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.0V$	−1.0		mA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−0.4		mA
TRI-STATE Configuration				
L_0–L_7 Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.2V$	−0.8		mA
	$V_{CC} = 4.5V$, $V_{OH} = 1.5V$	−0.9		mA
LED Configuration				
L_0–L_7 Outputs	$V_{CC} = 6.3V$, $V_{OH} = 3.0V$	−1.0		mA
	$V_{CC} = 4.5V$, $V_{OH} = 2.0V$	−0.5		mA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

COP320/COP321/COP322

DC Electrical Characteristics $-40°C \le T_A \le +85°C, 4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	5.5	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	$T_A = -40°C$, Outputs Open		40	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High		2.2		V
Logic Low		−0.3	0.3	V
TTL Input	$V_{CC} = 5V \pm 5\%$			
Logic High		2.2		V
Logic Low		−0.3	0.6	V
Schmitt Trigger Inputs				
RESET, CKI (÷4)				
Logic High		$0.7 V_{CC}$		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 2)	2.0	3.0	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	$V_{CC} = 5V \pm 5\%$	2.2		V
Logic Low		−0.3	0.6	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Load Source Current	$V_{CC} = 5V$			
CKO		−4	−800	μA
All Others		−100	−800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
Standard Outputs				
TTL Operation	$V_{CC} = 5V \pm 10\%$			
Logic High	$I_{OH} = -75\ \mu A$	2.4		V
Logic Low	$I_{OL} = 1.6\ mA$	−0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	$I_{OH} = -10\ \mu A$	$V_{CC} - 1$		V
Logic Low	$I_{OL} = +10\ \mu A$	−0.3	0.2	V
Output Current Levels				
LED Direct Drive Output	$V_{CC} = 5V$ (Note 4)			
Logic High	$V_{OH} = 2.0V$	1.0	12	mA
CKI Sink Current (R/C Option)	$V_{IN} = 3.5V$	2		mA
CKO (RAM Supply Current)	$V_R = 3.3V$		4	mA
TRI-STATE or Open Drain Leakage Current		−5	+5	μA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

Note 1: TRI-STATE and LED configurations are excluded.

Note 2: SO output "0" level must be less than 0.6V for normal operation.

AC Electrical Characteristics

COP420/COP421/COP422 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted

COP320/COP321/COP322 −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	÷16 mode	1.6	4.0	MHz
	÷8 mode	0.8	2.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time	Freq. = 4 MHz		60	ns
Fall Time	Freq. = 4 MHz		40	ns
CKI Using RC *(Figure 8c)*	÷4 mode			
Frequency	R = 15 kΩ ±5%, C = 100 pF	0.5	1.0	MHz
Instruction Cycle Time (Note 5)		4	8	μs
CKO as SYNC Input *(Figure 8d)*				
t_{SYNC}	*Figure 3a*	50		ns
Inputs:				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
Output Propagation Delay	Test Conditions:	300		ns
	R_L = 5 kΩ, C_L = 50 pF, V_{OUT} = 1.5V			
SO and SK				
t_{pd1}			1.0	μs
t_{pd0}			1.0	μs
CKO				
t_{pd1}			0.25	μs
t_{pd0}			0.25	μs
All Other Outputs				
t_{pd1}			1.4	μs
t_{pd0}			1.4	μs

Note 1: Duty cycle = $t_{W1}/(t_{W1} + t_{W0})$.

Note 2: See *Figure 9* for additional I/O characteristics.

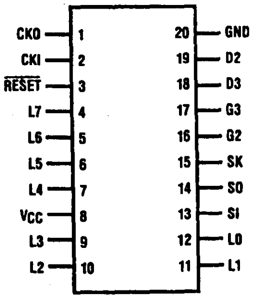
Note 3: Voltage change must be less than 0.5V in a 1 ms period.

Note 4: LED direct drive must not be used. Exercise great care not to exceed maximum device power dissipation limits when sourcing similar loads at high temperature.

Note 5: Variation due to the device included.

Connection Diagrams

COP422, COP322
DIP



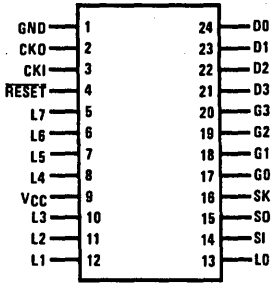
TL/DD/6921-4

Top View

Order Number COP322-XXX/N
or COP422-XXX/N
See NS Molded Package N20A

Order Number COP322-XXX/D
or COP422-XXX/D
See NS Hermetic Package D20A

COP421, COP321
DIP and SO Wide



TL/DD/6921-3

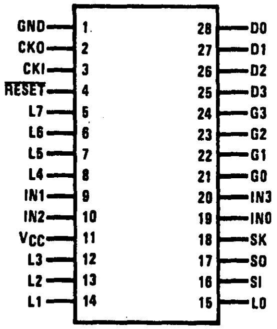
Top View

Order Number COP321-XXX/N
or COP421-XXX/N
See NS Molded Package N24A

Order Number COP321-XXX/D
or COP421-XXX/D
See NS Hermetic Package D24C

Order Number COP321-XXX/WM
or COP421-XXX/WM
See NS Surface Mount Package M24B

COP420, COP320
Dual-In-Line Package



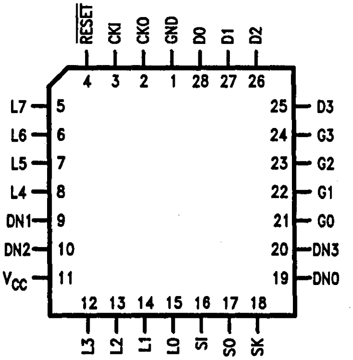
TL/DD/6921-2

Top View

Order Number COP320-XXX/N
or COP420-XXX/N
See NS Molded Package N28B

Order Number COP320-XXX/D
or COP320-XXX/D
See NS Hermetic Package D28C

28 PLCC



TL/DD/6921-31

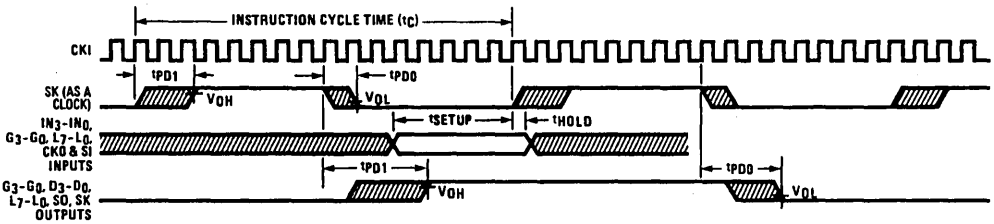
Order Number COP320-XXX/V
or COP420-XXX/V
See NS PLCC Package V28A

FIGURE 2

Pin Descriptions

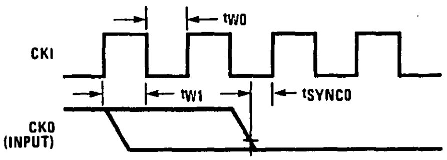
Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs (COP420/320 only)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output (or general purpose input or RAM power supply)
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams


INSTRUCTION CYCLE TIME (tC)
CKI
tPD1
tPD0
SK (AS A CLOCK)
VOH
VOL
tSETUP
tHOLD
IN3-IN0, G3-G0, L7-L0, CKO & SI INPUTS
tPD1
tPD0
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
VOH
VOL

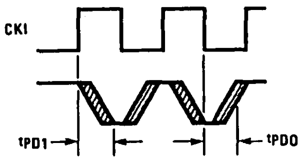
TL/DD/6921-5

FIGURE 3. Input/Output Timing Diagrams (Crystal Divide by 16 Mode)



TL/DD/6921-6

FIGURE 3A. Synchronization Timing



TL/DD/6921-7

FIGURE 3B. CKO Output Timing

Functional Description COP420/COP421/COP422, COP320/COP321/COP322

For ease of reading this description, only COP420 and/or COP421 are referenced; however, all such references apply equally to the COP422, COP322, COP320 and/or COP321, respectively.

A block diagram of the COP420 is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024 byte ROM. As can be seen by an examination of the COP420/421 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 10-bit binary count value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit **B register** whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit **A register** (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load the input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

A **4-bit adder** performs the arithmetic and logic functions of the COP420/421, storing its results in A. It also outputs a carry bit to the 1-bit **C register**, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four **general-purpose inputs, IN_3–IN_0,** are provided; IN_1, IN_2 and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select and Write Strobe inputs, respectively, for use in MICROBUS applications.

The **D register** provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The **G register** contents are outputs to 4 general-purpose bidirectional I/O ports. G_0 may be mask-programmed as an output for MICROBUS applications.

The **Q register** is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The **8 L drivers**, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The **SIO register** functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers. For example of additional parallel output capacity see **Application #2.**

The XAS instruction copies C into the **SKL latch.** In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The **EN register** is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0 selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0" occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting let each instruction cycle time. The data present at DI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With the EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN enables SO as the output of the SIO shift register outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides summary of the modes associated with EN_3 and EN_1.

OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 8.*

a. **Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16 (optional by 8).

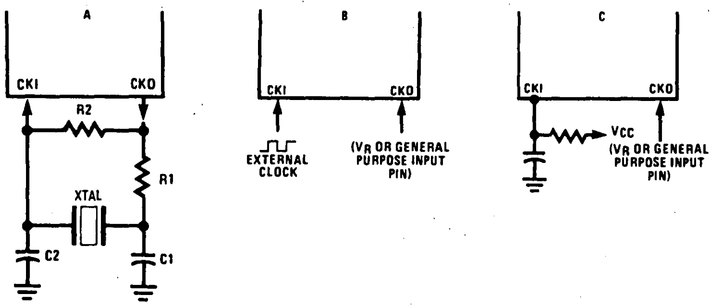
b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 16 (optional by 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R) of as a general purpose input.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available for non-timing functions.

Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)


A
CKI
CKO
R2
R1
XTAL
C2
C1
B
CKI
CKO
EXTERNAL CLOCK
(V_R OR GENERAL PURPOSE INPUT PIN)
C
CKI
CKO
VCC
(V_R OR GENERAL PURPOSE INPUT PIN)

TL/DD/6921-10

Crystal Oscillator **External Oscillator** **RC Controlled Oscillator**

Crystal Oscillator

Crystal Value	Component Values			
	R1(Ω)	R2(Ω)	C1(pF)	C2(pF)
4 MHz	4.7k	1M	22	22
3.58 MHz	3.3k	1M	22	27
2.09 MHz	8.2k	1M	47	33

RC Controlled Oscillator

R(kΩ)	C(pF)	Instruction Cycle Time (μs)
12	100	5 ±20%
6.8	220	5.3 ±23%
8.2	300	8 ±29%
22	100	8.6 ±16%

Note: 50 kΩ ≥ R ≥ 5 kΩ
360 pF ≥ C ≥50 pF

FIGURE 8. COP420/421/COP320/321 Oscillator

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP420/421 system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION (NOT AVAILABLE ON COP422)

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power up.
2. V_R must be within the operating range of the chip, and equal to V_{CC} ±1V during normal operation.
3. V_R must be ≥ 3.3V with V_{CC} off.

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:
   1. EN_1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time,* the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

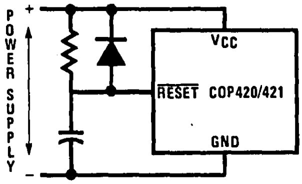
e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic, internal to the COP420/421, will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}.

Initialization will occur whenever a logic "0" is applied to the RESET input, provided it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



TL/DD/6921-13

FIGURE 7. Power-Up Clear Circuit

I/O OPTIONS

COP420/421 outputs have the following optional configurations, illustrated in *Figure 9a*:

a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. **Standard L**—same as **a.**, but may be disabled. Available on L outputs only.

e. **Open Drain L**—same as **b.**, but may be disabled. Available on L outputs only.

Functional Description COP420/COP421/COP422, COP320/COP321/COP322 (Continued)

f. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display.

g. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers.

COP420/COP421 inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

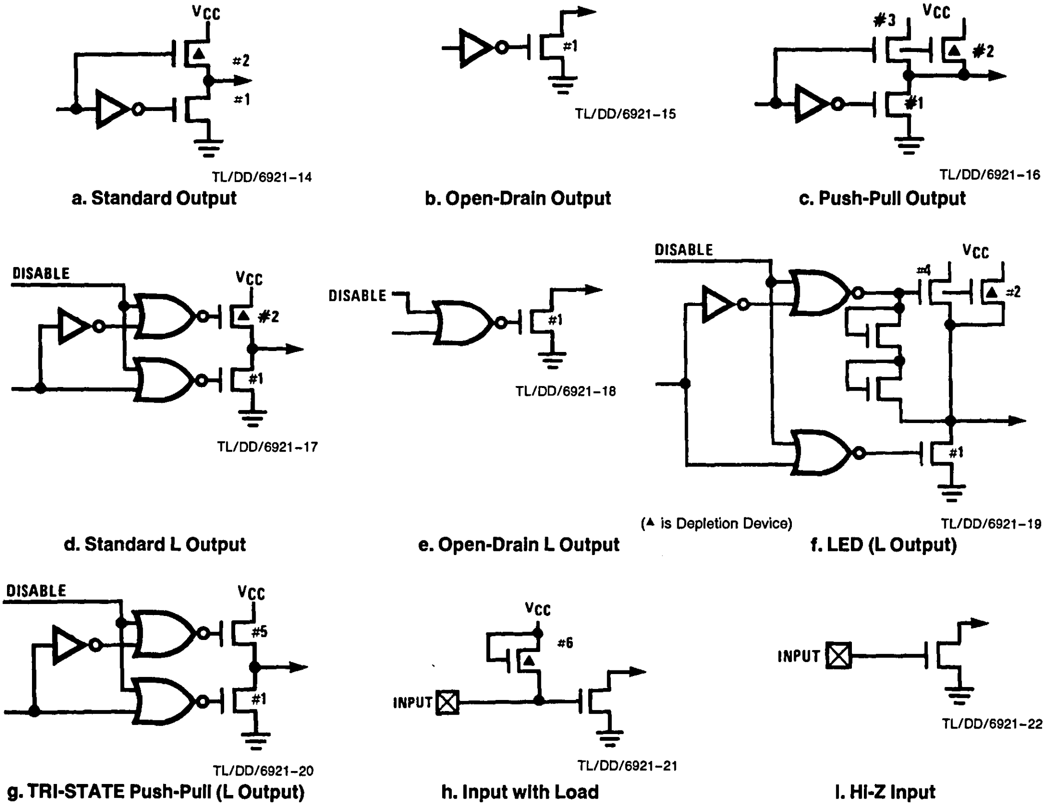
The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 9b* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP420/421 system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1." The L outputs can be configured as in **d.**, **e.**, **f.** or **g.**

An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 9b,* device 2); however, when the L lines are used as input, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to logic "1".

COP421

If the COP420 is bonded as a 24-pin device, it becomes the COP421, illustrated in *Figure 2,* COP420/421 Connection Diagrams. Note that the COP421 does not contain the four general purpose IN inputs (IN_3–IN_0). Use of this option precludes, of course, use of the IN options and interrupt feature. All other options are available for the COP421.



a. Standard Output

b. Open-Drain Output

c. Push-Pull Output

d. Standard L Output

e. Open-Drain L Output

f. LED (L Output)

g. TRI-STATE Push-Pull (L Output)

h. Input with Load

i. Hi-Z Input

FIGURE 9a. Input/Output Configurations

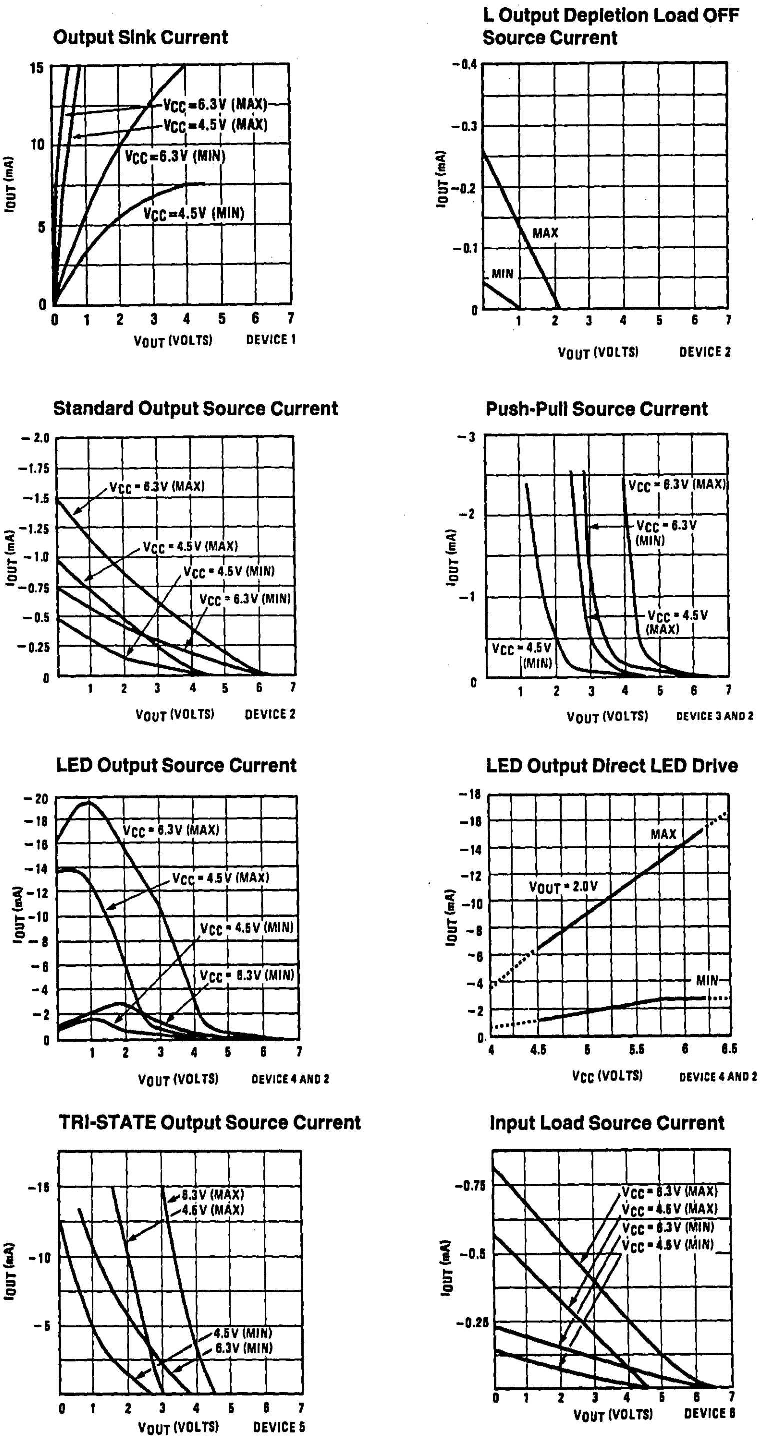
L-Bus Considerations

False states may be generated on L_0–L_7 during the execution of the CAMQ instruction. The L-ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. The following short program illustrates this situation.

```
START:
        CLRA          ;ENABLE THE Q
        LEI    4      ;REGISTER TO L LINES
        LBI    TEST
        STII   3
        AISC   12
LOOP:
        LBI    TEST   ;LOAD Q WITH X'C3
        CAMQ
        JP     LOOP
```

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L_0, L_1, L_6, L_7, and logic lows on L_2–L_5 via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L_0, L_1, L_6, L_7, and positive glitches on L_2–L_5. Glitch durations are under 2 microseconds, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics



Output Sink Current

L Output Depletion Load OFF Source Current

Standard Output Source Current

Push-Pull Source Current

LED Output Source Current

LED Output Direct LED Drive

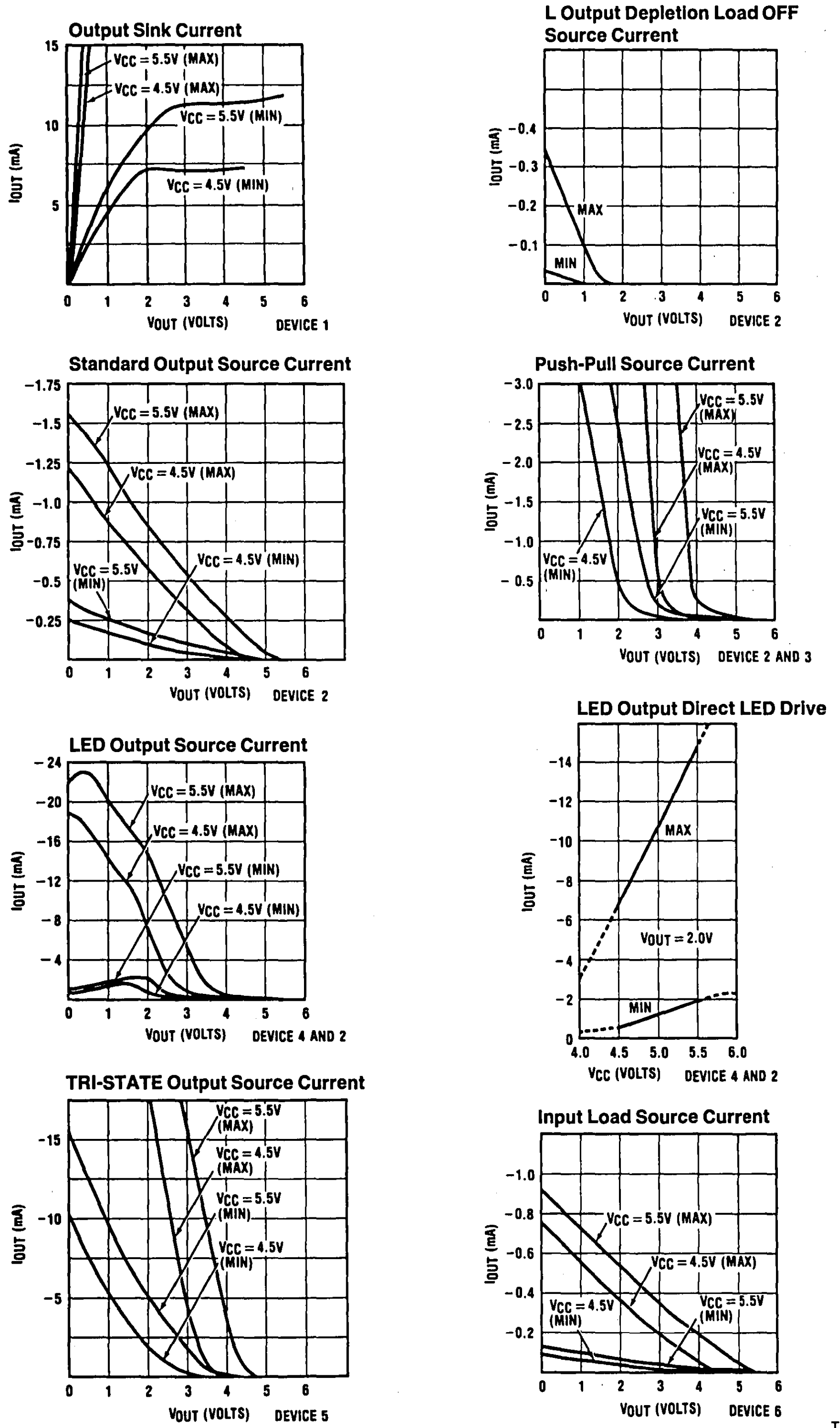
TRI-STATE Output Source Current

Input Load Source Current

TL/DD/6921-23

FIGURE 9b. COP420/COP421 Input/Output Characteristics

Typical Performance Characteristics (Continued)



TL/DD/6921-24

FIGURE 9c. COP320/COP321 Input/Output Characteristics

Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP420/COP421/COP422 instruction set.

TABLE I. COP420/421/422/320/321/322 Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10 Subroutine Save Register A
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	10-bit Operand Field, 0-1023 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE II. COP420/421/422/320/321/322 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	$A + C + RAM(B) \rightarrow A$ $Carry \rightarrow C$	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	$A + RAM(B) \rightarrow A$	None	Add RAM to A
ADT		4A	|0100|1010|	$A + 10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5−	|0101| y |	$A + y \rightarrow A$	Carry	Add immediate, Skip on Carry ($y \neq 0$)
CASC		10	|0001|0000|	$\overline{A} + RAM(B) + C \rightarrow A$ $Carry \rightarrow C$	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	$0 \rightarrow A$	None	Clear A
COMP		40	|0100|0000|	$\overline{A} \rightarrow A$	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	$A \oplus RAM(B) \rightarrow A$	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– – –	0110 00 a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	– – – –	1 $a_{6:0}$ (pages 2,3 only) or 11 $a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 4)
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → SA → SB → SC 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	0110 10 $a_{9:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LDD	r,d	23 – –	0010 0011 00 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{9:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 1101 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 – –	0010 0011 10 r d	RAM(r,d) ←→ A	None	Exchange A with RAM pointed to directly by r,d

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	−7	00 | r | 0111	RAM(B) ←→ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	−4	00 | r | 0100	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	- -	00 | r | (d - 1)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
			(d = 0,9:15)			
			or			
		33	0011 | 0011			
		- -	10 | r | d			
			(any d)			
LEI	y	33	0011 | 0011	y → EN	None	Load EN Immediate (Note 7)
		6−	0010 | y			
XABR		12	0001 | 0010	A ←→ Br (0,0 → A_3,A_2)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C = "1"	Skip if C is True
SKE		21	0010 | 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 | 0011		$G_{3:0}$ = 0	Skip if G is Zero (all 4 bits)
		21	0010 | 0001			
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	G_0 = 0	
	1	11	0001 | 0001		G_1 = 0	
	2	03	0000 | 0011		G_2 = 0	
	3	13	0010 | 0011		G_3 = 0	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0$ = 0	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1$ = 0	
	2	03	0000 | 0011		$RAM(B)_2$ = 0	
	3	13	0001 | 0011		$RAM(B)_3$ = 0	
SKT		41	0100 | 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE II. COP420/421/422/320/321/322 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN → A	None	Input IN Inputs to A (Note 2)
INIL		33 29	0011 0011 0010 1001	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5–	0011 0011 0101 y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit register.

Note 2: The ININ instruction is not available on the COP421/COP321 and COP422/COP322 since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP420/421 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, $PC_{9:8}$, A, M. PC_9 and PC_8 are not affected by this instruction.

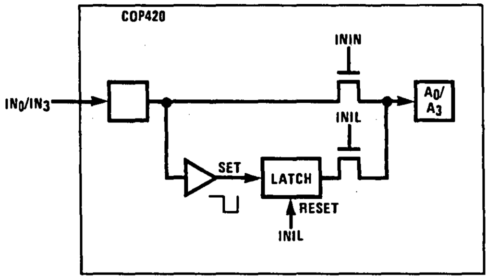
Note that JID requires 2 instruction cycles to execute.

Description of Selected Instructions (Continued)

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 10*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3-IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset.



TL/DD/6921-25

FIGURE 10

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the content of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP420/421 to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 131 kHz (crystal frequency ÷ 16) and the binary counter output pulse frequency will be 128 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 128 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP420/421 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instruction are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed except JID and LQID. LQID and JID take two cycle times if executed and one if skipped.

c. The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11 or 15 will access data in the next group of four pages.

Option List

The COP420/421/422 mask-programmable options are assigned numbers which correspond with the COP*420* pins.

The following is a list of COP420 options. When specifying a COP421 or COP422 chip, Options 9, 10, 19, 20 and 29 must all be set to zero. When specifying a COP422 chip, Options 21, 22, 27 and 28 must also be zero, and Option 2 must not be a 1. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground—no options available

Option 2: CKO Pin
 = 0: clock generator output to crystal
  0 not available if option 3 = 4 or 5)
 = 1: Pin is RAM power supply (V_R) input
  (Not available on COP422/COP322)
 = 2: general purpose input with load device
 = 4: general purpose Hi Z input

Option 3: CKI Input
 = 0: crystal input devided by 16
 = 1: crystal input divided by 8
 = 2: TTL external clock input divided by 16
 = 3: TTL external clock input divided by 8
 = 4: single-pin RC controlled oscillator (÷4)
 = 5: Schmitt trigger clock input (÷4)

Option 4: RESET Pin
 = 0: load devices to V_{CC}
 = 1: Hi-Z input

Option 5: L_7 Driver
 = 0: Standard output *(Figure 9D)*
 = 1: Open-Drain output (E)
 = 2: LED direct drive output (F)
 = 3: TRI-STATE push-pull output (G)

Option 6: L_6 Driver
 same as Option 5

Option 7: L_5 Driver
 same as Option 5

Option 8: L_4 Driver
 same as Option 5

Option 9: IN_1 Input
 = 0: load devices to V_{CC} (H)
 = 1: Hi-Z input (I)

Option 10: IN_2 Input
 same as Option 9

Option 11 = 0: V_{CC} Pin—no options available

Option 12: L_3 Driver
 same as Option 5

Option 13: L_2 Driver
 same as Option 5

Option 14: L_1 Driver
 same as Option 5

Option 15: L_0 Driver
 same as Option 5

Option 16: SI Input
 same as Option 9

Option 17: SO Driver
 = 0: standard output (A)
 = 1: open-drain output (B)
 = 2: push-pull output (C)

Option 18: SK Driver
 same as Option 17

Option 19: IN_0 Input
 same as Option 9

Option 20: IN_3 Input
 same as Option 9

Option 21: G_0 I/O Port
 = 0: Standard output (A)
 = 1: Open-Drain output (B)

Option 22: G_1 I/O Port
 same as Option 21

Option 23: G_2 I/O Port
 same as Option 21

Option 24: G_3 I/O Port
 same as Option 21

Option 25: D_3 Output
 = 0: Standard output (A)
 = 1: Open-Drain output (B)

Option 26: D_2 Output
 same as Option 25

Option 27: D_1 Output
 same as Option 25

Option 28: D_0 Output
 same as Option 25

Option 29: COP Function
 = 0: normal operation

Option 30: COP Bonding
 = 0: COP420 (28-pin device)
 = 1: COP421 (24-pin device)
 = 2: 28- and 24-pin device
 = 3: COP422 (20-pin device)
 = 4: 28- and 20-pin device
 = 5: 24- and 20-pin device
 = 6: 28-, 24- and 20-pin device

Option 31: In Input Levels
 = 0: normal input levels
 = 1: Higher voltage input levels
  ("0" = 1.2V, "1" = 3.6V)

Option 32: G Input Levels
 same as Option 31

Option 33: L Input Levels
 same as Option 31

Option 34: CKO Input Levels
 same as Option 31

Option 35: SI Input Levels
 same as Option 31

Option List (Continued)

COP OPTION LIST

The following option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE = __0__IS: GROUND PIN
OPTION 2 VALUE = _____IS: CKO PIN
OPTION 3 VALUE = _____IS: CKI INPUT
OPTION 4 VALUE = _____IS: RESET INPUT
OPTION 5 VALUE = _____IS: L_7 DRIVER
OPTION 6 VALUE = _____IS: L_6 DRIVER
OPTION 7 VALUE = _____IS: L_5 DRIVER
OPTION 8 VALUE = _____IS: L_4 DRIVER
OPTION 9 VALUE = _____IS: IN1 INPUT
OPTION 10 VALUE = _____IS: IN2 INPUT
OPTION 11 VALUE = _____IS: VCC PIN
OPTION 12 VALUE = _____IS: L_3 DRIVER
OPTION 13 VALUE = _____IS: L_2 DRIVER
OPTION 14 VALUE = _____IS: L_1 DRIVER
OPTION 15 VALUE = _____IS: L_0 DRIVER
OPTION 16 VALUE = _____IS: SI INPUT
OPTION 17 VALUE = _____IS: SO DRIVER
OPTION 18 VALUE = _____IS: SK DRIVER
OPTION 19 VALUE = _____IS: IN_0 INPUT
OPTION 20 VALUE = _____IS: IN_3 INPUT
OPTION 21 VALUE = _____IS: G_0 I/O PORT
OPTION 22 VALUE = _____IS: G_1 I/O PORT
OPTION 23 VALUE = _____IS: G_2 I/O PORT
OPTION 24 VALUE = _____IS: G_3 I/O PORT
OPTION 25 VALUE = _____IS: D_3 OUTPUT
OPTION 26 VALUE = _____IS: D_2 OUTPUT
OPTION 27 VALUE = _____IS: D_1 OUTPUT
OPTION 28 VALUE = _____IS: D_0 OUTPUT
OPTION 29 VALUE = __0__IS: COP FUNCTION
OPTION 30 VALUE = _____IS: COP BONDING
OPTION 31 VALUE = _____IS: IN INPUT LEVELS
OPTION 32 VALUE = _____IS: G INPUT LEVELS
OPTION 33 VALUE = _____IS: L INPUT LEVELS
OPTION 34 VALUE = _____IS: CKO INPUT LEVELS
OPTION 35 VALUE = _____IS: SI INPUT LEVELS

TEST MODE (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP420. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATION # 1: COP420 General Controller

Figure 8 shows an interconnect diagram for a COP420 used as a general controller. Operation of the system is as follows:

1. The L_7–L_0 outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display.
2. The D_3–D_0 outputs drive the digits of the mulitplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN_3–IN_0 inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a V_R RAM power supply pin. RAM data integrity is thereby assured when the main power supply is shut down (see RAM Keep-Alive option description).
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G_3–G_0) are available for use as required by the user's application.

APPLICATION # 2: MUSICAL ORGAN AND MUSIC BOX

Play Mode: Twenty-five musical keys and 25 LEDs are provided to denote F to F with half notes in between. All the keys and LEDs are directly detected and driven by the microprocessor. Depression of the key will give the corresponding musical note and light up the corresponding LED.

Clear: Memory is provided to store a played tune. Depression of the CLEAR key erases the memory and the microprocessor is ready to store new musical notes. A maximum of 28 notes can be stored where each note can be of one to eight musical beats. (Two bytes of memory are required to store one musical note. Any note longer than eight musical beats will require additional memory space for storage.)

Playback: Depression of this button will playback the tune stored in the memory since last "clear."

Preprogrammed Tunes: There are ten preprogrammed tunes (each has an average of 55 notes) masked in the chip. Any tune can be recalled by depressing the "Tune Button" followed by the corresponding "Sharp Key."

Learn Mode: This mode is for the player to learn the ten preprogrammed tunes. By pressing the "Learn Button" followed by the corresponding "Sharp Key," the LEDs will be lighted up one by one to indicate the notes of the selected tune. The LED will remain "on" until the player presses the correct musical key; the LED for the next note will then be lighted up.

Pause: In addition to the 25 musical keys, there is a special pause key. The depression of this key generates a blank note to the memory.

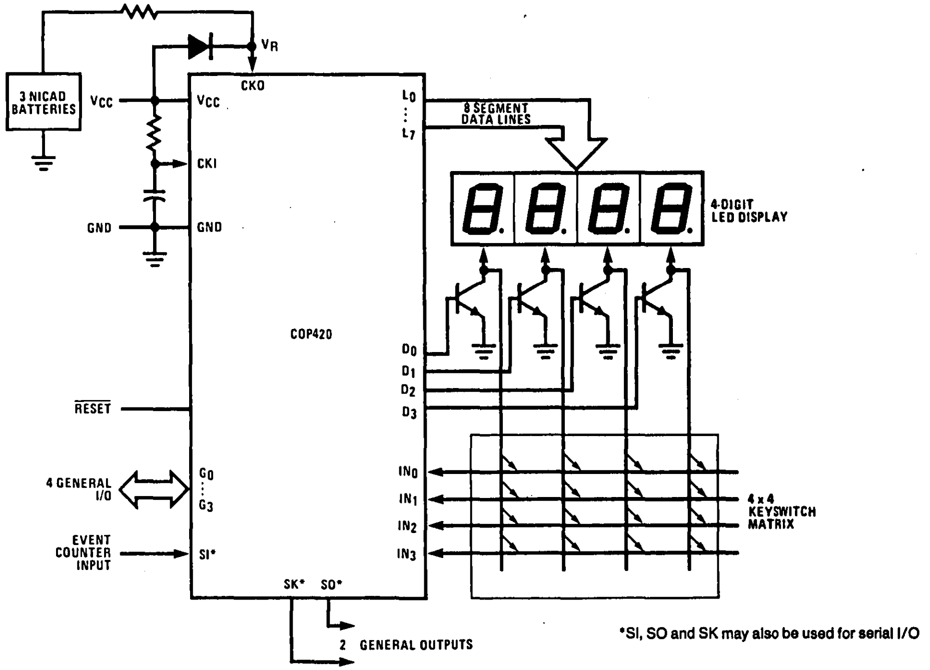
Note: In the Learn Mode when playing "Oh Susanna," the pause key must be used.

Tempo: This is a control input to the musical beat time oscillator for varying the speed of the musical tunes.

Vibrato: This is a switch control to vary the frequency vibration of the note.

Tunes Listing: The following is a listing of the ten preprogrammed tunes: 1) Jingle Bells, 2) Twinkle, Twinkle Little Star, 3) Happy Birthday, 4) Yankee Doodle, 5) Silent Night, 6) This Old Man, 7) London Bridge Is Falling Down, 8) Auld Lang Syne, 9) Oh Susanna, 10) Clementine.

Typical Applications

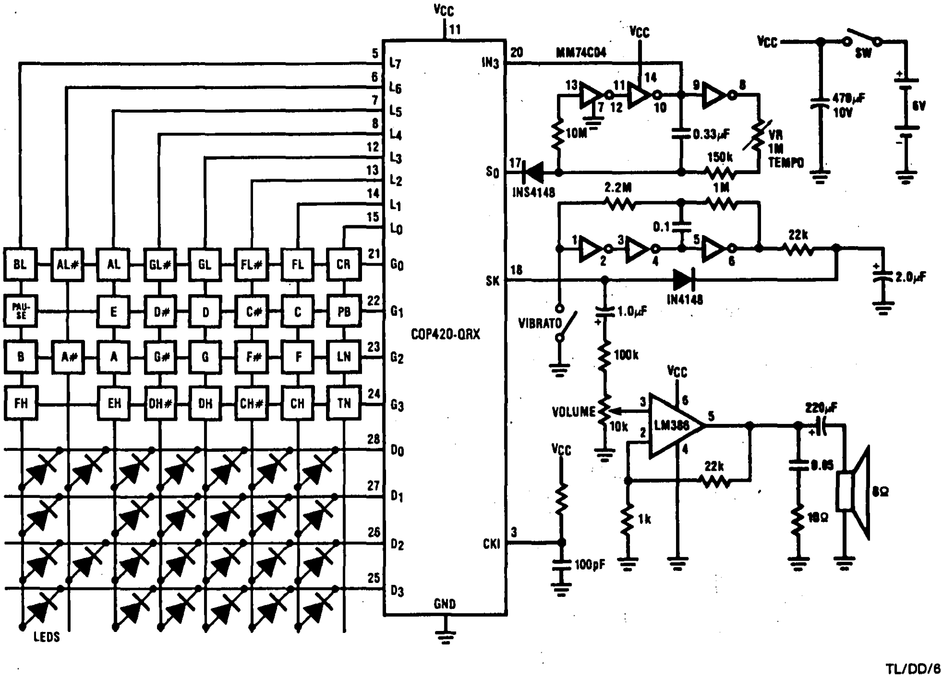


*SI, SO and SK may also be used for serial I/O

TL/DD/6921-26

FIGURE 11. COP420 Keyboard Display Interface

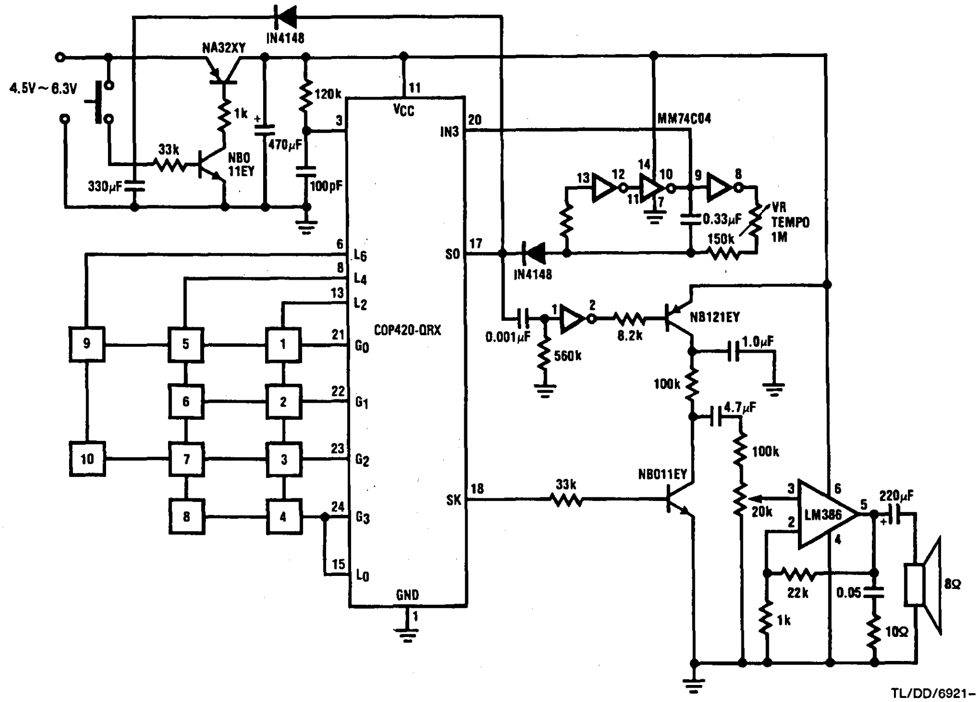
Circuit Diagram of COP420 Musical Organ



TL/DD/6921-27

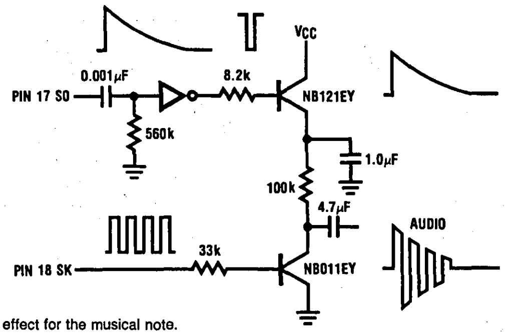
Typical Applications (Continued)

Music Box Application with Direct Key Access



TL/DD/6921-28

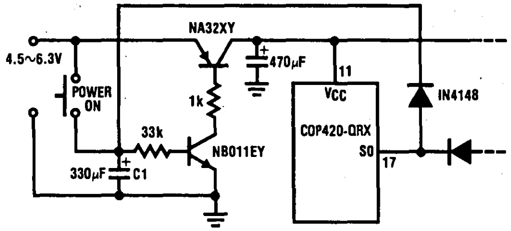
Bell Sound Circuit



This additional circuit provides tinkling effect for the musical note.

TL/DD/6921-29

Auto Power Shut-Off Circuit



TL/DD/6921-30

This circuit automatically turns off the musical organ if none of the keys are pressed within approximately 30 seconds.

National Semiconductor

COP420L/COP421L/COP422L/COP320L/COP321L/ COP322L Single-Chip N-Channel Microcontrollers

General Description

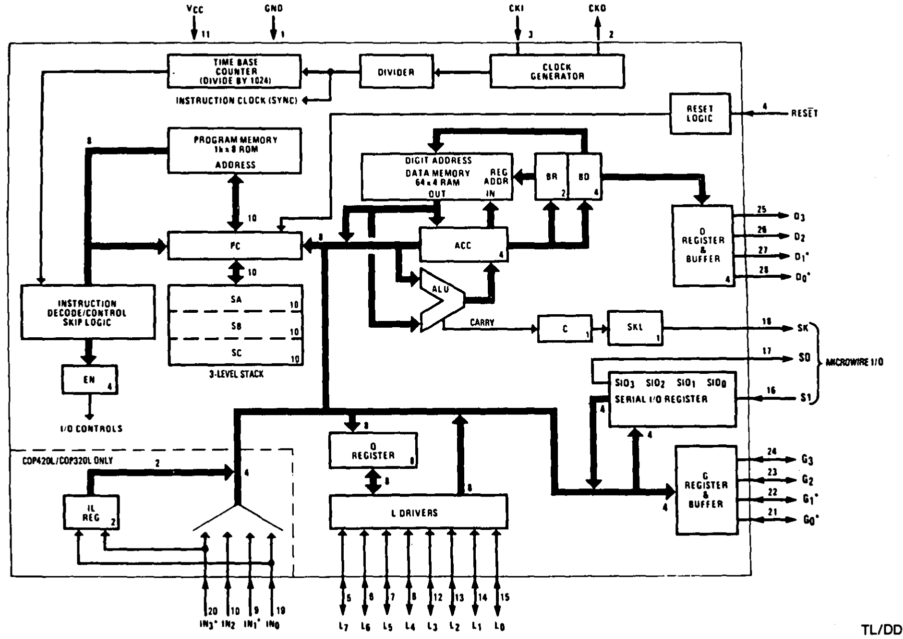
The COP420L, COP421L, COP422L, COP320L, COP321L, and COP322L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These controller oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and BCD data manipulation. The COP421L and COP422L are identical to the COP420L, but with 19 and 15 I/O lines, respectively, instead of 23. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

The COP320L, COP321L, and COP322L are exact functional equivalents, but extended temperature range versions, of the COP420L, COP421L, and COP422L respectively.

Features

- Low cost
- Powerful instruction set
- 1k x 8 ROM, 64 x 4 RAM
- 23 I/O lines (COP420L)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 16 µs instruction time
- Single supply operation (4.5V–6.3V)
- Low current drain (9 mA max)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range device— COP320L/COP321L/COP322L (−40°C to +85°C)

Block Diagram



TL/DD/8825–1

*Not available on COP422L/COP322L

FIGURE 1

COP420L/COP421L/COP422L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Power Dissipation	
COP420L/COP421L	0.75W at 25°C
	0.4W at 70°C
COP422L	0.65W at 25°C
	0.3W at 70°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		9	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷32, ÷16, ÷8)				
Logic High (V_{IH}) V_{CC} = Max		3.0		V
Logic High (V_{IH})				
V_{CC} = 5V ±5%		2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 3)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	with TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.8	V
Logic High	with High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 ma		0.4	V
CMOS Operation (Note 2)	V_{CC} = 4.5V			
Logic High	I_{OH} = −10 μA	V_{CC} −1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.8V for normal operation.

COP420L/COP421L/COP422L

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{CC} = 6.3V, V_{OL} = 0.4V	1.2		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L_0–L_7 Outputs and Standard	V_{CC} = 6.3V, V_{OL} = 0.4V	0.4		mA
G_0–G_3, D_0–D_3 Outputs (I_{OL})	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	V_{CC} = 6.3V, V_{OL} = 1.0V	11		mA
High Current Options (I_{OL})	V_{CC} = 4.5V, V_{OL} = 1.0V	7.5		mA
G_0–G_3 and D_0–D_3 Outputs with	V_{CC} = 6.3V, V_{OL} = 1.0V	22		mA
Very High Current Options (I_{OL})	V_{CC} = 4.5V, V_{OL} = 1.0V	15		mA
CKI (Single-Pin RC Oscillator)	V_{CC} = 4.5V, V_{IH} = 3.5V	2		mA
CKO	V_{CC} = 4.5V, V_{OL} = 0.4V	0.2		mA
Output Source Current				
Standard Configuration,	V_{CC} = 6.3V, V_{OH} = 2.0V	−75	−480	µA
All Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.0V	−30	−250	µA
Push-Pull Configuration	V_{CC} = 6.3V, V_{OH} = 2.4V	−1.4		mA
SO and SK Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 1.0V	−1.2		mA
LED Configuration, L_0–L_7 Outputs, Low Current Driver Option (I_{OH})	V_{CC} = 6.0V, V_{OH} = 2.0V	−1.5	−13	mA
LED Configuration, L_0–L_7 Outputs, High Current Driver Option (I_{OH})	V_{CC} = 6.0V, V_{OH} = 2.0V	−3.0	−25	mA
TRI-STATE Configuration,	V_{CC} = 6.3V, V_{OH} = 3.2V	−0.8		mA
L_0–L_7 Outputs, Low	V_{CC} = 4.5V, V_{OH} = 1.5V	−0.9		mA
Current Driver Option (I_{OH})				
TRI-STATE Configuration,	V_{CC} = 6.3V, V_{OH} = 3.2V	−1.6		mA
L_0–L_7 Outputs, High	V_{CC} = 4.5V, V_{OH} = 1.5V	−1.8		mA
Current Driver Option (I_{OH})				
Input Load Source Current	V_{CC} = 5.0V	−10	−140	µA
CKO Output RAM Power Supply Option Power Requirement	V_R = 3.3V		3.0	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	µA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

COP320L/COP321L/COP322L

Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	
COP320L/COP321L	0.75W at 25°C
	0.4W at 70°C
	0.25W at 85°C
COP322L	0.65W at 25°C
	0.20W at 70°C

Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		11	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH}) V_{CC} = Max		3.0		V
Logic High (V_{IH})				
V_{CC} = 5V ±5%		2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)	(Note 3)	2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	with TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	with High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)	V_{CC} = 4.5V			
Logic High	I_{OH} = −10 μA	V_{CC} −1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

COP320L/COP321L/COP322L

DC Electrical Characteristics

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0–L_7 Outputs and Standard	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 5.5V, V_{OL} = 1.0V$	9		mA
High Current Options (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 5.5V, V_{OL} = 1.0V$	18		mA
Very High Current Options (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 1.0V$	14		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	µA
All Outputs (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	µA
Push-Pull Configuration	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.4	−17	mA
Outputs, Low Current	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−0.7	−15	mA
Driver Option (I_{OH})				
LED Configuration, L_0–L_7	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−2.7	−34	mA
Outputs, High Current	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.4	−30	mA
Driver Option (I_{OH})				
TRI-STATE Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−0.6		mA
L_0–L_7 Outputs, Low	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
Current Driver Option (I_{OH})				
TRI-STATE Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−1.2		mA
L_0–L_7 Outputs, High	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Current Driver Option (I_{OH})				
Input Load Source Current	$V_{CC} = 5.0V$	−10	−200	µA
CKO Output				
RAM Power Supply Option	$V_R = 3.3V$		4.0	mA
Power Requirement				
TRI-STATE Output Leakage Current		−5	+5	µA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics

COP420L/COP421L/COP422L: 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted
COP320L/COP321L/COP322L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

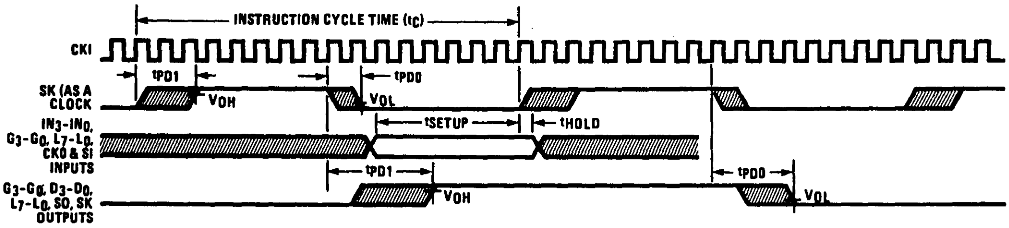
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time—t_C		16	40	µs
CKI				
Input Frequency—f_I	÷32 Mode	0.8	2.0	MHz
	÷16 Mode	0.4	1.0	MHz
	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 2 MHz		120	ns
Fall Time			80	ns

CKI Using RC (÷4)
Instruction Cycle Time (Note 1)
	R = 56 kΩ ±5%
	C = 100 pF ±10%
		16	28	µs

CKO as SYNC Input t_{SYNC}		400		ns
INPUTS:				
IN_3–IN_0, G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		µs
t_{HOLD}		1.3		µs
SI				
t_{SETUP}		2.0		µs
t_{HOLD}		1.0		µs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	µs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	µs

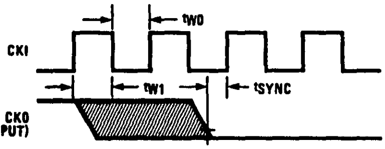
Note 1: Variation due to the device included.

Timing Diagrams


INSTRUCTION CYCLE TIME (tC)
CKI
tPD1
tPD0
SK (AS A CLOCK
VOH
VOL
IN3-IN0, G3-G0, L7-L0, CKO & SI INPUTS
tSETUP
tHOLD
tPD1
tPD0
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
VOH
VOL

TL/DD/8825-5

FIGURE 3. Input/Output Timing Diagrams (Crystal Divide-by-16 Mode)

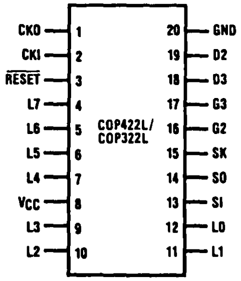


TL/DD/8825-6

FIGURE 3a. Synchronization Timing

Connection Diagrams

SO Wide and DIP



TL/DD/8825-4

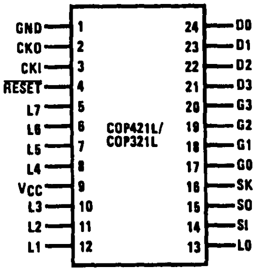
Top View

Order Number COP422L-XXX/N
or COP322L-XXX/N
See NS Molded Package Number N24A

Order Number COP322L-XXX/D
or COP422L-XXX/D
See NS Hermetic Package Number D20A

Order Number COP322L-XXX/WM
or COP422L-XXX/WM
See NS Surface Mount Package Number M20B

SO Wide and DIP



TL/DD/8825-3

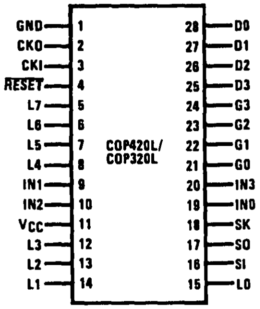
Top View

Order Number COP421L-XXX/N
or COP321L-XXX/N
See NS Molded Package Number N20A

Order Number COP321L-XXX/D
or COP421L-XXX/D
See NS Hermetic Package Number D24C

Order Number COP321L-XXX/WM
or COP421L-XXX/WM
See NS Surface Mount Package Number M24B

Dual-In Line Package



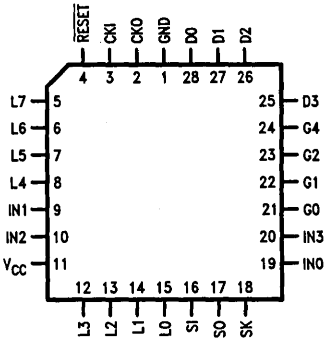
TL/DD/8825-2

Top View

Order Number COP420L-XXX/N
or COP320L-XXX/N
See NS Molded Package Number N28B

Order Number COP320L-XXX/D
or COP420L-XXX/D
See NS Hermetic Package Number D28C

PLCC



TL/DD/8825-27

Order Number COP320L-XXX/V
or COP420L-XXX/V
See NS PLCC Package Number V28A

FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs (COP420L only)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)

Pin	Description
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKO	System oscillator output (or general purpose input, RAM power supply or SYNC input)
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground

Functional Description

For ease of reading this description, only COP420L and/or COP421L are referenced; however, all such references apply also to COP320L, COP321L, COP322L, or COP422L.

A block diagram of the COP420L is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024 byte ROM. As can be seen by an examination of the COP420L/421L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binary value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 10-bit binary count value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions is based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP420/421L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunctions with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four general-purpose inputs, IN_3–IN_0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers. For example of additional parallel output capacity see Application #2.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables

Functional Description (Continued)

the L drivers, placing the L I/O ports in a high-impedance input state.

4. EN$_3$, in conjunction with EN$_0$, affects the SO output. With EN$_0$ set (binary counter option selected) SO will output the value loaded into EN$_3$. With EN$_0$ reset (serial shift register option selected), setting EN$_3$ enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN$_3$ with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides a summary of the modes associated with EN$_3$ and EN$_0$.

Enable Register Modes—Bits EN$_3$ and EN$_0$

EN$_3$	EN$_0$	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

INTERRUPT

The following features are associated with the IN$_1$ interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once aknowledged as explained below, pushes the next sequential program counter address (PC + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN$_1$ is reset.

b. An interrupt will be acknowledged only after the following conditions are met:

   1. EN$_1$ has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN$_1$ input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time,* the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt servicing routine since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

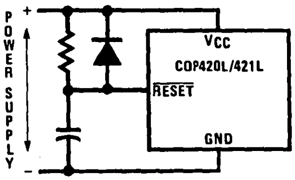
e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V$_{CC}$. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

Power-Up Clear Circuit



TL/DD/8825-7

RC ≥ 5 × Power Supply Rise Time

Functional Description (Continued)

OSCILLATOR

There are three basic clock oscillator configurations available as shown by *Figure 4.*

a. **Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32 (optional by 16 or 8).

b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 32 (optional by 16 or 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R) or as a general purpose input.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or as a general purpose input.

CKO PIN OPTIONS

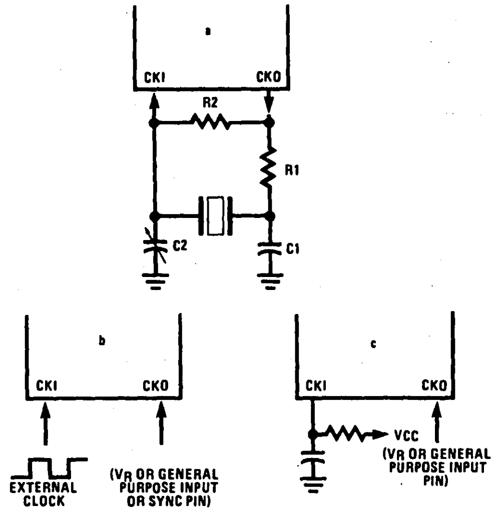
In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP420L/421L system timing configuration does not require use of the CKO pin.

RAM KEEP-ALIVE OPTION (Not available on COP422L)

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. During normal operation V_R must be within the operating range of the chip, with $(V_{CC} - 1) \leq V_R \leq V_{CC}$.
3. V_R must be $\geq$ 3.3V with V_{CC} off.

Crystal Oscillator


a
CKI
CKO
R2
R1
C2
C1
b
CKI
CKO
EXTERNAL CLOCK
(VR OR GENERAL PURPOSE INPUT OR SYNC PIN)
c
CKI
CKO
VCC
(VR OR GENERAL PURPOSE INPUT PIN)

TL/DD/8825-8

Crystal Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220
2.097 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (μs)
51	100	19 ±15%
82	56	19 ± 13%

Note: 200k ≥ R ≥ 25k
360 pF ≥ C ≤ 50 pF

FIGURE 4. COP420L/421L Oscillator

Functional Description (Continued)

I/O OPTIONS

COP420L/421L outputs have the following optional configurations, illustrated in *Figure 5*:

a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. **Standard L**—same as **a.**, but may be disabled. Available on L outputs only.

e. **Open Drain L**—same as **b.**, but may be disabled. Available on L outputs only.

f. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

g. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

COP420L/COP421L inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP420L/421L system.

The SO, SK outputs can be configured as shown in **a.**, **b.**, or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured as in **d.**, **e.**, **f.** or **g.**

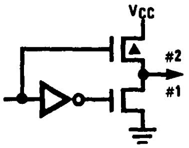
An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L lines are used as inputs, the disabled depletion device *cannot* be relied on to source sufficient current to pull an input to a logic 1.

COP421L

If the COP420L is bonded as a 24-pin device, it becomes the COP421L, illustrated in *Figure 2*, COP420L/421L Connection Diagrams. Note that the COP421L does not contain the four general purpose IN inputs (IN_3–IN_0). Use of this option precludes, of course, use of the IN options and the interrupt feature. All other options are available for the COP421L.

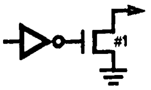
COP422L

If the COP421L is bonded as a 20-pin device, it becomes the COP422L, as illustrated in *Figure 2*. Note that the COP422L contains all the COP421L pins except D_0, D_1, G_0, and G_1. COP422L also does not allow RAM power supply input as a valid CKO pin option.



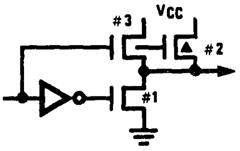
TL/DD/8825-9

a. Standard Output



TL/DD/8825-10

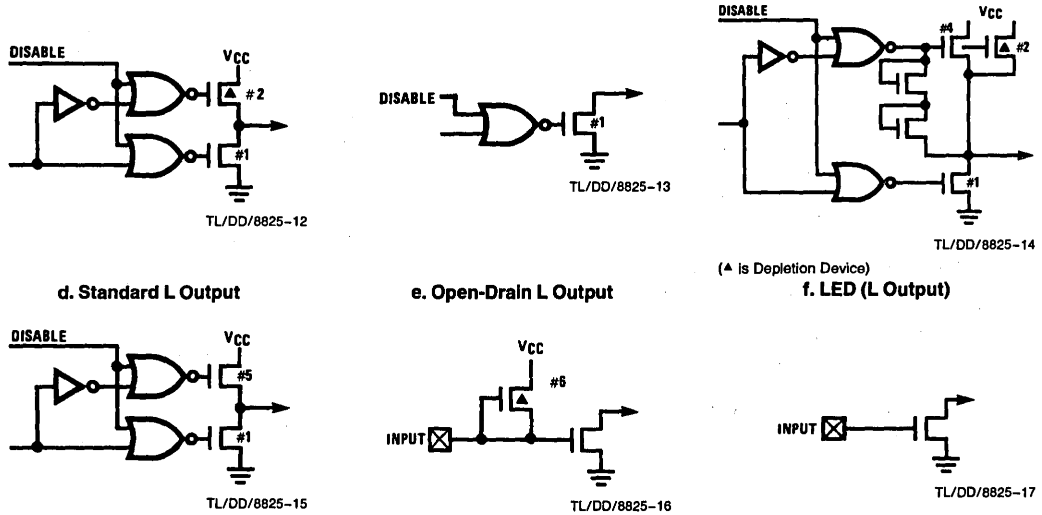
b. Open-Drain Output



TL/DD/8825-11

c. Push-Pull Output

Functional Description (Continued)



d. Standard L Output

e. Open-Drain L Output

(▲ is Depletion Device)

f. LED (L Output)

g. TRI-STATE Push-Pull (L Output)

h. Input with Load

i. HI-Z Input

FIGURE 5. Output Configurations

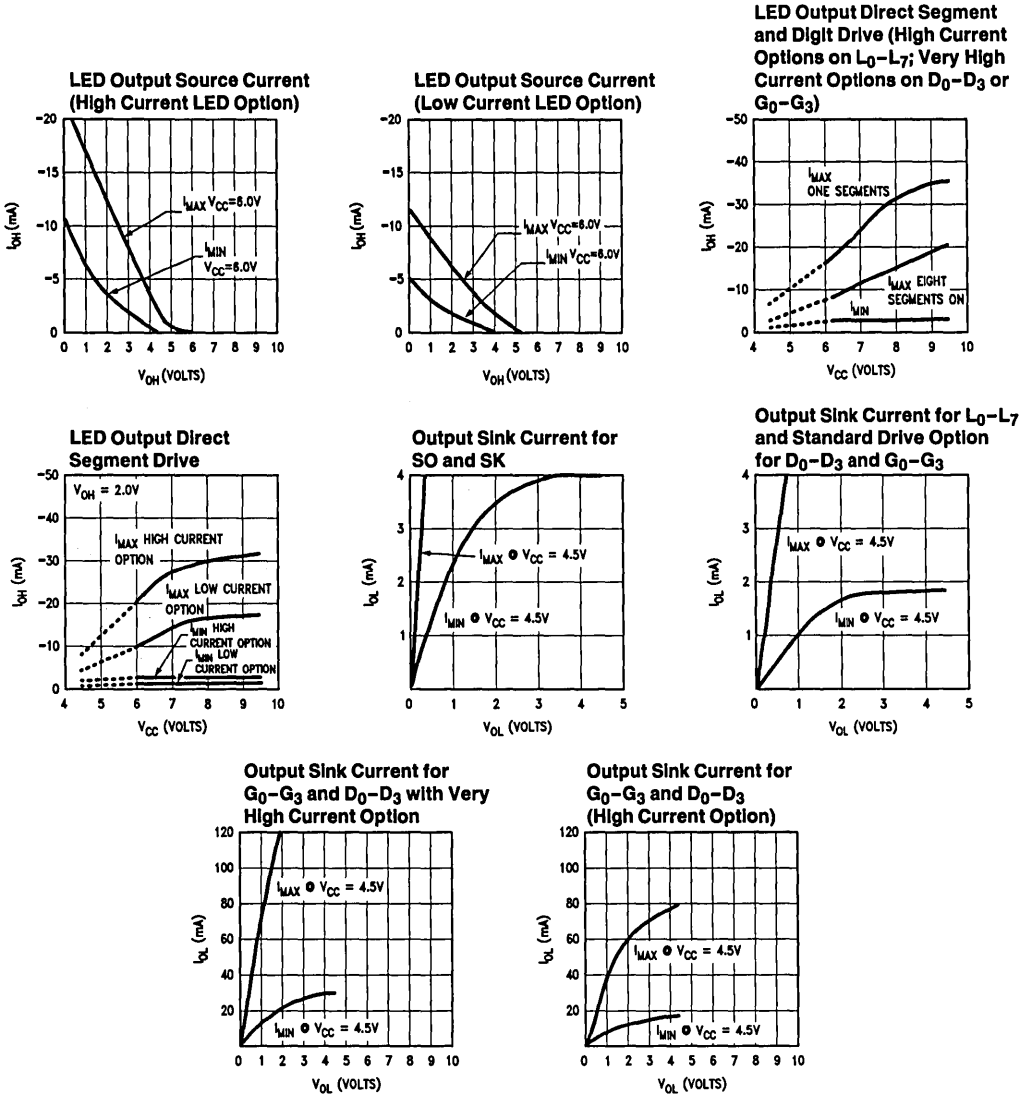
L-Bus Considerations

False states may be generated on L_0–L_7 during the execution of the CAMQ instruction. The L-ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. the following short program that illustrates this situation.

```
START:
         CLRA          ;ENABLE THE Q
         LEI    4      ;REGISTER TO L LINES
         LBI    TEST
         STII   3
         AISC   12
LOOP:
         LBI    TEST   ;LOAD Q WITH X'C3
         CAMQ
         JP     LOOP
```

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L_0, L_1, L_6, L_7, and logic lows on L_2–L_5 via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L_0, L_1, L_6, L_7, and positive glitches on L_2–L_5. Glitch durations are under 2 μs, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics (Continued)



LED Output Source Current (High Current LED Option)

LED Output Source Current (Low Current LED Option)

LED Output Direct Segment and Digit Drive (High Current Options on L_0–L_7; Very High Current Options on D_0–D_3 or G_0–G_3)

LED Output Direct Segment Drive

Output Sink Current for SO and SK

Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3

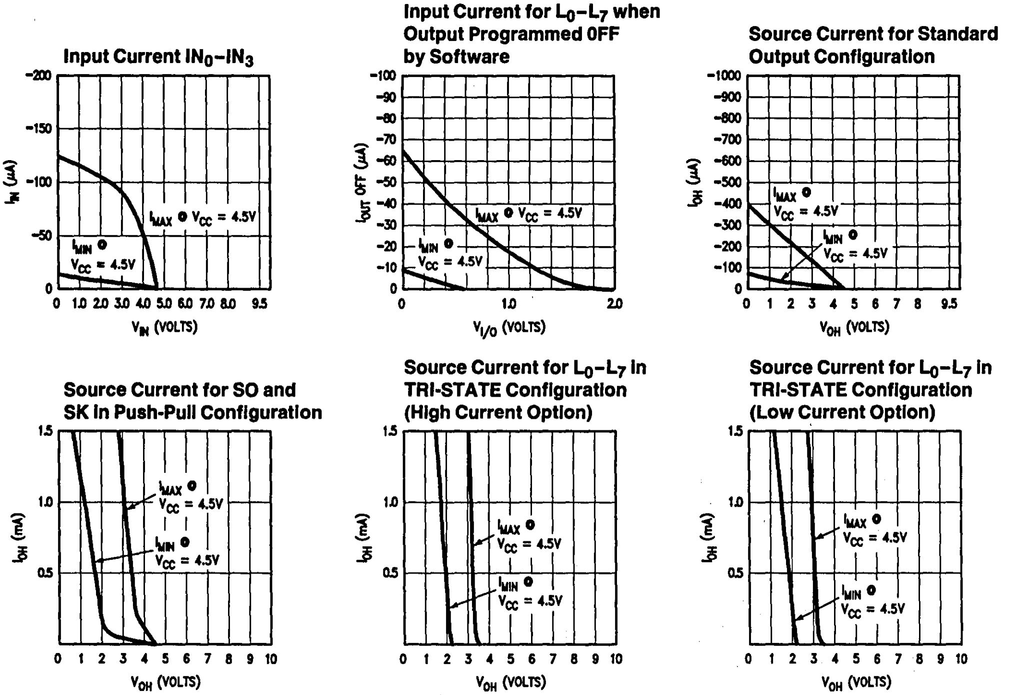
Output Sink Current for G_0–G_3 and D_0–D_3 with Very High Current Option

Output Sink Current for G_0–G_3 and D_0–D_3 (High Current Option)

TL/DD/8825-19

FIGURE 6. COP420L/COP421L/COP422L Input/Output Characteristics

Typical Performance Characteristics (Continued)



Input Current IN_0–IN_3

Input Current for L_0–L_7 when Output Programmed OFF by Software

Source Current for Standard Output Configuration

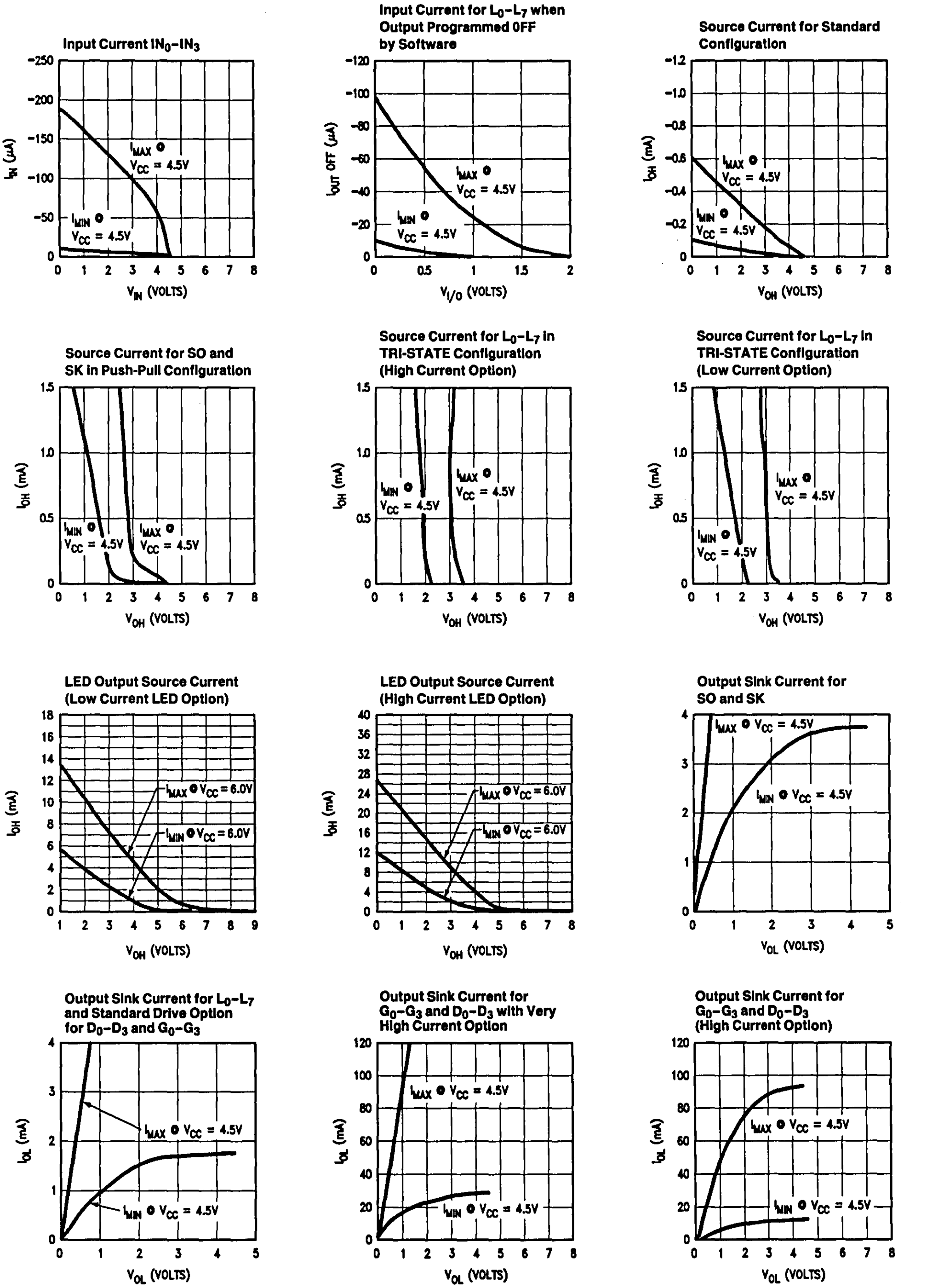
Source Current for SO and SK in Push-Pull Configuration

Source Current for L_0–L_7 in TRI-STATE Configuration (High Current Option)

Source Current for L_0–L_7 in TRI-STATE Configuration (Low Current Option)

TL/DD/8825-18

Typical Performance Characteristics (Continued)



Input Current IN_0–IN_3

Input Current for L_0–L_7 when Output Programmed OFF by Software

Source Current for Standard Configuration

Source Current for SO and SK in Push-Pull Configuration

Source Current for L_0–L_7 in TRI-STATE Configuration (High Current Option)

Source Current for L_0–L_7 in TRI-STATE Configuration (Low Current Option)

LED Output Source Current (Low Current LED Option)

LED Output Source Current (High Current LED Option)

Output Sink Current for SO and SK

Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3

Output Sink Current for G_0–G_3 and D_0–D_3 with Very High Current Option

Output Sink Current for G_0–G_3 and D_0–D_3 (High Current Option)

TL/DD/8825–20

FIGURE 7. COP320L/DOP321L/COP322L Input/Output Characteristics

COP420L/COP421L Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP410L/411L instruction set.

TABLE I. COP420L/421L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit Latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	10-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	10-bit Operand Field, 0–1023 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\bar{A}$	The ones complement of A
⊕	Exclusive-OR
:	Range of values

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
ADT		4A	0100 1010	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	0101 y	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	0001 0000	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Compliment and Add with Carry, Skip on Carry
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{9:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– – –	0110 00 $a_{9:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	– –	1 $a_{6:0}$ (pages 2,3 only)	a → $PC_{6:0}$	None	Jump within Page (Note 4)
			or			
		– –	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → SA → SB → SC 0010 → $PC_{9:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	0110 10 $a_{9:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33	0011 0011	A → $Q_{7:4}$	None	Copy A, RAM to Q
		3C	0011 1100	RAM(B) → $Q_{3:0}$		
CQMA		33	0011 0011	$Q_{7:4}$ → RAM(B)	None	Copy Q to RAM, A
		2C	0010 1100	$Q_{3:0}$ → A		
LD	r	-5	00 r 0101	RAM(B) → A	None	Load RAM into A, Exclusive-OR Br with r
				Br ⊕ r → Br		
LDD	r,d	23	0010 0011	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
		--	00 r d			
LQID		BF	1011 1111	ROM($PC_{9:8}$,A,M) → Q	None	Load Q Indirect (Note 3)
				SB → SC		
RMB	0	4C	0100 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	0 → $RAM(B)_1$		
	2	42	0100 0010	0 → $RAM(B)_2$		
	3	43	0100 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 1101	1 → $RAM(B)_1$		
	2	46	0100 0110	1 → $RAM(B)_2$		
	3	4B	0100 1011	1 → $RAM(B)_3$		
STII	y	7-	0111 y	y → RAM(B)	None	Store Memory Immediate and Increment Bd
				Bd + 1 → Bd		
X	r	-6	00 r 0110	RAM(B) ←→ A	None	Exchange RAM with A, Exclusive-OR Br with r
				Br ⊕ r → Br		
XAD	r,d	23	0010 0011	RAM(r,d) ←→ A	None	Exchange A with RAM pointed to directly by (r,d)
		--	10 r d			
XDS	r	-7	00 r 0111	RAM(B) ←→ A	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
				Bd – 1 → Bd		
				Br ⊕ r → Br		
XIS	r	-4	00 r 0100	RAM(B) ←→ A	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
				Bd + 1 → Bd		
				Br ⊕ r → Br		

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	- - 33 - -	00 r (d−1) (d=0,9:15) or 0011 0011 10 r d (any d)	r,d → B	Skip until not an LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6–	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A ←→ Br (0,0 → A_3,A_2)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0 1 2 3	01 11 03 13	0000 0001 0001 0001 0000 0011 0001 0011	} 2nd byte	$G_0 = 0$ $G_1 = 0$ $G_2 = 0$ $G_3 = 0$	
SKMBZ	0 1 2 3	01 11 03 13	0000 0001 0001 0001 0000 0011 0001 0011		$RAM(B)_0 = 0$ $RAM(B)_1 = 0$ $RAM(B)_2 = 0$ $RAM(B)_3 = 0$	Skip if RAM Bit is Zero
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE II. COP420L/421L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN → A	None	Input IN Inputs to A (Note 2)
INIL		33 29	0011 0011 0010 1001	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5-	0011 0011 0101 y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is only available on the 28-pin COP420L as the other devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP420L/421L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, $PC_{9:8}$, A, M. PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

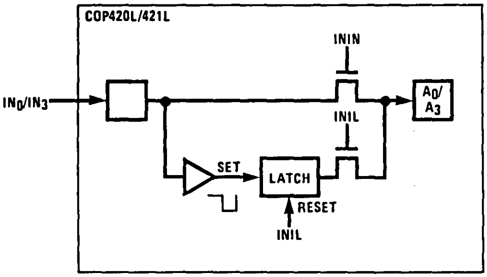
Description of Selected Instructions (Continued)

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 8*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction. IL latches are *not cleared* on reset.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note the LQID takes two instruction cycle times to execute.



TL/DD/8825-21

FIGURE 8. INIL Hardware Implementation

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP420L/421L to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷ 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP420L/421L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, or 15 will access data in the next group of four pages.

Option List

The COP420L/421L mask-programmable options are assigned numbers which correspond with the COP*420L* pins.

The following is a list of COP420L options. When specifying a COP421L chip, Options 9, 10, 19, and 20 must all be set to zero. When specifying a COP422L chip, options 9, 10, 19, and 20 must all be set to zero; options 21 and 22 may not be set to one, three or five; and option 2 may not be set to one. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

The Option Table should be copied and sent in with your EPROM or disc.

Option 1 = 0: Ground Pin—no options available

Option 2: CKO Output
- = 0: clock generator output to crystal/resonator (0 not allowable value if Option 3 = 3)
- = 1: pin is RAM power supply (V_R) input (not available on the COP422L)
- = 2: general purpose input with load device to V_{CC}
- = 3: general purpose input, Hi-Z

Option 3: CKI Input
- = 0: oscillator input divided by 32 (2 MHz max.)
- = 1: oscillator input divided by 16 (1 MHz max.)
- = 2: oscillator input divided by 8 (500 kHz max.)
- = 3: single-pin RC controlled oscillator (÷4)
- = 4: Schmitt trigger clock input (÷4)

Option 4: RESET Input
- = 0: load device to V_{CC}
- = 1: Hi-Z Input

Option 5: L_7 Driver
- = 0: Standard output
- = 1: Open-drain output
- = 2: High current LED direct segment drive output
- = 3: High current TRI-STATE push-pull output
- = 4: Low-current LED direct segment drive output
- = 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: IN_1 Input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN_2 Input
same as Option 9

Option 11: V_{CC} pin
- = 0: Standard V_{CC}

Option 12: L_3 Driver
same as Option 5

Option 13: L_2 Driver
same as Option 5

Option 14: L_1 Driver
same as Option 5

Option 15: L_0 Driver
same as Option 5

Option 16: SI Input
same as Option 9

Option 17: SO Driver
- = 0: standard output
- = 1: open-drain output
- = 2: push-pull output

Option 18: SK Driver
same as Option 17

Option 19: IN_0 Input
same as Option 9

Option 20: IN_3 Input
same as Option 9

Option 21: G_0 I/O Port
- = 0: very-high current standard output
- = 1: very-high current open-drain output
- = 2: high current standard output
- = 3: high current open-drain output
- = 4: standard LSTTL output (fanout = 1)
- = 5: open-drain LSTTL output (fanout = 1)

Option 22: G_1 I/O Port
same as Option 21

Option 23: G_2 I/O Port
same as Option 21

Option 24: G_3 I/O Port
same as Option 21

Option 25: D_3 Output
same as Option 21

Option 26: D_2 Output
same as Option 21

Option 27: D_1 Output
same as Option 21

Option 28: D_0 Output
same as Option 21

Option 29: L Input Levels
- = 0: standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
- = 1: higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 30: IN Input Levels
same as Option 29

Option 31: G Input Levels
same as Option 29

Option 32: SI Input Levels
same as Option 29

Option 33: $\overline{\text{RESET}}$ Input
- = 0: Schmitt trigger input
- = 1: standard TTL input levels
- = 2: higher voltage input levels

Option 34: CKO Input Levels
(CKO = input; Option 2 = 2,3)
same as Option 29

Option 35: COP Bonding
- = 0: COP420L (28-pin device)
- = 1: COP421L (24-pin device)
- = 2: 28- and 24-pin versions
- = 3: COP422L (20-pin device)
- = 4: 28- and 20-pin versions
- = 5: 24- and 20-pin versions
- = 5: 28-, 24-, and 20-pin versions

Option 36: Internal Initialization Logic
- = 0: normal operation
- = 1: no internal initialization logic

Option Table

The following EPROM option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE = ___0___ IS: GROUND PIN

OPTION 2 VALUE = ________ IS: CKO OUTPUT

OPTION 3 VALUE = ________ IS: CKI INPUT

OPTION 4 VALUE = ________ IS: RESET INPUT

OPTION 5 VALUE = ________ IS: L_7 DRIVER

OPTION 6 VALUE = ________ IS: L_6 DRIVER

OPTION 7 VALUE = ________ IS: L_5 DRIVER

OPTION 8 VALUE = ________ IS: L_4 DRIVER

OPTION 9 VALUE = ________ IS: IN1 INPUT

OPTION 10 VALUE = ________ IS: IN2 INPUT

OPTION 11 VALUE = ___0___ IS: VCC PIN

OPTION 12 VALUE = ________ IS: L_3 DRIVER

OPTION 13 VALUE = ________ IS: L_2 DRIVER

OPTION 14 VALUE = ________ IS: L_1 DRIVER

OPTION 15 VALUE = ________ IS: L_0 DRIVER

OPTION 16 VALUE = ________ IS: SI INPUT

OPTION 17 VALUE = ________ IS: SO DRIVER

OPTION 18 VALUE = ________ IS: SK DRIVER

OPTION DATA

OPTION 19 VALUE = ________ IS: IN_0 INPUT

OPTION 20 VALUE = ________ IS: IN_3 INPUT

OPTION 21 VALUE = ________ IS: G_0 I/O PORT

OPTION 22 VALUE = ________ IS: G_1 I/O PORT

OPTION 23 VALUE = ________ IS: G_2 I/O PORT

OPTION 24 VALUE = ________ IS: G_3 I/O PORT

OPTION 25 VALUE = ________ IS: D_3 OUTPUT

OPTION 26 VALUE = ________ IS: D_2 OUTPUT

OPTION 27 VALUE = ________ IS: D_1 OUTPUT

OPTION 28 VALUE = ________ IS: D_0 OUTPUT

OPTION 29 VALUE = ________ IS: L INPUT LEVELS

OPTION 30 VALUE = ________ IS: IN INPUT LEVELS

OPTION 31 VALUE = ________ IS: G INPUT LEVELS

OPTION 32 VALUE = ________ IS: SI INPUT LEVELS

OPTION 33 VALUE = ________ IS: $\overline{\text{RESET}}$ INPUT

OPTION 34 VALUE = ________ IS: CKO INPUT LEVELS

OPTION 35 VALUE = ________ IS: COP BONDING

OPTION 36 VALUE = ________ IS: INTERNAL INITIALIZATION LOGIC

TEST MODE (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the customer-programmed COP420L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

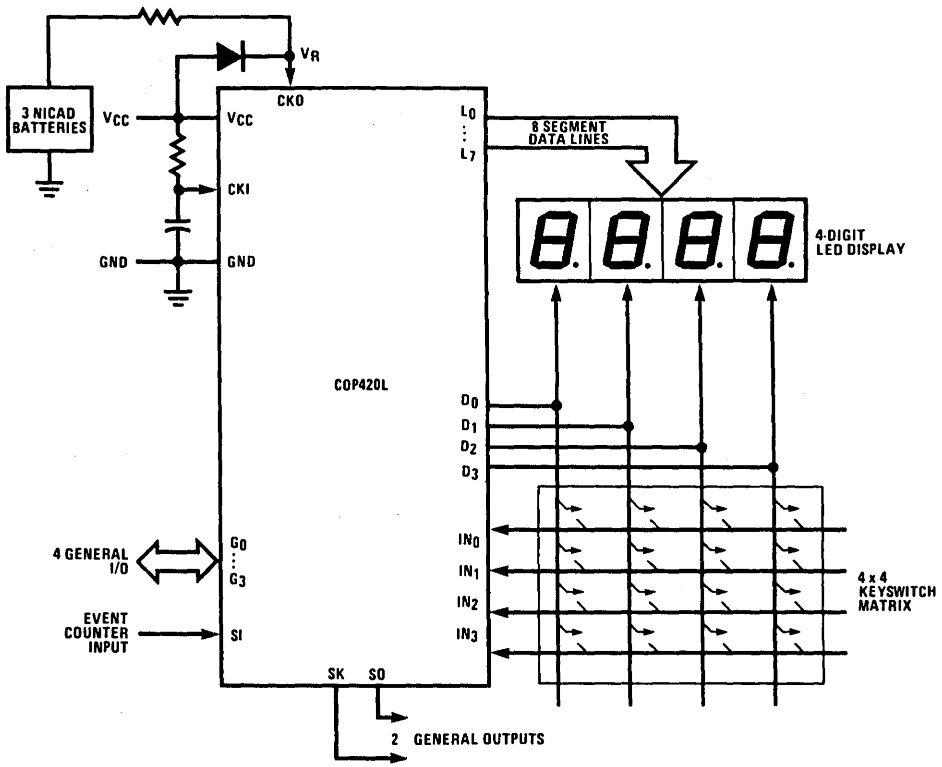
These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATIONS # 1: COP420L General Controller

Figure 9 shows an interconnect diagram for a COP420L used as a general controller. Operation of the system is as follows:

1. The L_7–L_0 outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display.
2. The D_3–D_0 outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN_3–IN_0 inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a V_R RAM power supply pin. RAM data integrity is thereby assured when the main power supply is shut down (see RAM Keep-Alive option description).
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G_3–G_0) are available for use as required by the user's application.

Typical Applications



TL/DD/8825-22

*SO, SI, SK may also be used for Serial I/O

FIGURE 9. COP420L Keyboard/Display Interface

APPLICATION #2:

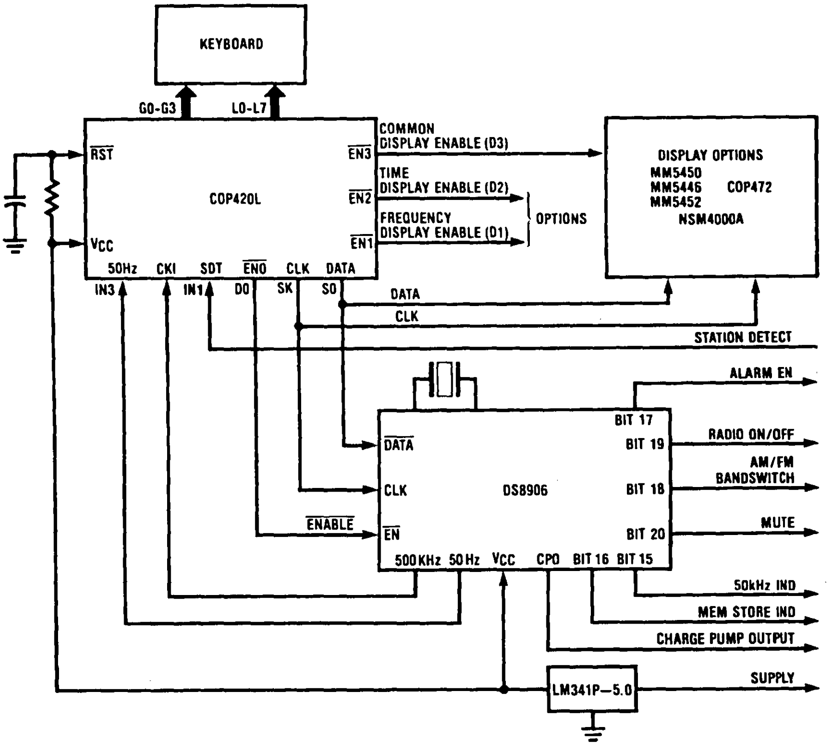
Digitally Tuned Radio Controller and Clock

Keyboard Matrix Configuration

	G0	G1	G2	G3
L7	MEMORY SCAN	MEMORY STORE	OFF	STRAP 0
L6	DOWN	HALT		
L5	UP	SEARCH	AM/FM	STRAP 1
L4	MEM5	MEM10		STRAP 2
L3	MEM4	MEM9	SLEEP	STRAP 3
L2	MEM3	MEM8	ALARM	STRAP 4
L1	MEM2	MEM7	HOUR	STRAP 5
L0	MEM1	MEM6	MINUTE	STRAP 6

TL/DD/8825-23

Typical Applications (Continued)



TL/DD/8825-24

FIGURE 10. Digital Tuning System Block

Functional Description

LOGIC I/Os

CKI Input: This input accepts an external 500 kHz signal, divides it by eight and outputs the quotient at the CLK output as the system clock.

$\overline{\text{RST}}$ Input: Schmitt trigger input to clear device upon initialization.

SDT Input: Interrupt input for station detection. The SDT signal is generated by the radio's station detector and used by the COP420L to determine if there is a valid station on the active frequency. The status of the SDT input is only relevant during station searching mode. A high on SDT will temporarily terminate the search mode for eight seconds.

ALM Input: A high on ALM will activate alarm output via slave device at alarm time. A low on the input will disable alarm function.

DATA Output: Push-pull output providing serial data to external devices.

CLK Output: Push-pull output providing system clock at data transmitting time.

50 Hz Input: A normally high input to accept a 50 Hz external time base for real-time calculation.

MOMENTARY KEYS DESCRIPTION

MEM 1–MEM 10: Each memory represents data of a favorite station in a certain band. Depression of one of these keys will recall the previous stored data and transmit it to the PLL. The PLL will in turn change the radio's receiving frequency as well as the band if necessary. Memory recall keys can also turn on the radio.

UP: This key will manually increment receiving frequency. The first four steps of increment will be for fine tuning a station, after which will be fast slewing meant for manual receive frequency changing.

DOWN: Has the same function as UP key except that frequency is decremented.

MEMORY SCAN: This will start the radio scanning through all ten memories automatically at eight seconds per memory starting from Memory 1. This will also turn on the radio if it was off.

MEMORY STORE: Enables the memory store mode which lasts for three seconds. Depression of any memory key will store the active frequency and band in that memory and disable the store mode. Any function key will also disable the mode to prevent memory data being accidentally destroyed.

HALT: Depression of the HALT key will stop the search and scan functions at current frequency or memory. HALT also turns on the radio during off time and recall frequency display in signal display mode.

SEARCH: Activates station searching in the current band. Search speed is 50 ms per frequency step with wrapping

Functional Description (Continued)

around at end of band. An 8-second stop will take place on reaching a valid station. The HALT key or any function key will terminate the search. Search direction will normally be upwards unless the DOWN key has been depressed prior to the SEARCH key or during the search function in which case search direction will be downwards.

OFF: Turns off the radio or alarm when active.

AM/FM: Radio band switch.

SLEEP: Activates sleep mode, turns on radio on depression and off radio at the end of sleep period. Setting of sleep period is done by depressing the SLEEP and MINUTE key simultaneously.

ALARM: Enables alarm time setting. Depressing the HOUR or MINUTE key and ALARM key simultaneously will set the alarm hour and minute respectively.

HOUR: Sets the hour digits of time-related functions.

MINUTE: Sets the minute digits of time-related functions.

DIODE STRAPS CONNECTIONS

STRAP 0: Controls the on and off of radio. In applications where a toggle type ON/OFF switch is used, momentary OFF key can be omitted; connecting the strap will turn on the radio and vice versa. Must be connected to use momentary OFF key.

STRAP 1, 2: Selects the AM IF options.

STRAP 3: 12/24-hour clock select.

STRAP 4: 3/5 kHz AM step size select.

STRAP 5, 6: FM IF offsets select.

	STRAP 0	STRAP 3	STRAP 4
Connected	Radio ON	12 hour	5 kHz step
Open	Radio OFF	24 hour	3 kHz step

AM/FM IF OPTIONS

AM	STRAP 1	STRAP 2
455 kHz	X	X
460 kHz	X	✓
450 kHz	✓	X
260 kHz	✓	✓
FM	**STRAP 5**	**STRAP 6**
10.7 MHz	X	X
10.75 MHz	X	✓
10.65 MHz	✓	X
10.8 MHz	✓	✓

X = No connection.

✓ = Diode inserted.

INDIRECT FEATURES AND OPTIONS

As indicated in *Figure 10,* there are a few options and indirect features provided via the help of a slave device, namely the Phase Lock Loop, DS8906N.

DISPLAY OPTIONS

As mentioned above, the COP420L-HSB is MICROWIRE compatible. Internal circuitry enables it to directly interface with all of National's serial input MICROWIRE compatible display drivers whether they are of a direct drive or multiplex drive format. On *Figure 10* is a list of drivers available for the system. EN1 and EN2 are optional enable outputs meant for a dual display system in which EN3 will not be used. By dual display, it means that one display will be constantly showing time information and the other showing frequency information. Whereas in conventional single display systems, the display shows both time and frequency information in a time-sharing method. The National system provides a time-prioritized display-sharing method. That is, whenever a tuning function is completed, the frequency information will stay on the display for eight seconds then time display will take over. This is achieved by using EN3 for the driver's enable logic.

CONTROL OUTPUTS

Six open collector outputs controlled by the COP420L are provided from DS8906N, the phase lock loop for controlling radio switching circuits.

Radio ON/OFF: A high from this output indicates that the radio should be switched on and vice versa.

AM/FM: Output for controlling the AM/FM bandswitch. A high level output indicates FM and a low indicates the AM band.

MUTE: For muting the audio output when performing any frequency related function. The output will go high prior to the frequency change except when doing fine tuning.

ALARM ENABLE: Active high output for turning on the alarm circuit at alarm time.

50 kHz IND: For driving the 50 kHz indicator in FM band or the LSB in a 5-digit display. Output is active high.

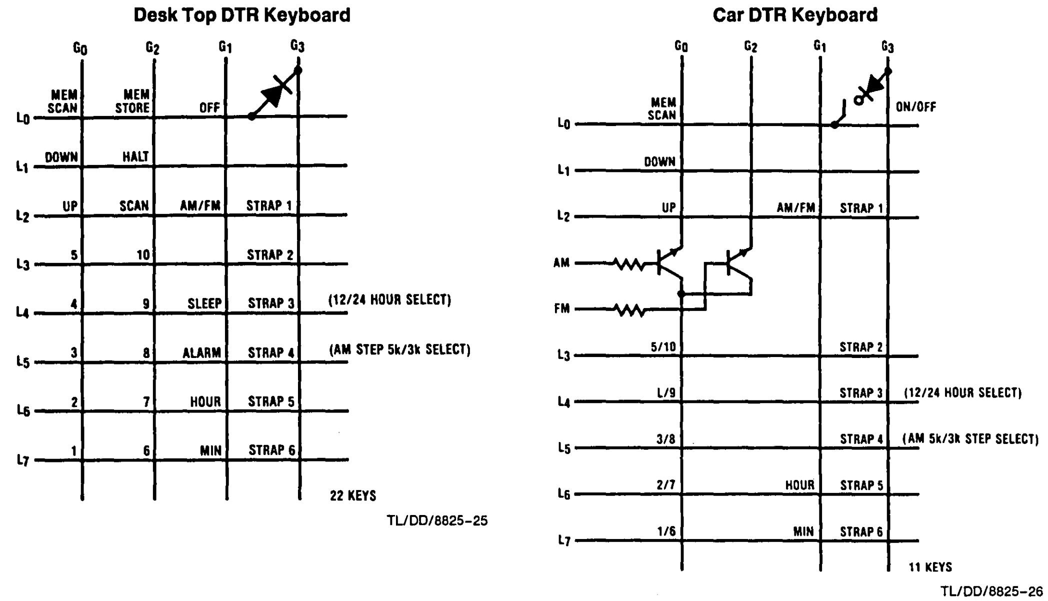
MEM STORE IND: For driving the memory store mode indicator. Output is active high.

TYPICAL IMPLEMENTATION ALTERNATIVES

A full keyboard or any portion of it can be implemented with various applications for features/functions vs. cost/size.

Figure 11 shows two keyboard configurations with 22-key and 11-key keyboards for a desk-top/tuner system or auto-radio system, respectively.

Functional Description (Continued)


Desk Top DTR Keyboard

Car DTR Keyboard

FIGURE 11

National Semiconductor

COP424C, COP425C, COP426C, COP324C, COP325C, COP326C and COP444C, COP445C, COP344C, COP345C Single-Chip 1k and 2k CMOS Microcontrollers

General Description

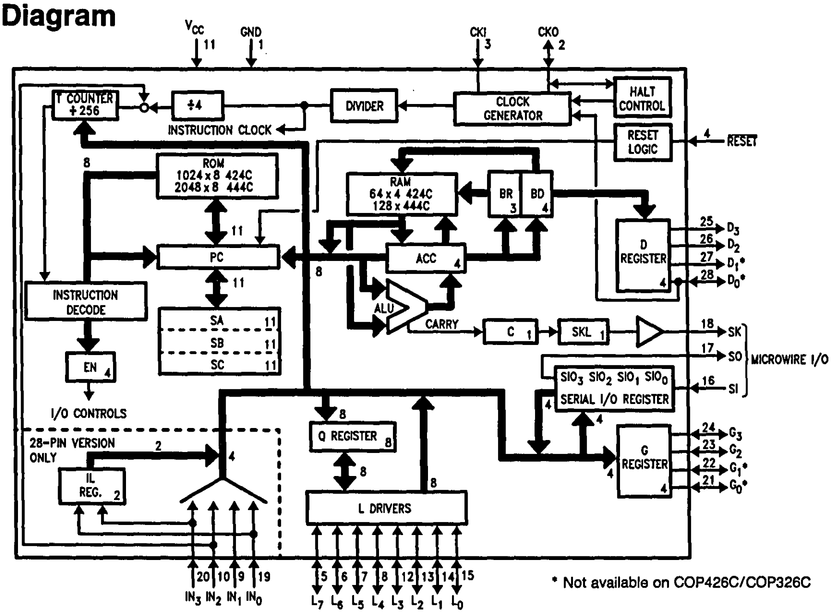
The COP424C, COP425C, COP426C, COP444C and COP445C fully static, Single-Chip CMOS Microcontrollers are members of the COPS™ family, fabricated using double-poly, silicon gate microCMOS technology. These Controller Oriented Processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP424C and COP444C are 28 pin chips. The COP425C and COP445C are 24-pin versions (4 inputs removed) and COP426C is 20-pin version with 15 I/O lines. Standard test procedures and reliable high-density techniques provide the medium to large volume customers with a customized microcontroller at a low end-product cost. These microcontrollers are appropriate choices in many demanding control environments especially those with human interface.

The COP424C is an improved product which replaces the COP420C.

Features

- Lowest power dissipation (50 μW typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4 μs instruction time, plus software selectable clocks
- 2k x 8 ROM, 128 x 4 RAM (COP444C/COP445C)
- 1k x 8 ROM, 64 x 4 RAM (COP424C/COP425C/COP426C)
- 23 I/O lines (COP444C and COP424C)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (2.4V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS output compatible
- Microbus™ compatible
- Software/hardware compatible with COP400 family
- Extended temperature range devices COP324C/COP325C/COP326C and COP344C/COP345C (−40°C to +85°C)
- Military devices (−55°C to +125°C) to be available

Block Diagram



* Not available on COP426C/COP326C

TL/DD/5259-1

FIGURE 1

COP424C/COP425C/COP426C and COP444C/COP445C

Absolute Maximum Ratings

Supply Voltage (V_{CC})	6V
Voltage at any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \leq T_A \leq 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple (Note 5)	Peak to Peak		0.1 V_{CC}	V
Supply Current	$V_{CC}=2.4V$, tc = 64 μs		120	μA
(Note 1)	$V_{CC}=5.0V$, tc = 16 μs		700	μA
	$V_{CC}=5.0V$, tc = 4 μs		3000	μA
	(tc is instruction cycle time)			
HALT Mode Current	$V_{CC}=5.0V$, $F_{IN}=0$ kHz		40	μA
(Note 2)	$V_{CC}=2.4V$, $F_{IN}=0$ kHz		12	μA
Input Voltage Levels				
$\overline{RESET}$, CKI, D_0 (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	$V_{CC}=4.5V$, $V_{IN}=0$	30	330	μA
Hi-Z Input Leakage		−1	+1	μA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels	Standard Outputs			
LSTTL Operation	$V_{CC}=5.0V \pm 10\%$			
Logic High	$I_{OH}=-100$ μA	2.7		V
Logic Low	$I_{OL}=400$ μA		0.4	V
CMOS Operation				
Logic High	$I_{OH}=-10$ μA	$V_{CC}-0.2$		V
Logic Low	$I_{OL}=10$ μA		0.2	V
Output Current Levels (except CKO)				
Sink (Note 6)	$V_{CC}=4.5V$, $V_{OUT}=V_{CC}$	1.2		mA
	$V_{CC}=2.4V$, $V_{OUT}=V_{CC}$	0.2		mA
Source (Standard Option)	$V_{CC}=4.5V$, $V_{OUT}=0V$	−0.5		mA
	$V_{CC}=2.4V$, $V_{OUT}=0V$	−0.1		mA
Source (Low Current Option)	$V_{CC}=4.5V$, $V_{OUT}=0V$	−30	−330	μA
	$V_{CC}=2.4V$, $V_{OUT}=0V$	−6	−80	μA
CKO Current Levels (As Clock Out)				
Sink ÷4		0.3		mA
÷8	$V_{CC}=4.5V$, CKI $=V_{CC}$, $V_{OUT}=V_{CC}$	0.6		mA
÷16		1.2		mA
Source ÷4		−0.3		mA
÷8	$V_{CC}=4.5V$, CKI = 0V, $V_{OUT}=0V$	−0.6		mA
÷16		−1.2		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	$V_{CC}=4.5V$, $V_{IN}=0.2V_{CC}$		0.7	mA
To Halt	$V_{CC}=4.5V$, $V_{IN}=0.7V_{CC}$		1.6	mA
TRI-STATE or Open Drain Leakage Current		−2.5	+2.5	μA

COP324C/COP325C/COP326C and COP344C/COP345C

Absolute Maximum Ratings

Supply Voltage	6V
Voltage at any Pin	−0.3V to V_{CC} + 0.3V
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics −40°C ≤ T_A ≤ +85°C unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		3.0	5.5	V
Power Supply Ripple (Note 5)	Peak to Peak		0.1 V_{CC}	V
Supply Current	V_{CC} = 3.0V, tc = 64 µs		180	µA
(Note 1)	V_{CC} = 5.0V, tc = 16 µs		800	µA
	V_{CC} = 5.0V, tc = 4 µs		3600	µA
	(tc is instruction cycle time)			
HALT Mode Current	V_{CC} = 5.0V, F_{IN} = 0 kHz		60	µA
(Note 2)	V_{CC} = 3.0V, F_{IN} = 0 kHz		30	µA
Input Voltage Levels				
$\overline{\text{RESET}}$, CKI, D_O (clock input)				
Logic High		0.9 V_{CC}		V
Logic Low			0.1 V_{CC}	V
All Other Inputs				
Logic High		0.7 V_{CC}		V
Logic Low			0.2 V_{CC}	V
Input Pull-Up Current	V_{CC} = 4.5V, V_{IN} = 0	30	440	µA
Hi-Z Input Leakage		−2	+2	µA
Input Capacitance (Note 4)			7	pF
Output Voltage Levels	Standard Outputs			
LSTTL Operation	V_{CC} = 5.0V ±10%			
Logic High	I_{OH} = −100 µA	2.7		V
Logic Low	I_{OL} = 400 µA		0.4	V
CMOS Operation				
Logic High	I_{OH} = −10 µA	V_{CC} − 0.2		V
Logic Low	I_{OL} = 10 µA		0.2	V
Output Current Levels (except CKO)				
Sink (Note 6)	V_{CC} = 4.5V, V_{OUT} = V_{CC}	1.2		mA
	V_{CC} = 3.0V, V_{OUT} = V_{CC}	0.2		mA
Source (Standard Option)	V_{CC} = 4.5V, V_{OUT} = 0V	−0.5		mA
	V_{CC} = 3.0V, V_{OUT} = 0V	−0.1		mA
Source (Low Current Option)	V_{CC} = 4.5V, V_{OUT} = 0V	−30	−440	µA
	V_{CC} = 3.0V, V_{OUT} = 0V	−8	−200	µA
CKO Current Levels (As Clock Out)				
Sink ÷4	V_{CC} = 4.5V, CKI = V_{CC}, V_{OUT} = V_{CC}	0.3		mA
÷8		0.6		mA
÷16		1.2		mA
Source ÷4	V_{CC} = 4.5V, CKI = 0V, V_{OUT} = 0V	−0.3		mA
÷8		−0.6		mA
÷16		−1.2		mA
Allowable Sink/Source Current per Pin (Note 6)			5	mA
Allowable Loading on CKO (as HALT)			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	V_{CC} = 4.5V, V_{IN} = 0.2V_{CC}		0.9	mA
To Halt	V_{CC} = 4.5V, V_{IN} = 0.7V_{CC}		2.1	mA
TRI-STATE or Open Drain Leakage Current		−5	+5	µA

COP424C/COP425C/COP426C and COP444C/COP445C

AC Electrical Characteristics $0°C \le T_A \le 70°C$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (tc)	$V_{CC} \ge 4.5V$	4	DC	μs
	$4.5V > V_{CC} \ge 2.4V$	16	DC	μs
Operating CKI Frequency ÷4 mode	$V_{CC} \ge 4.5V$	DC	1.0	MHz
÷8 mode		DC	2.0	MHz
÷16 mode		DC	4.0	MHz
÷4 mode	$4.5V > V_{CC} \ge 2.4V$	DC	250	kHz
÷8 mode		DC	500	kHz
÷16 mode		DC	1.0	MHz
Duty Cycle (Note 4)	$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 4$ MHz External Clock		60	ns
Fall Time (Note 4)	$f_1 = 4$ MHz External Clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)	$R = 30k \pm 5\%, V_{CC} = 5V$ $C = 82$ pF $\pm 5\%$ (÷4 Mode)	5	11	μs
Inputs: (See *Figure 3*)				
t_{SETUP}	G Inputs, $V_{CC} \ge 4.5V$	tc/4+.7		μs
	SI Input	0.3		μs
	All Others	1.7		μs
t_{HOLD}	$V_{CC} \ge 4.5V$	0.25		μs
	$4.5V > V_{CC} \ge 2.4V$	1.0		μs
Output Propagation Delay	$V_{OUT} = 1.5V, C_L = 100$ pF, $R_L = 5k$			
t_{PD1}, t_{PD0}	$V_{CC} \ge 4.5V$		1.0	μs
t_{PD1}, t_{PD0}	$4.5V > V_{CC} \ge 2.4V$		4.0	μs
Microbus Timing	$CL = 50$ pF, $V_{CC} = 5V \pm 5\%$			
Read Operation (*Figure 4*)				
Chip Select Stable before $\overline{RD}$ $-t_{CSR}$		65		ns
Chip Select Hold Time for $\overline{RD}$ $-t_{RCS}$		20		ns
$\overline{RD}$ Pulse Width $-t_{RR}$		400		ns
Data Delay from $\overline{RD}$ $-t_{RD}$			375	ns
$\overline{RD}$ to Data Floating $-t_{DF}$ (Note 4)			250	ns
Write Operation (*Figure 5*)				
Chip Select Stable before $\overline{WR}$ $-t_{CSW}$		65		ns
Chip Select Hold Time for $\overline{WR}$ $-t_{WCS}$		20		ns
$\overline{WR}$ Pulse Width $-t_{WW}$		400		ns
Data Set-Up Time for $\overline{WR}$ $-t_{DW}$		320		ns
Data Hold Time for $\overline{WR}$ $-t_{WD}$		100		ns
INTR Transition Time from $\overline{WR}$ $-t_{WI}$			700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 17.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}, L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than $0.2V_{CC}$ when part is running in order to prevent entering test mode.

COP324C/COP325C/COP326C and COP344C/COP345C

AC Electrical Characteristics $-40°C \leq T_A \leq +85°C$ unless otherwise specified.

Parameter		Conditions	Min	Max	Units
Instruction Cycle Time (tc)		$V_{CC} \geq 4.5V$	4	DC	μs
		$4.5V > V_{CC} \geq 3.0V$	16	DC	μs
Operating CKI Frequency	÷4 mode	$V_{CC} \geq 4.5V$	DC	1.0	MHz
	÷8 mode		DC	2.0	MHz
	÷16 mode		DC	4.0	MHz
	÷4 mode	$4.5V > V_{CC} \geq 3.0V$	DC	250	kHz
	÷8 mode		DC	500	kHz
	÷16 mode		DC	1.0	MHz
Duty Cycle (Note 4)		$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)		$f_1 = 4$ MHz external clock		60	ns
Fall Time (Note 4)		$f_1 = 4$ MHz external clock		40	ns
Instruction Cycle Time RC Oscillator (Note 4)		R = 30k ±5%, V_{CC} = 5V C = 82 pF ±5% (÷4 Mode)	5	11	μs
Inputs: (See *Figure 3*)					
t_{SETUP}		G Inputs, $V_{CC} \geq 4.5V$	tc/4+.7		μs
		SI Inputs	0.3		μs
		All Others	1.7		μs
t_{HOLD}		$V_{CC} \geq 4.5V$	0.25		μs
		$4.5V > V_{CC} \geq 3.0V$	1.0		μs
Output Propagation Delay		$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5k$			
t_{PD1}, t_{PD0}		$V_{CC} \geq 4.5V$		1.0	μs
t_{PD1}, t_{PD0}		$4.5V > V_{CC} \geq 3.0V$		4.0	μs
Microbus Timing		$C_L = 50$ pF, $V_{CC} = 5V \pm 5\%$			
Read Operation (*Figure 4*)					
Chip Select Stable before $\overline{RD}$ $-t_{CSR}$			65		ns
Chip Select Hold Time for $\overline{RD}$ $-t_{RCS}$			20		ns
$\overline{RD}$ Pulse Width $-t_{RR}$			400		ns
Data Delay from $\overline{RD}$ $-t_{RD}$				375	ns
$\overline{RD}$ to Data Floating $-t_{DF}$ (Note 4)				250	ns
Write Operation (*Figure 5*)					
Chip Select Stable before $\overline{WR}$ $-t_{CSW}$			65		ns
Chip Select Hold Time for $\overline{WR}$ $-t_{WCS}$			20		ns
$\overline{WR}$ Pulse Width $-t_{WW}$			400		ns
Data Set-Up Time for $\overline{WR}$ $-t_{DW}$			320		ns
Data Hold Time for $\overline{WR}$ $-t_{WD}$			100		ns
INTR Transition Time from $\overline{WR}$ $-t_{WI}$				700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI, CKO open, and all other pins pulled up to V_{CC} with 5k resistors. See current drain equation on page 17.

Note 2: The HALT mode will stop CKI from oscillating in the RC and crystal configurations. Test conditions: all inputs tied to V_{CC}, L lines in TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

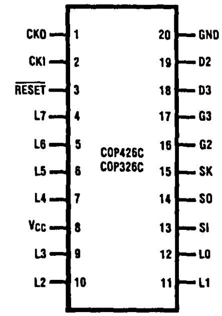
Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than $0.2V_{CC}$ when part is running in order to prevent entering test mode.

Connection Diagrams

DIP and S.O. Wide



TL/DD/5259-16

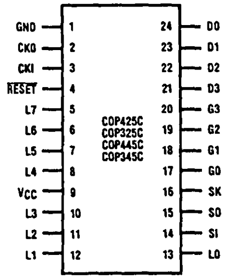
Top View

Order Number COP326C-XXX/D
or COP426C-XXX/D
See NS Hermetic Package D20A

Order Number COP326C-XXX/N
or COP426C-XXX/N
See NS Molded Package N20A

Order Number COP326C-XXX/WM
or COP426C-XXX/WM
See NS Surface Mount Package M20B

DIP and S.O. Wide



TL/DD/5259-2

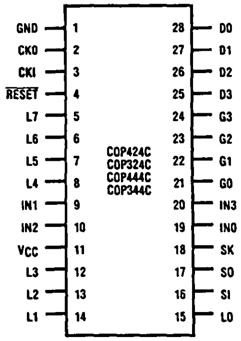
Top View

Order Number COP325C-XXX/D
or COP425C-XXX/D
See NS Hermetic Package D24C

Order Number COP325C-XXX/N
or COP425C-XXX/N
See NS Molded Package N24A

Order Number COP325C-XXX/WM
or COP425C-XXX/WM
See NS Surface Mount Package M24B

Dual-In-Line Package



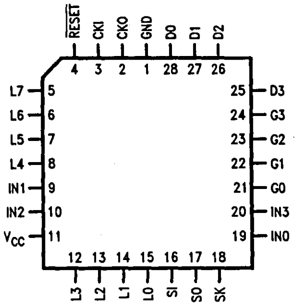
TL/DD/5259-3

Top View

Order Number COP324C-XXX/D
or COP424C-XXX/D
See NS Hermetic Package D28C

Order Number COP324C-XXX/N
or COP424C-XXX/N
See NS Molded Package N28B

28 Lead PLCC



TL/DD/5259-18

Order Number COP324C-XXX/V
or COP424C-XXX/V
See NS PLCC Package V28A

FIGURE 2

Pin	Description
L7–L0	8-bit bidirectional port with TRI-STATE
G3–G0	4-bit bidirectional I/O port
D3–D0	4-bit output port
IN3–IN0	4-bit input port (28-pin package only)
SI	Serial input or counter input
SO	Serial or general purpose output

Pin	Description
SK	Logic controlled clock output
CKI	Chip oscillator input
CKO	Oscillator output, HALT I/O port or general purpose input
RESET	Reset input
V_{CC}	Most positive power supply
GND	Ground

Functional Description

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

For ease of reading only the COP424C/425C/COP426C/444C/445C are referenced; however, all such references apply equally to COP324C/325C/COP326C/344C/345C.

PROGRAM MEMORY

Program Memory consists of ROM, 1024 bytes for the COP424C/425C/426C and 2048 bytes for the COP444C/445C. These bytes of ROM may be program instructions, constants or ROM addressing data.

ROM addressing is accomplished by a 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

Data memory consists of a 512-bit RAM for the COP444C/445C, organized as 8 data registers of 16 × 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register.

Data memory consists of a 256-bit RAM for the COP424C/425C/426C, organized as 4 data registers of 16 × 4-bits digits. The B register is 6 bits long. Upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions.

The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, to input 4 bits of L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes depending on a mask-programmable option: as a timer or as an external event counter. When the T counter overflows, an overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 10a*.

Four general-purpose inputs, IN3-IN0, are provided. IN1, IN2 and IN3 may be selected, by a mask-programmable option as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in Microbus application.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. In the dual clock mode, D0 latch controls the clock selection (see dual oscillator below).

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G0 may be mask-programmed as an output for Microbus applications.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the Microbus option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M. As explained above, the Microbus option allows L I/O port data to be latched into the Q register.

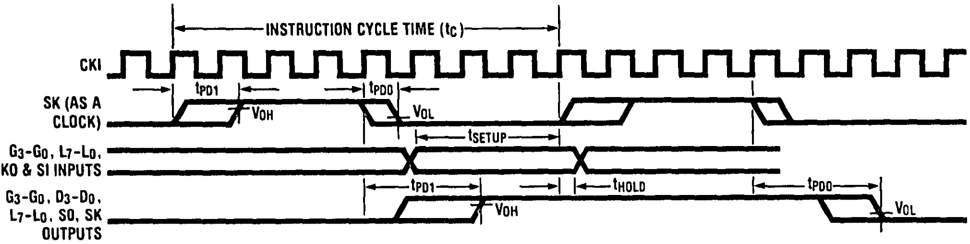
Functional Description (Continued)

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

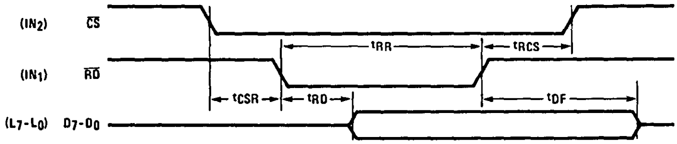
EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.


INSTRUCTION CYCLE TIME (tc)
CKI
SK (AS A CLOCK)
tPD1
tPD0
VOH
VOL
tSETUP
G3-G0, L7-L0, CKO & SI INPUTS
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
tPD1
tHOLD
tPD0
VOH
VOL

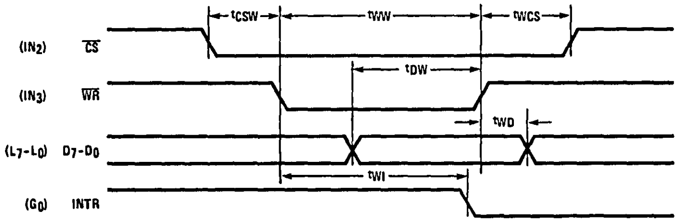
TL/DD/5259-4

FIGURE 3. Input/Output Timing Diagrams (divide by 8 mode)



TL/DD/5259-5

FIGURE 4. Microbus Read Operation Timing



TL/DD/5259-6

FIGURE 5. Microbus Write Operation Timing

Functional Description (Continued)

3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.

b. An interrupt will be recognized only on the following conditions:
   1. EN1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN_1 input.
   3. A currently executing instruction has been completed.
   4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

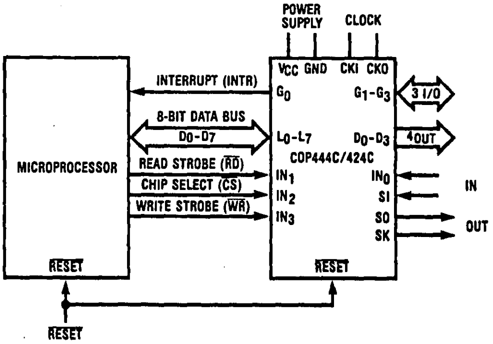
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

MICROBUS INTERFACE

The COP444C/424C has an option which allows it to be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN1, IN2 and IN3 general purpose inputs become Microbus compatible read-strobe, chip-select, and write-strobe lines, respectively. IN1 becomes $\overline{RD}$ — a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the uP. IN2 becomes $\overline{CS}$ — a logic "0" on this line selects the COP444C/424C as the uP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN3 becomes $\overline{WR}$ — a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP444C/424C. G0 becomes INTR a "ready" output, reset by a write pulse from the uP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP444C/424C.

This option has been designed for compatibility with National's Microbus — a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See Microbus National Publication.) The functioning and timing relationships between the signal lines affected by this option are as specified for the Microbus interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection of the COP444C/424C to the Microbus is shown in *Figure 6*.



TL/DD/5259-7

FIGURE 6. Microbus Option Interconnect

TABLE I. Enable Register Modes — Bits EN0 and EN3

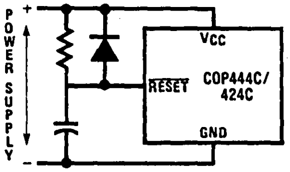
EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1,SK=clock If SKL=0,SK=0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL=1,SK=clock If SKL=0,SK=0
1	0	Binary Counter	Input to Counter	0	SK=SKL
1	1	Binary Counter	Input to Counter	1	SK=SKL

Functional Description (Continued)

INITIALIZATION

The internal reset logic will initialize the device upon power-up if the power supply rise time is less than 1 ms and if the operating frequency at CKI is greater than 32 kHz, otherwise the external RC network shown in *Figure 7* must be connected to the $\overline{\text{RESET}}$ pin (the conditions in *Figure 7* must be met). The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

Note: If CKI clock is less than 32 kHz, the internal reset logic (option #29 = 1) MUST be disabled and the external RC circuit must be used.



RC>5X POWER SUPPLY RISE TIME
AND RC>100X CKI PERIOD.

TL/DD/5259-8

FIGURE 7. Power-Up Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).

TIMER

There are two modes selected by mask option:

a. Time-base counter. In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset.

   For example, using a 4 MHz crystal with a divide-by-16 option, the instruction cycle frequency of 250 kHz increments the 10-bit timer every 4 μs. By presetting the counter and detecting overflow, accurate timeouts between 16 μs (4 counts) and 4.096 ms (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

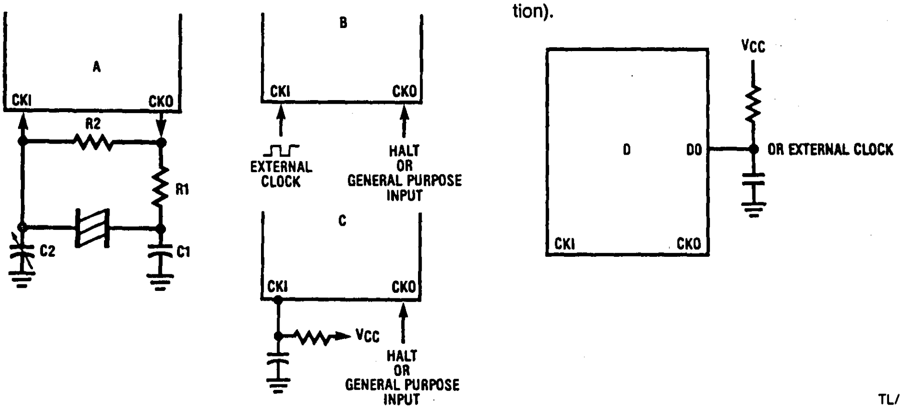
b. External event counter. In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

Note: The IT instruction is not allowed in this mode.

HALT MODE

The COP444C/445C/424C/425C/426C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by the HALT instruction or by forcing CKO high when it is mask-programmed as an HALT I/O port. Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing. The chip may be awakened by one of two different methods:

- Continue function: by forcing CKO low, if it mask-programmed as an HALT I/O port, the system clock is re-enabled and the circuit continues to operate from the point where it was stopped.
- Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization).



TL/DD/5259-9

Crystal or Resonator

Crystal Value	Component Values			
	R1	R2	C1(pF)	C2(pF)
32 kHz	220k	20M	30	6–36
455 kHz	5k	10M	80	40
2.096 MHz	2k	1M	30	6–36
4.0 MHz	1k	1M	30	6–36

RC Controlled Oscillator (±5% R, ±5% C)

R	C	Cycle Time	V_{CC}
30k	82 pF	5–11 μs	≥4.5V
60k	100 pF	12–24 μs	2.4–4.5V

Note: 15k≤R≤150k
50 pF≤C≤150 pF

FIGURE 8. Oscillator Component Values

Functional Description (Continued)

The HALT mode is the minimum power dissipation state.

Note: If the user has selected dual-clock with D0 as external oscillator (option 30 = 2) AND the COP444C/424C is running with the D0 clock, the HALT mode — either hardware or software — will NOT be entered. Thus, the user should switch to the CKI clock to HALT. Alternatively, the user may stop the D0 clock to minimize power.

CKO PIN OPTIONS

a. Two-pin oscillator — (Crystal). See *Figure 9A*.

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. The HALT mode may be entered by program control (HALT instruction) which forces CKO high, thus inhibiting the crystal network. The circuit can be awakened only by forcing the $\overline{\text{RESET}}$ pin to a logic "0" (restart).

b. One-pin oscillator — (RC or external). See *Figure 9B*.

If a one-pin oscillator system is chosen, two options are available for CKO:

- CKO can be selected as the HALT I/O port. In that case, it is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing a high level to CKO, the chip will stop as soon as CKI is high and CKO output will stay high to keep the chip stopped if the external driver returns to high impedance state. By forcing a low level to CKO, the chip will continue and CKO will stay low.
- As another option, CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction.

OSCILLATOR OPTIONS

There are four basic clock oscillator configurations available as shown by *Figure 8*.

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency optionally divided by 4, 8 or 16.

b. External Oscillator. The external frequency is optionally divided by 4, 8 or 16 to give the instruction cycle time. CKO is the HALT I/O port or a general purpose input.

c. RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is the HALT I/O port or a general purpose input.

d. Dual oscillator. By selecting the dual clock option, pin D0 is now a single pin oscillator input. Two configurations are available: RC controlled Schmitt trigger oscillator or external oscillator.

The user may software select between the D0 oscillator (in that case, the instruction cycle time equals the D0 oscillation frequency divided by 4) by setting the D0 latch high or the CKI (CKO) oscillator by resetting D0 latch low. Note that even in dual clock mode, the counter, if mask-programmed as a time-base counter, is always connected to the CKI oscillator.

For example, the user may connect up to a 1 MHz RC circuit to D0 for faster processing and a 32 kHz watch crystal to CKI and CKO for minimum current drain and time keeping.

Note: CTMA instruction is not allowed when chip is running from D0 clock.

Figures 10A and *10B* show the clock and timer diagrams with and without Dual clock.

COP445C AND COP425C 24-PIN PACKAGE OPTION

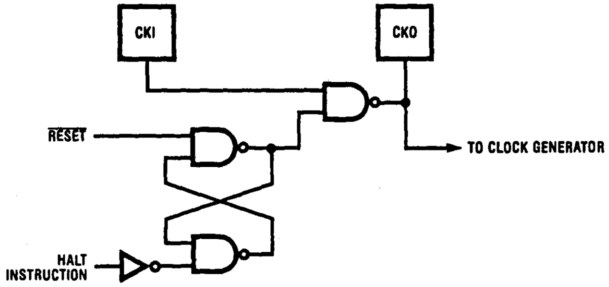
If the COP444C/424C is bonded in a 24-pin package, it becomes the COP445C/425C, illustrated in *Figure 2*, Connection diagrams. Note that the COP445C/425C does not contain the four general purpose IN inputs (IN3–IN0). Use of this option precludes, of course, use of the IN options, interrupt feature, external event counter feature, and the Microbus option which uses IN1–IN3. All other options are available for the COP445C/425C.

Note: If user selects the 24-pin package, options 9, 10, 19 and 20 must be selected as a "0" (load to V_{CC} on the IN inputs). See option list.

COP426C 20-PIN PACKAGE OPTION

If the COP425C is bonded as 20-pin device it becomes the COP426C. Note that the COP426C contains all the COP425C pins except D_0, D_1, G_0, and G_1.

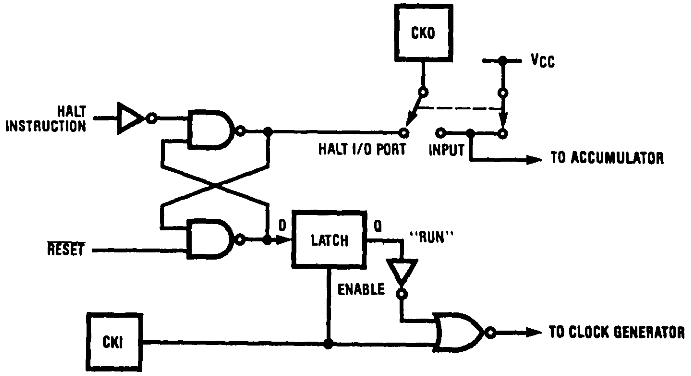
Block Diagram (Continued)



TL/DD/5259-10

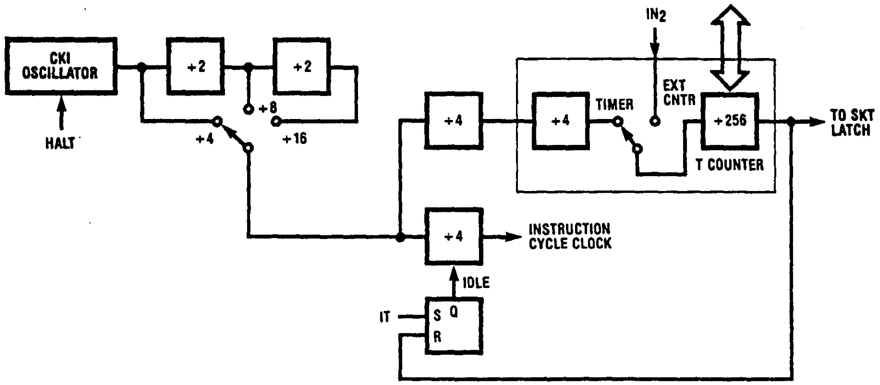
FIGURE 9A. Halt Mode — Two-Pin Oscillator

Block Diagram (Continued)



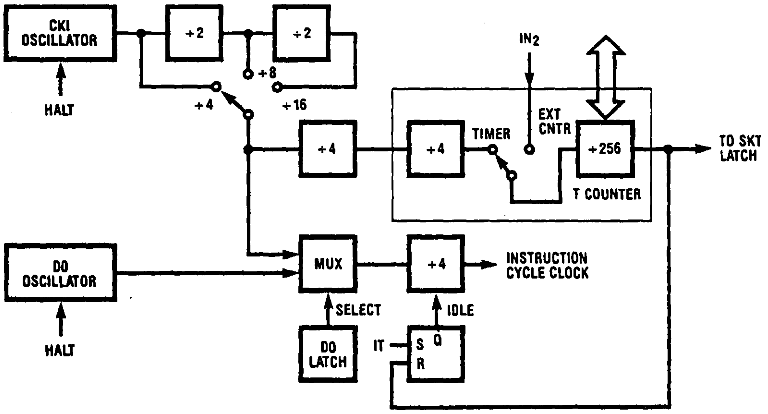
TL/DD/5259-11

FIGURE 9B. Halt Mode — One-Pin Oscillator



TL/DD/5259-12

FIGURE 10A. Clock and Timer without Dual-Clock



TL/DD/5259-13

FIGURE 10B. Clock and Timer with Dual-Clock

Instruction Set

Table II is a symbol table providing internal architecture, instruction operan and operation symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

TABLE II. Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit accumulator
B	7-bit RAM address register (6-bit for COP424C)
Br	Upper 3 bits of B (register address) (2-bit for COP424C)
Bd	Lower 4 bits of B (digit address)
C	1-bit carry register
D	4-bit data output port
EN	4-bit enable register
G	4-bit general purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA,SB,SC	11-bit 3-level subroutine stack
SIO	4-bit shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer

Instruction Operand Symbols	
d	4-bit operand field, 0–15 binary (RAM digit select)
r	3(2)-bit operand field, 0–7(3) binary (RAM register select)
a	11-bit operand field, 0–2047 (1023)
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x

Operational Symbols	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	One's complement of A
⊕	Exclusive-or
:	Range of values

TABLE III. COP444C/445C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A+RAM(B) → A	None	Add RAM to A
ADT		4A	0100 1010	$A+10_{10} \rightarrow A$	None	Add Ten to A
AISC	y	5–	0101 y	A+y → A	Carry	Add Immediate. Skip on Carry (y ≠ 0)
CASC		10	0001 0000	$\overline{A}$+RAM(B)+C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A⊕RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{10:8}$ A,M) → $PC_{7:0}$	None	Jump Indirect (Notes 1, 3)
JMP	a	6– – –	0110 0 $a_{10:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	– – – –	1 $a_{6:0}$ (pages 2,3 only) or 11 $a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 4)
JSRP	a	– –	10 $a_{5:0}$	PC+1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	0110 1 $a_{10:8}$ $a_{7:0}$	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	0011 0011 0011 1000		None	HALT Processor
IT		33 39	0011 0011 0011 1001		None	IDLE till Timer Overflows then Continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33 3F	0011 0011 0011 1111	A → $T_{7:4}$ RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CTMA		33 2F	0011 0011 0010 1111	$T_{7:4}$ → RAM(B) $T_{3:0}$ → A	None	Copy T to RAM, A (Note 9)
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101 (r=0:3)	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 – –	0010 0011 0 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
STII	y	7−	0111 | y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate 1 and Increment Bd
X	r	−6	00 | r | 0110 (r=0:3)	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 −−	0010 | 0011 1 | r | d	RAM(r,d) ←→ A	None	Exchange A with RAM Pointed to Directly by r,d
XDS	r	−7	00 | r | 0111 (r=0:3)	RAM(B) ←→ A Bd−1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd. Exclusive-OR Br with r
XIS	r	−4	00 | r | 0100 (r=0:3)	RAM(B) ←→ A Bd+1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 | 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 | 1110	Bd → A	None	Copy Bd to A
LBI	r,d	−−	00 | r | (d−1) (r=0:3: d=0,9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33 −−	0011 | 0011 1 | r | d (any r, any d)			
LEI	y	33 6−	0011 | 0011 0110 | y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 | 0010	A ←→ Br	None	Exchange A with Br (Note 8)
TEST INSTRUCTIONS						
SKC		20	0010 | 0000		C="1"	Skip if C is True
SKE		21	0010 | 0001		A=RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 | 0011 0010 | 0001		$G_{3:0}=0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 | 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 | 0001	2nd byte	$G_0=0$	
	1	11	0001 | 0001		$G_1=0$	
	2	03	0000 | 0011		$G_2=0$	
	3	13	0001 | 0011		$G_3=0$	
SKMBZ	0	01	0000 | 0001		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	0001 | 0001		$RAM(B)_1=0$	
	2	03	0000 | 0011		$RAM(B)_2=0$	
	3	13	0001 | 0011		$RAM(B)_3=0$	
SKT		41	0100 | 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

Table III. COP444C/445C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN → A	None	Input IN Inputs to A (Note 2)
INIL		33 29	0011 0011 0010 1001	IL_3, CKO,"0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM,A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5−	0011 0011 0101 y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin packages since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B(Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: For 2K ROM devices, A ←→ Br (0 → A3). For 1K ROM devices, A ←→ Br (0,0 → A3, A2).

Note 9: Do not use CTMA instruction when dual-clock option is selected and part is running from D_0 clocks.

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10:PC8,A,M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC+1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC(7:4), RAM(B) → PC(3:0), leaving PC(10), PC(9) and PC(8) unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10:8,A,M. PC10,PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT   ; load T counter
SKT    ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is mask-programmed as an external event counter (option #31 = 1).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKO and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A0 is input into A1. IL latches are cleared on reset. IL latches are not available on the COP445C/425C, and COP426C.

INSTRUCTION SET NOTES

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.

c. The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Note: The COP424C/425C/426C needs only 10 bits to address its ROM. Therefore, the eleventh bit (P10) is ignored.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. A crystal or resonator generated clock input will draw additional current. An R/C oscillator will draw even more current since the input is a slow rising signal.

If using an external squarewave oscillator, the following equation can be used to calculate operating current drain.

$$I_{CO} = I_Q + V \times 40 \times Fi + V \times 1400 \times Fi/Dv$$

where I_{CO} = chip operating current drain in microamps

quiescent leakage current (from curve)

CKI frequency in MegaHertz

chip V_{CC} in volts

divide by option selected

For example at 5 volts V_{CC} and 400 kHz (divide by 4)

$$I_{CO} = 20 + 5 \times 40 \times 0.4 + 5 \times 1400 \times 0.4/4$$

$$I_{CO} = 20 + 80 + 700 = 800\ \mu A$$

At 2.4 volts V_{CC} and 30 kHz (divide by 4)

$$I_{CO} = 6 + 2.4 \times 40 \times 0.03 + 2.4 \times 1400 \times 0.03/4$$

$$I_{CO} = 6 + 2.88 + 25.2 = 34.08\ \mu A$$

Power Dissipation (Continued)

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$Ici = I_Q + V \times 40 \times Fi$$

For example, at 5 volts V_{CC} and 400 kHz

$$Ici = 20 + 5 \times 40 \times 0.4 = 100\ \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$Ita = I_{CO} \times \frac{To}{To+Ti} + Ici \times \frac{Ti}{To+Ti}$$

where: Ita = total average current
I_{CO} = operating current
Ici = idle current
To = operating time
Ti = idle time

I/O OPTIONS

Outputs have the following optional configurations, illustrated in *Figure 11*:

a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL.

b. Low Current — This is the same configuration as a. above except that the sourcing current is much less.

c. Open Drain — An N-channel device to ground only, allowing external pull-up as required by the user's application.

d. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control.

e. Low-Current TRI-STATE L Output — This is the same as d. above except that the sourcing current is much less.

f. Open-Drain TRI-STATE L Output — This has the N-channel device to ground only.

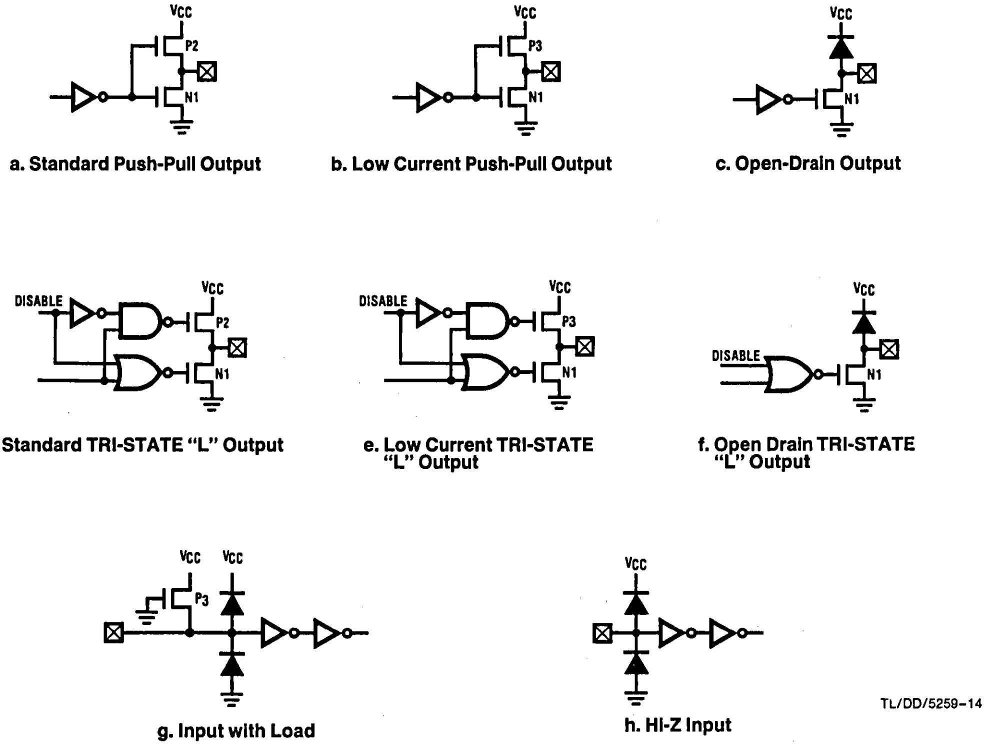
All inputs have the following options:

g. Input with on chip load device to V_{CC}.

h. Hi-Z input which must be driven by the users logic.

When using either the G or L I/O ports as inputs, a pull-up device is necessary. This can be an external device or the following alternative is available: Select the low-current output option. Now, by setting the output registers to a logic "1" level, the P-channel devices will act as the pull-up load. Note that when using the L ports in this fashion the Q registers must be set to a logic "1" level and the L drivers MUST BE ENABLED by an LEI instruction (see description above).

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 12* for each of these devices to allow the designer to effectively use these I/O configurations.



a. Standard Push-Pull Output

b. Low Current Push-Pull Output

c. Open-Drain Output

d. Standard TRI-STATE "L" Output

e. Low Current TRI-STATE "L" Output

f. Open Drain TRI-STATE "L" Output

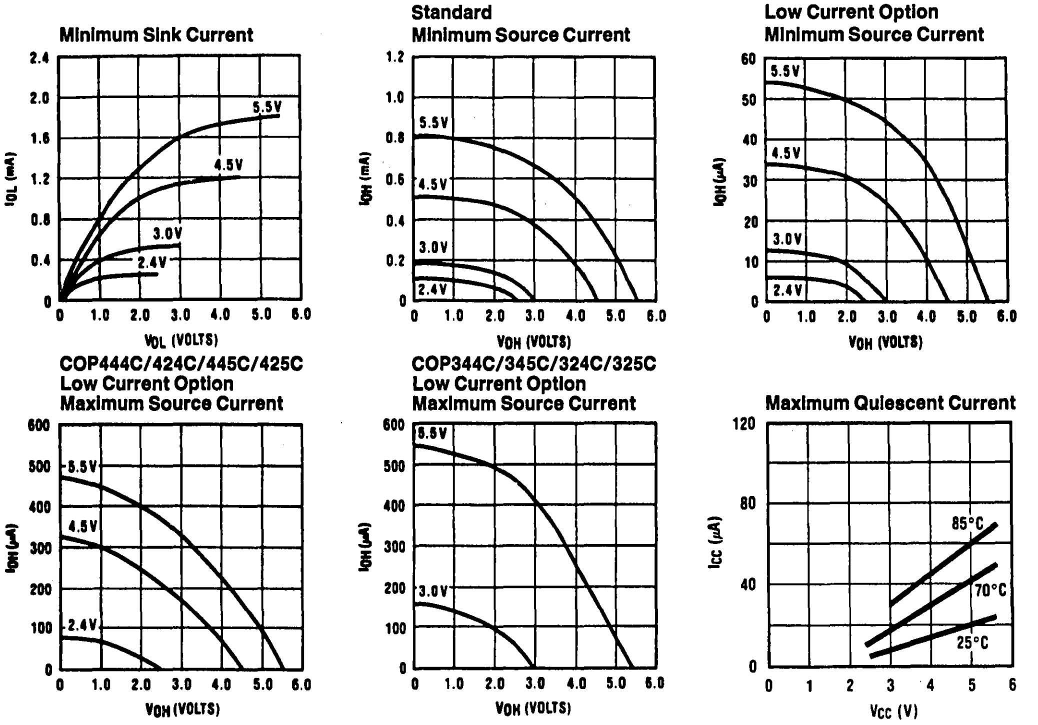
g. Input with Load

h. Hi-Z Input

TL/DD/5259–14

FIGURE 11. Input/Output Configurations

Power Dissipation (Continued)



TL/DD/5259-15

FIGURE 12. Input/Output Characteristics

Option List

The COP444C/445C/424C/425C/COP426C mask-programmable options are assigned numbers which correspond with the COP444C/424C pins.

The following is a list of options. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

PLEASE FILL OUT THE OPTION TABLE on the next page. Xerox the option data and send it in with your disk or EPROM.

Option 1 = 0: Ground Pin — no options available

Option 2: CKO Pin
- = 0: clock generator output to crystal/resonator
- = 1: HALT I/O port
- = 2: general purpose input with load device to V_{CC}
- = 3: general purpose input, high-Z

Option 3: CKI input
- = 0: Crystal controlled oscillator input divide by 4
- = 1: Crystal controlled oscillator input divide by 8
- = 2: Crystal controlled oscillator input divide by 16
- = 4: Single-pin RC controlled oscillator (divide by 4)
- = 5: External oscillator input divide by 4
- = 6: External oscillator input divide by 8
- = 7: External oscillator input divide by 16

Option 4: $\overline{\text{RESET}}$ input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 5: L7 Driver
- = 0: Standard TRI-STATE push-pull output
- = 1: Low-current TRI-STATE push-pull output
- = 2: Open-drain TRI-STATE output

Option 6: L6 Driver — (same as option 5)

Option 7: L5 Driver — (same as option 5)

Option 8: L4 Driver — (same as option 5)

Option 9: IN1 input
- = 0: load device to V_{CC}
- = 1: Hi-Z input

Option 10: IN2 input — (same as option 9)

Option 11 = 0: V_{CC} Pin — no option available

Option 12: L3 Driver — (same as option 5)

Option 13: L2 Driver — (same as option 5)

Option 14: L1 Driver — (same as option 5)

Option 15: L0 Driver — (same as option 5)

Option 16: SI input — (same as option 9)

Option 17: SO Driver
- = 0: Standard push-pull output
- = 1: Low-current push-pull output
- = 2: Open-drain output

Option List (Continued)

Option 18: SK Driver — (same as option 17)
Option 19: IN0 Input — (same as option 9)
Option 20: IN3 Input — (same as option 9)
Option 21: G0 I/O Port — (same as option 17)
Option 22: G1 I/O Port — (same as option 17)
Option 23: G2 I/O Port — (same as option 17)
Option 24: G3 I/O Port — (same as option 17)
Option 25: D3 Output — (same as option 17)
Option 26: D2 Output — (same as option 17)
Option 27: D1 Output — (same as option 17)
Option 28: D0 Output — (same as option 17)
Option 29: Internal Initialization Logic
=0: Normal operation
=1: No internal initialization logic
Option 30: Dual Clock
=0: Normal operation
=1: Dual Clock. D0 RC oscillator
=2: Dual Clock. D0 ext. clock input } (opt. #28 must=2)
Option 31: Timer
=0: Time-base counter
=1: External event counter

Option 32: Microbus
=0: Normal
=1: Microbus (opt. #31 must=0)
Option 33: COP bonding
(1k and 2K Microcontroller)
=0: 28-pin package
=1: 24-pin package
=2: Same die purchased in both
24 and 28 pin version.
(1K Microcontroller only)
=3: 20-pin package
=4: 28- and 20-pin package
=5: 24- and 20-pin package
=6: 28-, 24- and 20-pin package

Note:—if opt. #33=1 or 2 then opt.#9, 10, 19, 20 and 32 must = 0—if opt. #33=3, 4, 5 or 6 then opt. #9, 10, 19, 20, 21, 22, 30 and 32 must = 0.

Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE = ____0____ IS: GROUND PIN
OPTION 2 VALUE = _________ IS: CKO PIN
OPTION 3 VALUE = _________ IS: CKI INPUT
OPTION 4 VALUE = _________ IS: RESET INPUT
OPTION 5 VALUE = _________ IS: L(7) DRIVER
OPTION 6 VALUE = _________ IS: L(6) DRIVER
OPTION 7 VALUE = _________ IS: L(5) DRIVER
OPTION 8 VALUE = _________ IS: L(4) DRIVER
OPTION 9 VALUE = _________ IS: IN1 INPUT
OPTION 10 VALUE = _________ IS: IN2 INPUT
OPTION 11 VALUE = _________ IS: VCC PIN
OPTION 12 VALUE = _________ IS: L(3) DRIVER
OPTION 13 VALUE = _________ IS: L(2) DRIVER
OPTION 14 VALUE = _________ IS: L(1) DRIVER
OPTION 15 VALUE = _________ IS: L(0) DRIVER
OPTION 16 VALUE = _________ IS: SI INPUT

OPTION DATA

OPTION 17 VALUE = _________ IS: SO DRIVER
OPTION 18 VALUE = _________ IS: SK DRIVER
OPTION 19 VALUE = _________ IS: IN0 INPUT
OPTION 20 VALUE = _________ IS: IN3 INPUT
OPTION 21 VALUE = _________ IS: G0 I/O PORT
OPTION 22 VALUE = _________ IS: G1 I/O PORT
OPTION 23 VALUE = _________ IS: G2 I/O PORT
OPTION 24 VALUE = _________ IS: G3 I/O PORT
OPTION 25 VALUE = _________ IS: D3 OUTPUT
OPTION 26 VALUE = _________ IS: D2 OUTPUT
OPTION 27 VALUE = _________ IS: D1 OUTPUT
OPTION 28 VALUE = _________ IS: D0 OUTPUT
OPTION 29 VALUE = _________ IS: INT INIT LOGIC
OPTION 30 VALUE = _________ IS: DUAL CLOCK
OPTION 31 VALUE = _________ IS: TIMER
OPTION 32 VALUE = _________ IS: MICROBUS
OPTION 33 VALUE = _________ IS: COP BONDING

National Semiconductor

COP440/COP441/COP442 and COP340/COP341/COP342 Single-Chip N-Channel Microcontrollers

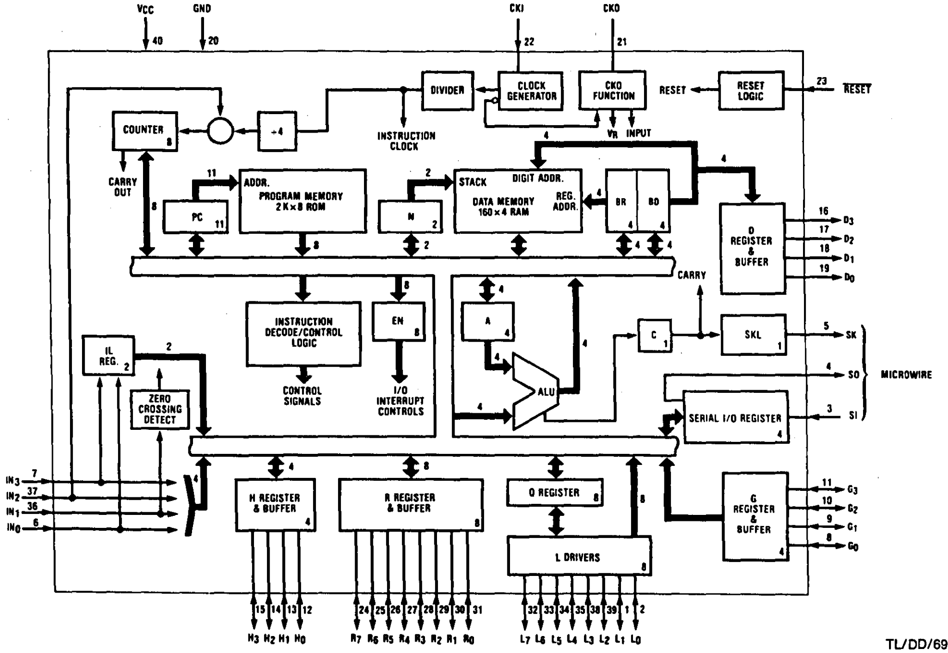
General Description

The COP440, COP441, COP442, COP340, COP341, and COP342 Single-Chip N-Channel Microcontrollers are members of the COPS™ microcontrollers family, fabricated using N-channel, silicon gate MOS technology. These are complete microcontrollers with all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, various output configuration options, and an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output, and data manipulation. The COP440 is a 40-pin chip and the COP441 is a 28-pin version of the same circuit (12 I/O lines removed). The COP442 is a 24-pin version (4 more input lines removed). The COP340, COP341, COP342 are functional equivalents of the above devices respectively, but operate with an extended temperature range (−40°C to +85°C). Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

Features

- Enhanced, more powerful instruction set
- 2k x 8 ROM, 160 x 4 RAM
- 35 I/O lines (COP440)
- Zero-crossing detect circuitry with hysteresis
- True multi-vectored interrupt from 4 selectable sources (plus restart)
- Four-level subroutine stack (in RAM)
- 4 µs cycle time
- Single supply operation (4.5V–6.3V)
- Programmable time-base counter
- Internal binary counter/register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- LED drive capability
- MICROBUS™ compatible
- Software/hardware compatible with other members of the COP400 family
- Extended temperature range devices COP340, COP341, COP342 (−40°C to +85°C)
- Compatible dual CPU device available (COP2440 series)

Block Diagram



TL/DD/6926-1

FIGURE 1

COP440/COP441/COP442
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Zero-Crossing Detect Pin Relative to GND	−1.2V to +15V
Voltage at Any Other Pin Relative to GND	−0.5V to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	150 mA
Total Sink Current	75 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0^\circ C \le T_A \le +70^\circ C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 3)	4.5	6.3	V
Power Supply Ripple	(Peak to Peak)		0.4	V
Operating Supply Current	(All Inputs and Outputs Open)			
	$T_A = 0^\circ C$		44	mA
	$T_A = 25^\circ C$		35	mA
	$T_A = 70^\circ C$		27	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷16, ÷8)				
Logic High (V_{IH})	V_{CC} = Max	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
Zero-Crossing Detect Input	See *Figure 7*			
Trip Point		−0.15	0.15	V
Logic High (V_{IH}) Limit			12	V
Logic Low (V_{IL}) Limit		−0.8		V
SO Input Level (Test Mode)	(Note 5)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Capacitance			7.0	pF
Hi-Z Input Leakage		−1.0	+1.0	μA

Note 1: Duty Cycle = $t_{WI}/(t_{WI} + t_{WO})$.

Note 2: See Figure for additional I/O Characteristics.

Note 3: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 4: Exercise great care not to exceed maximum device power dissipation limits when direct-driving LEDs (or sourcing similar loads) at high temperature.

Note 5: SO output "0" level must be less than 0.8V for normal operation.

COP440/COP441/COP442

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Voltage Levels				
Standard Output				
TTL Operation				
Logic High (V_{OH})	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low (V_{OL})	I_{OL} = 1.6 mA		0.4	V
CMOS Operation (Note 1)				
Logic High (V_{OH})	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.4$		V
Logic Low (V_{OL})	$I_{OL} = 10\ \mu A$		0.2	V
Output Current Levels				
Standard Output Source Current	$V_{CC} = 4.5V$, $V_{OH} = 2.4V$	−100	−650	μA
LED Direct Drive Output	$V_{CC} = 6V$, $V_{OH} = 2V$			
Logic High (I_{OH})		−2.5	−17	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
CKO Output				
Oscillator Output Option				
Logic High	$V_{OH} = 2V$	−0.2		mA
Logic Low	$V_{OL} = 0.4V$	0.4		mA
V_R RAM Power Supply Option				
Supply Current	$V_R = 3.3V$		3.0	mA
CKI Sink Current (RC Option)	$V_{IH} = 3.5V$, $V_{CC} = 4.5V$	2.0		mA
Input Current Levels				
Zero-Crossing Detect Input				
Resistance	$V_{IH} = 1.0V$	0.9	4.6	kΩ
Input Load Source Current	$V_{IH} = 2.0V$, $V_{CC} = 4.5V$	14	230	μA
Total Sink Current Allowed				
All I/O Combined			75	mA
Each L, R Port			20	mA
Each D, G, H Port			10	mA
SO, SK			2.5	mA
Total Source Current Allowed				
All I/O Combined			150	mA
L Port			120	mA
L_7–L_4			70	mA
L_3–L_0			70	mA
Each L Pin			23	mA
All Other Output Pins			1.6	mA

Note 1: TRI-STATE and LED configurations are excluded.

COP340/COP341/COP342 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Zero-Crossing Detect Pin Relative to GND	−1.2V to +15V
Voltage at Any Other Pin Relative to GND	−0.5V to +7V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.25W at 85°C
Total Source Current	150 mA
Total Sink Current	75 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 3)	4.5	5.5	V
Power Supply Ripple	(Peak to Peak)		0.4	V
Operating Supply Current	(All Inputs and Outputs Open)			
	$T_A = -40°C$		54	mA
	$T_A = 25°C$		35	mA
	$T_A = 85°C$		25	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷16, ÷8)	V_{CC} = Max	3.0		V
Logic High (V_{IH})		2.2		V
Logic Low (V_{IL})		−0.3	0.3	V
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
Zero-Crossing Detect Input	See *Figure 7*			
Trip Point		−0.15	0.15	V
Logic High (V_{IH}) Limit			12	V
Logic Low (V_{IL}) Limit		−0.8		V
SO Input Level (Test Mode)	(Note 5)	2.2	2.4	V
All Other Inputs	V_{CC} = Max	3.0		V
Logic High		2.2		V
Logic Low		−0.3	0.6	V
Input Levels High Trip Option				
Logic High		3.6		V
Logic Low		−0.3	1.2	V
Input Capacitance			7.0	pF
Hi-Z Input Leakage		−2.0	+2.0	µA

Note 1: Duty Cycle = $t_{WI}/(t_{WI} + t_{WO})$.

Note 2: See Figure for additional I/O Characteristics.

Note 3: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 4: Exercise great care not to exceed maximum device power dissipation limits when direct-driving LEDs (or sourcing similar loads) at high temperature.

Note 5: SO output "0" level must be less than 0.6V for normal operation.

COP340/COP341/COP342

DC Electrical Characteristics

$-40°C \le T_A \le +85°C$, $4.5V \le V_{CC} \le 5.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Voltage Levels				
Standard Output				
TTL Operation				
Logic High (V_{OH})	$I_{OH} = -100\ \mu A$	2.4		V
Logic Low (V_{OL})	$I_{OL} = 1.6\ mA$		0.4	V
CMOS Operation (Note 1)				
Logic High (V_{OH})	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.5$		V
Logic Low (V_{OL})	$I_{OL} = 10\ \mu A$		0.2	V
Output Current Levels				
Standard Output Source Current	$V_{CC} = 4.5V$, $V_{OH} = 2.4V$	−100	−800	μA
LED Direct Drive Output	$V_{CC} = 5V$ (Note 4)			
Logic High (I_{OH})	$V_{OH} = 2V$	−1.5	−15	mA
TRI-STATE Output Leakage Current		−5.0	+5.0	μA
CKO Output				
Oscillator Output Option				
Logic High	$V_{OH} = 2V$	−0.2		mA
Logic Low	$V_{OL} = 0.4V$	0.4		mA
V_R RAM Power Supply Option				
Supply Current	$V_R = 3.3V$		4.0	mA
CKI Sink Current (RC Option)	$V_{CC} = 4.5V$, $V_{IH} = 3.5V$	2.0		mA
Input Current Levels				
Zero-Crossing Detect Input				
Resistance	$V_{IH} = 1.0V$	0.9	4.6	kΩ
Input Load Source Current	$V_{IH} = 2.0V$, $V_{CC} = 4.5V$	14	280	μA
Total Sink Current Allowed				
All I/O Combined			75	mA
Each L, R Port			20	mA
Each D, G, H Port			10	mA
SO, SK			2.5	mA
Total Source Current Allowed				
All I/O Combined			150	mA
L Port			120	mA
L_7–L_4			70	mA
L_3–L_0			70	mA
Each L Pin			23	mA
All Other Output Pins			1.6	mA

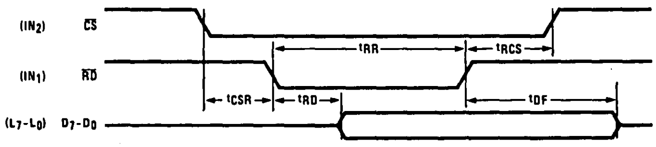
Note 1: TRI-STATE and LED configurations are excluded.

AC Electrical Characteristics

COP440/COP441/COP442: 0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted
COP340/COP341/COP342: − 40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

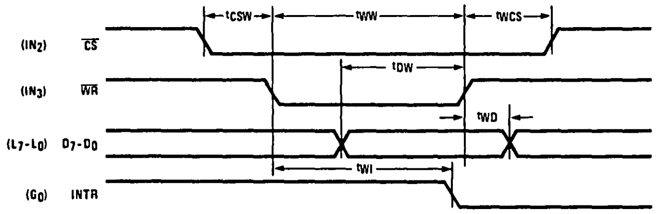
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time-t_E		4.0	10	µs
CKI Frequency	÷16 Mode	1.6	4.0	MHz
	÷8 Mode	0.8	2.0	MHz
	÷4 Mode	0.4	1.0	MHz
Duty Cycle (Note 1)	f_I = 4 MHz	30	60	%
Rise Time	f_I = 4 MHz External Clock		60	ns
Fall Time	f_I = 4 MHz External Clock		40	ns
CKI Using RC *(Figure 9c)*	÷4 Mode			
Frequency	R = 15 kΩ ±5%, C = 100 pF ±10%	0.5	1.0	MHz
Instruction Execution Time-t_E (Note 1)		4.0	8.0	µs
INPUTS: *(Figure 4)*				
SI				
t_{SETUP}		0.3		µs
t_{HOLD}		300		ns
All Other Inputs				
t_{SETUP}		1.7		µs
t_{HOLD}		300		ns
OUTPUT PROPAGATION DELAY	Test Condition: C_L = 50 pF, V_{OUT} = 1.5V			
CKO				
t_{pd1}, t_{pd0}	Crystal Input		0.17	µs
t_{pd1}, t_{pd0}	Schmitt Trigger Input		0.3	µs
SO, SK				
t_{pd1}, t_{pd0}	R_L = 2.4 kΩ		1.0	µs
All Other Outputs	R_L = 5.0 kΩ		1.4	µs
MICROBUS TIMING	C_L = 100 pF, V_{CC} = 5V ±5%			
Read Operation *(Figure 2a)*	TRI-STATE Outputs			
Chip Select Stable Before $\overline{RD}$-t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$-t_{RCS}		20		ns
$\overline{RD}$ Pulse Width-t_{RR}		400		ns
Data Delay from $\overline{RD}$-t_{RD}			375	ns
$\overline{RD}$ to Data Floating-t_{DF}			250	ns
Write Operation *(Figure 2b)*				
Chip Select Stable Before $\overline{WR}$-t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$-t_{WCS}		20		ns
$\overline{WR}$ Pulse Width-t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$-t_{DW}		320		ns
Data Hold Time for $\overline{WR}$-t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$-t_{WI}			700	ns

Note 1: Variation due to the device included.



TL/DD/6926-2

FIGURE 2a. MICROBUS Read Operation Timing

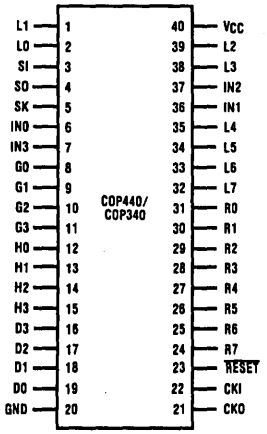


TL/DD/6926-3

FIGURE 2b. MICROBUS Write Operation Timing

Connection Diagrams

Dual-In-Line Package

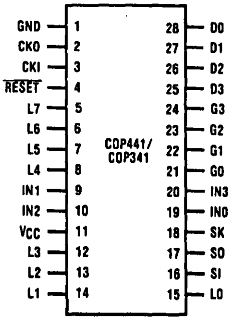


TL/DD/6926-4

Top View

Order Number COP440-XXX/D or COP340-XXX/D
See NS Hermetic Package Number D40C
Order Number COP440-XXX/N or COP340-XXX/N
See NS Molded Package Number N40A

Dual-In-Line Package

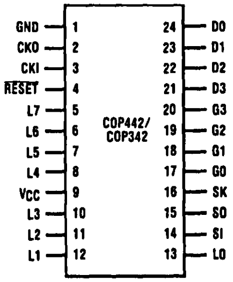


TL/DD/6926-5

Top View

Order Number COP441-XXX/D or COP341-XXX/D
See NS Hermetic Package Number D28C
Order Number COP441-XXX/N or COP341-XXX/N
See NS Molded Package Number N28B

Dual-In-Line Package



TL/DD/6926-6

Top View

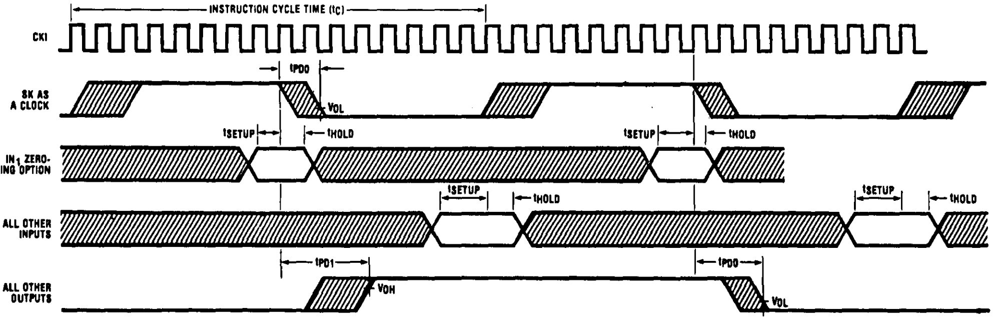
Order Number COP442-XXX/D or COP342-XXX/D
See NS Hermetic Package Number D24C
Order Number COP442-XXX/N or COP342-XXX/N
See NS Molded Package Number N24A

FIGURE 3

Pin Descriptions

Pin	Description
L_7–L_0	8-bit Bidirectional I/O Port with TRI-STATE
G_3–G_0	4-bit Bidirectional I/O Port
D_3–D_0	4-bit General Purpose Output Port
IN_3–IN_0	4-bit General Purpose Input Port (Not Available on COP442/COP342)
SI	Serial Input
SO	Serial Output (or General Purpose Output)
SK	Logic-Controlled Clock (or General Purpose Output)
CKI	System Oscillator Input
CKO	System Oscillator Output (or General Purpose Input or RAM Power Supply)
$\overline{\text{RESET}}$	System Reset Input
V_{CC}	Power Supply
GND	Ground
H_3–H_0	4-bit Bidirectional I/O Port (COP440/COP340 Only)
R_7–R_0	8-Bit Bidirectional I/O Port with TRI-STATE (COP440/COP340 Only)

Timing Diagram



TL/DD/6926-7

FIGURE 4. Input/Output Timing Diagrams (Divide by 16 Mode)

Functional Description

The block diagram of the COP440 is shown in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2.0V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 2,048 byte ROM. As can be seen by an examination of the COP440 instruction set, these words may be program instructions, constants, or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, LQID, and LID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by an 11-bit PC register. Its binary value selects one of the 2,048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 640-bit RAM, organized as 10 data registers of 16 4-bit digits. RAM addressing is implemented by an 8-bit B register whose upper 4 bits (Br) select 1 of 10 (0–9) data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into, or from, or exchanged with the A register (accumulator), it may also be loaded into or from the Q latches, L port, R port, EN register, and T counter (internal time base counter). RAM may also be loaded from 4 bits of a ROM word. RAM addressing may also be performed directly to the lower 8 registers by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs. RAM register 8 (Br = 8) also serves as a subroutine stack. Note that it is possible, but not recommended, to alter the contents of the stack by normal data memory access commands.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O arithmetic, logic, and data memory access operations. It can also be used to load the Br and Bd portions of the B register, N register, to load and input 4 bits of the 8-bit Q latch, EN register, or T counter, to input 4 bits of a ROM word, L or R I/O port data, to input 4-bit G, H, or IN ports, and to perform data exchanges with the SIO register. The accumulator is cleared upon reset.

A 4-bit adder performs the arithmetic and logic functions of the COP440, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

The 8-bit T counter is a binary up counter which can be loaded to and from M and A. The input to this counter is software selectable from two sources: the first coming from a divide-by-four prescaler (from instruction cycle frequency) thus providing a 10-bit time base counter; the second coming from IN_2 input, changing the T counter into an 8-bit external event counter (see EN register below). In this mode, a low-going pulse ("1" to "0") of at least 2 instruction cycles wide will increment the counter. When the counter overflows, an overflow flag will be set (see SKT instruction below) and an interrupt signal will be sent to processor X. The T counter is cleared on reset.

Four general-purpose inputs, IN_3–IN_0, are provided; IN_1, IN_2 and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in MICROBUS applications; IN_1, by another mask-programmable option, can be selected as a true zero-crossing detector with the output triggering an interrupt or being interrogated by an instruction. These two mask-programmable options are mutually exclusive.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G_0 may be mask-programmed as an output for MICROBUS applications.

The H register contents are outputs to a 4-bit general-purpose bidirectional I/O port.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU. Note that unlike most other COPS controllers, Q is cleared on reset.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register. The L I/O port can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The R register, when enabled, outputs to an 8-bit general-purpose, bidirectional, I/O port.

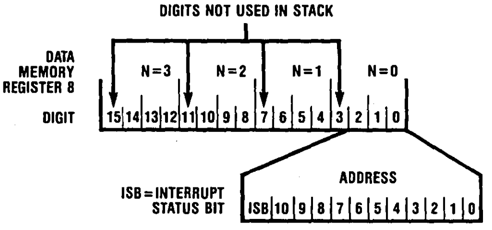
The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register; see EN register description, below). Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream.

The XAS instruction copies the C flag into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the instruction cycle clock.

The 2-bit N register is a stack pointer to the data memory register 8 where the subroutine return address is located. It points to the next location where the address may be stored

Functional Description (Continued)

and increments by 1 after each push of the stack, and decrements by 1 before each pop. The N register can be accessed by exchanging its value with A and is cleared on reset. The stack is 4 addresses deep, 12 bits wide, and does not check for overflow or empty conditions. The RAM digit locations where the addresses are stored are shown in *Figure 5*. The LSBs of the addresses are at digits 0, 4, 8, and 12. The MSBs of digits 2, 6, 10, and 14 contain an interrupt status bit (see Interrupt description, below). The four unused digits (3, 7, 11, and 15) can be used as general data storage. When a subroutine call or interrupt occurs, an 11-bit return address and an interrupt status bit are stored in the stack. The N register is then incremented. When a RET or RETSK instruction is executed, the N register is decremented and then the return address is fetched and loaded into the program counter. The address and interrupt status bits remain in the stack, but will be overwritten when the next subroutine call or interrupt occurs.



TL/DD/6926-8

FIGURE 5. Subroutine Return Address Stack Organization

The EN register in an internal 8-bit register loaded under program control by the LEI instruction (lower 4 bits) or by the CAME instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register.

0. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK output becomes a logic-controlled clock.

1. With EN_1 set, interrupt is enabled with EN_4 and EN_5 selecting the interrupt source. Immediately following an interrupt, EN_1 is reset to disable further interrupts.

2. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN_2 disables the L drivers, placing the L I/O port in a high-impedance input state. A special feature of the COP440 and COP441 is that the MICROBUS option will change the function of this bit to disable any writing into G_0 when EN_2 is set.

3. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0". Table I below provides a summary of the modes associated with EN_3 and EN_0.

4. EN_5 and EN_4 select the source of the interrupt signal.

5. The possible sources are as follows:

EN_5	EN_4	Interrupt Source
0	0	IN_1 (low-going pulse)
0	1	CKO input (if mask-programmed as an input)
1	0	Zero-crossing (or IN_1 level transition)
1	1	T counter overflows

EN_4 determines the interrupt routine location.

6. With EN_6 set, the internal 8-bit T counter will use IN_2 as its input. With EN_6 reset, the input to the T counter is the output of a divide by four prescaler (from instruction cycle frequency), thus providing a 10-bit time-base counter.

7. With EN_7 set, the R outputs are enabled; if EN_7 = 0, the R outputs are disabled.

INTERRUPT

The following features are associated with the interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) together with an interrupt status bit, onto the program counter stack residing in data memory. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset. If EN_4 is reset, the next program address is hex 100; if EN_4 is set, the next program address is hex 300; thus providing a different interrupt location for different interrupt sources.

TABLE I. Enable Register Modes — Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

b. An interrupt will be acknowledged only after the following conditions are met:

1. EN_1 has been set.
2. For an external interrupt input, the signal pulse must be at least two instruction cycles wide.
3. A currently executing instruction has been completed.
4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

c. The instruction at hex address 0FF must be a NOP.

d. A CAME or LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

e. If the interrupt signal source is being changed, the interrupt must be disabled prior to, or at, the same time with the change to avoid false interrupts. An interrupt may be enabled only if the interrupt source is not changing. A sample code for changing the interrupt source and enabling the interrupt is as follows:

```
CAME        ; disable interrupt & alter interrupt source
SMB  1      ; set interrupt enable bit
CAME        ; enable interrupt
```

f. An interrupt status bit is stored together with the return address in the stack. The status bit is set if an interrupt occurs at a point in the program where the next instruction is to be skipped; upon returning from the interrupt routine, this set status bit will cause the next instruction to be skipped. Subroutine and interrupt nesting inside interrupt routines are allowed. Note that this differs from the COP420/420C/420L/444L series.

MICROBUS INTERFACE (not available in COP442, COP342)

The COP440 series has an option which allows them to be used as peripheral microprocessor devices, inputting and outputting data from and to a host microprocessor (μP). IN_1, IN_2 and IN_3 general purpose inputs become MICROBUS-compatible read-strobe, chip-select, and write-strobe lines, respectively. IN_1 becomes $\overline{RD}$—a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN_2 becomes $\overline{CS}$—a logic "0" on this line selects the COPS processor as the μP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN_3 becomes $\overline{WR}$—a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COPS processor. G_0 becomes INTR, a "ready" output, reset by a write pulse from the μP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COPS processor. G_0 output can be separated from other G outputs by the EN_2 bit (see EN description above).

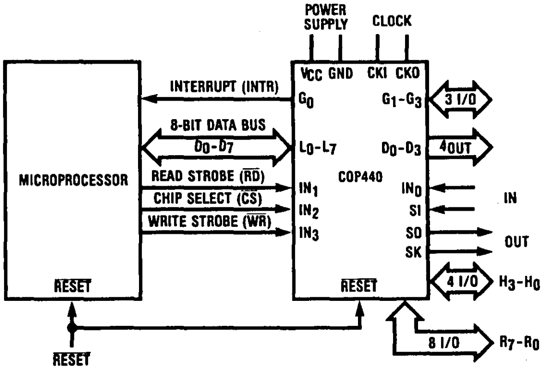
This option has been designed for compatibility with National's MICROBUS—a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS National Publication.) The functional and timing relationships between the COPS processor signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams *(Figure 2)*. Connection of the COP440 to the MICROBUS is shown in *Figure 6*.

Note: TRI-STATE outputs must be used on L port.

ZERO-CROSSING DETECTION (not available on the COP442, COP342)

The following features are associated with the IN_1 pin: ININ and INIL instructions input the state of IN_1 to A_1; IN_1 interrupt generates an interrupt pulse when a low-going transition ("1" to "0") occurs on IN_1; zero-crossing interrupt generates an interrupt pulse when an IN_1 transition occurs (both "1" to "0" and "0" to "1").

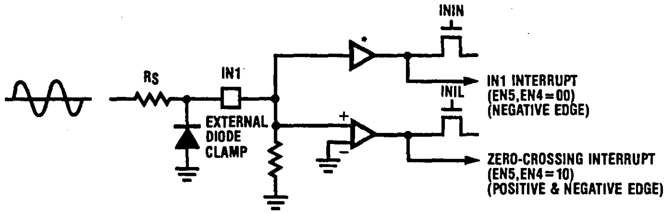
If the zero-crossing detector is mask-programmed in (see *Figure 7a*), the INIL instruction and zero-crossing interrupt will input the state of IN_1 through the true zero-crossing detector ("1" if input > 0V, "0" if input < 0V). The ININ instruction and IN_1 interrupt will then have unique logic HIGH and LOW levels depending on the IN port input level chosen. If normal (TTL) level is chosen, logic HIGH level is 3.0V (3.3V for COP340/341) and logic LOW level is 0.8V



TL/DD/6926-9

FIGURE 6. MICROBUS Option Interconnect

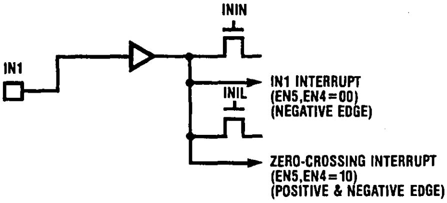
Functional Description (Continued)



TL/DD/6926-10

*Note: This input has a different set of logic HIGH and LOW levels; see above description.

a. Zero-Crossing Detect Logic Option



TL/DD/6926-11

b. IN, without Zero-Crossing Detect Logic

FIGURE 7. IN, Mask-Programmable Options

(0.6V for COP340/341); if high trip level is chosen, logic HIGH level is 5.4V and logic LOW level is 1.2V. If the zero-crossing detector is not mask-programmed in (see *Figure 7b*), IN_1 will have logic HIGH and LOW levels that are defined for the IN port (see option list).

The zero-crossing detector input contains a small hysteresis (50 mV typical) to eliminate signal noise, and is not a high impedance input but contains a resistive load to ground. Since this input can withstand a voltage range of −0.8V to +12V, an external clamping diode is needed for most input signals, as shown in *Figure 7a*, to limit the voltage below ground. An external resistor, R_S may be needed for the following two cases:

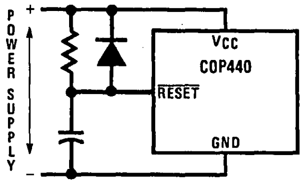
a. Input signal exceeds 12V; R_S and the internal resistor act as a voltage divider to reduce the voltage at the input pin to below 12V.

b. Signal comes from a low impedance source; when the voltage at the pin is clamped to −0.7V by the forward bias voltage of an external diode, R_S limits the current going through the diode.

INITIALIZATION

The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times. The user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as in *Figure 8*. The external POR (Power-on-Reset) delay must be greater than the internal POR. The internal POR delay is 2600 internal clock cycles.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, G, H, IL, L, N, Q, R, and T registers are cleared. The SK output is enabled as a SYNC output by setting the SKL latch, thus providing a clock. RAM (data memory and stack) is not cleared. The first instruction at address 0 must be a CLRA.



TL/DD/6926-12

RC ≥ 5 × power supply rise time

FIGURE 8. Power-Up Clear Circuit

OSCILLATOR

There are three basic clock oscillator configurations available, as shown by *Figure 9*.

a. **Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The cycle frequency equals the crystal frequency divided by 16 (optional by 8). Thus a 4 MHz crystal with the divide-by-16 option selected will give a 250 kHz cycle frequency (4 μs instruction cycle time).

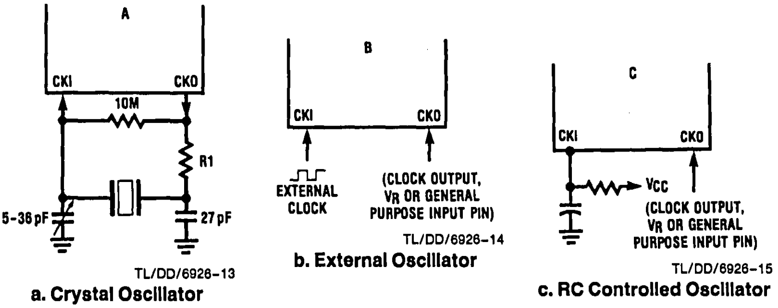
b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 16 (optional by 8 or 4) to give the cycle frequency. If the divide-by-4 option is selected, the CKI input level is the Schmitt-trigger level. CKO is now available to be used as the RAM power supply (V_R) or as a general purpose input.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The cycle frequency equals the oscillation frequency divided by 4. CKO is available for non-timing functions.

CKO PIN OPTIONS

As an option, CKO can be an oscillator output. In a crystal controlled oscillator system, this signal is used as an output to the crystal network. As another option, CKO can be an interrupt input or a general purpose input, reading into bit 2 of A (accumulator) through the INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing

Functional Description (Continued)


A
CKI
CKO
10M
R1
5-36pF
27pF
B
CKI
CKO
EXTERNAL CLOCK
(CLOCK OUTPUT, V_R OR GENERAL PURPOSE INPUT PIN)
C
CKI
CKO
V_{CC}
(CLOCK OUTPUT, V_R OR GENERAL PURPOSE INPUT PIN)

TL/DD/6926-13

a. Crystal Oscillator

TL/DD/6926-14

b. External Oscillator

TL/DD/6926-15

c. RC Controlled Oscillator

Crystal Oscillator

Crystal Value	R_1
4 MHz	1k
3.58 MHz	1k
2.10 MHz	2k

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Execution Time (μs)
13	100	5.0 ±20%
6.8	220	5.3 ±23%
8.2	300	8.0 ±22%
22	100	8.2 ±17%

Note: 5 kΩ ≤ R ≤ 50 kΩ
50 pF ≤ C ≤ 360 pF

FIGURE 9. COP440/441/442 Oscillators

its connection to a standby/backup power supply to maintain the data integrity of RAM registers 0–3 with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either of the two latter options is appropriate in applications where the system configuration does not require use of the CKO pin for timing functions.

RAM KEEP-ALIVE OPTION

Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the lower 4 registers of the RAM. To insure that RAM data integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. When V_{CC} is on, V_R must be within the operating voltage range of the chip, and within 1V of V_{CC}.
3. V_R must be ≥ 3.3V with V_{CC} off.

I/O OPTIONS

COP440 inputs have the following optional configurations, illustrated in *Figure 10.*

a. An on-chip depletion load device to V_{CC}.

b. A Hi-Z input which must be driven to a "1" or "0" by external components.

c. A resistive load to GND for the zero-crossing input option (IN_1 only).

COP440 outputs have the following optional configurations:

d. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements. Available on SO, SK, D, G, and H outputs.

e. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, D, G, L, H, and R outputs.

f. Push-Pull—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

g. Standard L,R—same as **d.**, but may be disabled. Available on L and R outputs only (disabled on reset).

h. LED Direct Drive—an enhancement-mode device to ground and V_{CC} together with a depletion device to V_{CC} meeting the typical current sourcing requirements of the segments of an LED display. The sourcing devices are clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the output in a high-impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note 1: When the driver is disabled, the depletion device may cause the output to settle down to an intermediate level between V_{CC} and GND. This voltage cannot be relied upon as a "1" level when reading the L inputs. The external signal must drive it to a "1" level.

Note 2: Much power is dissipated by this driver in driving an LED. Care must be taken to limit the power dissipation of the chip to within the absolute maximum ratings specified.

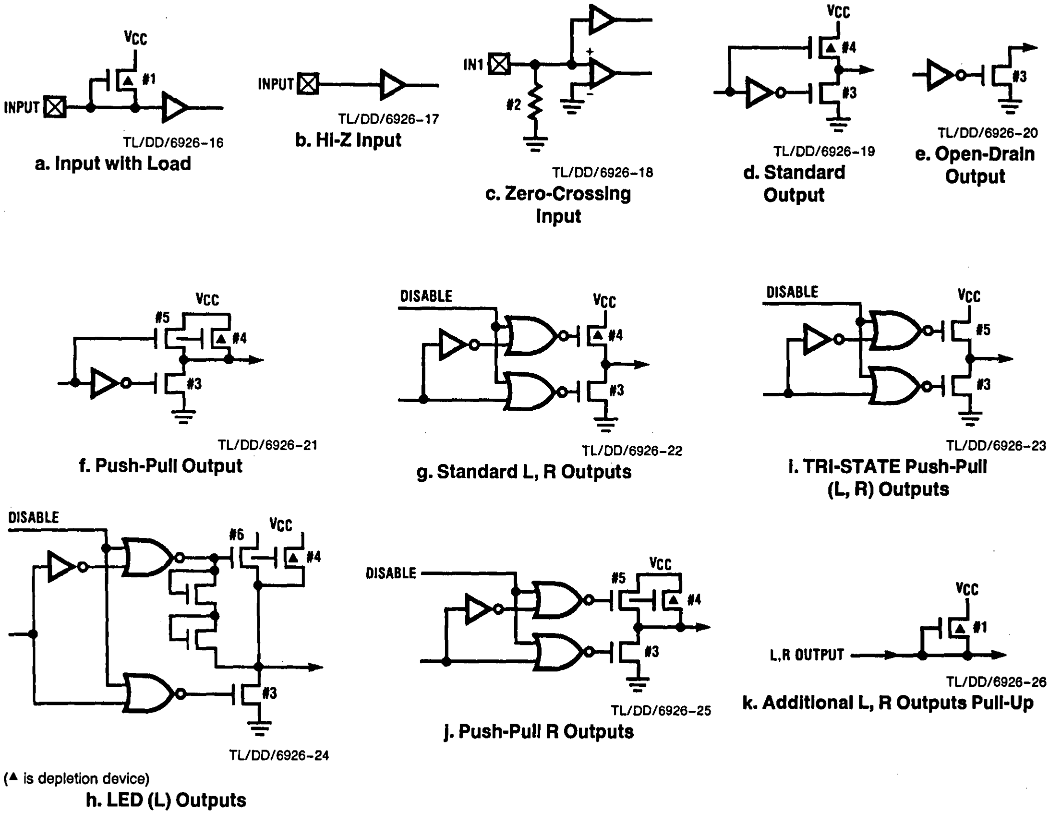
i. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L and R outputs only (in TRI-STATE mode on reset).

j. Push-Pull R—same as **f.**, but may be disabled. Available on R outputs only.

k. Additional depletion pull-up—a depletion load to V_{CC} with the same current sourcing capability as the input load **a.**, in addition to the output drive chosen. Available on L and R outputs only. *This device cannot be disabled*; therefore, open-drain outputs with "1" output and TRI-STATE outputs do not show high-impedance characteristics. This device is useful in applications where a pull-up with low source current is desired, e.g., reading keyboards and switches.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6 respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figures 11* and *12* for each of these devices to allow the designer to effectively use these I/O configurations in designing a COP440 system.

Functional Description (Continued)



a. Input with Load

b. Hi-Z Input

c. Zero-Crossing Input

d. Standard Output

e. Open-Drain Output

f. Push-Pull Output

g. Standard L, R Outputs

i. TRI-STATE Push-Pull (L, R) Outputs

(▲ is depletion device)

h. LED (L) Outputs

j. Push-Pull R Outputs

k. Additional L, R Outputs Pull-Up

FIGURE 10. Input/Output Configurations

L-BUS CONSIDERATIONS

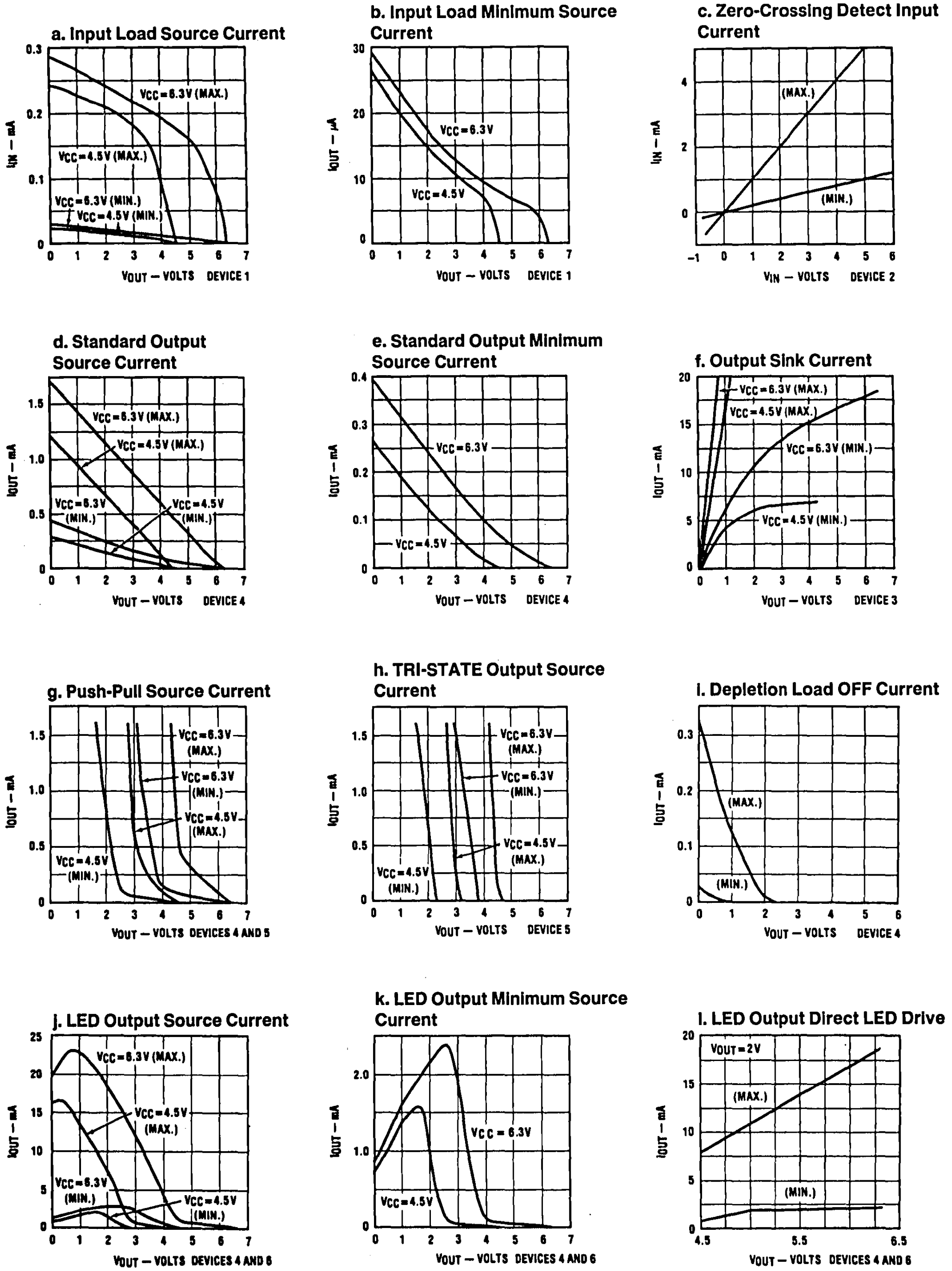
False states may be generated on L_0–L_7 during the execution of the CAMQ instruction. The L-Ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. The following short program illustrates this situation.

Glitch Test Program

```
START:
        CLRA            ;ENABLE THE Q
        LEI    4        ;REGISTER TO L LINES
        LBI    TEST
        STII   3
        AISC   12
LOOP:
        LBI    TEST     ;LOAD Q WITH X'C3
        CAMQ
        JP     LOOP
```

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L_0, L_1, L_6, L_7, and logic lows on L_2–L_5 via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L_0, L_1, L_6, L_7, and positive glitches on L_2–L_5. Glitch durations are under 2 μs, although the exact value may vary due to data patterns, processing parameters, and L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics



a. Input Load Source Current

b. Input Load Minimum Source Current

c. Zero-Crossing Detect Input Current

d. Standard Output Source Current

e. Standard Output Minimum Source Current

f. Output Sink Current

g. Push-Pull Source Current

h. TRI-STATE Output Source Current

i. Depletion Load OFF Current

j. LED Output Source Current

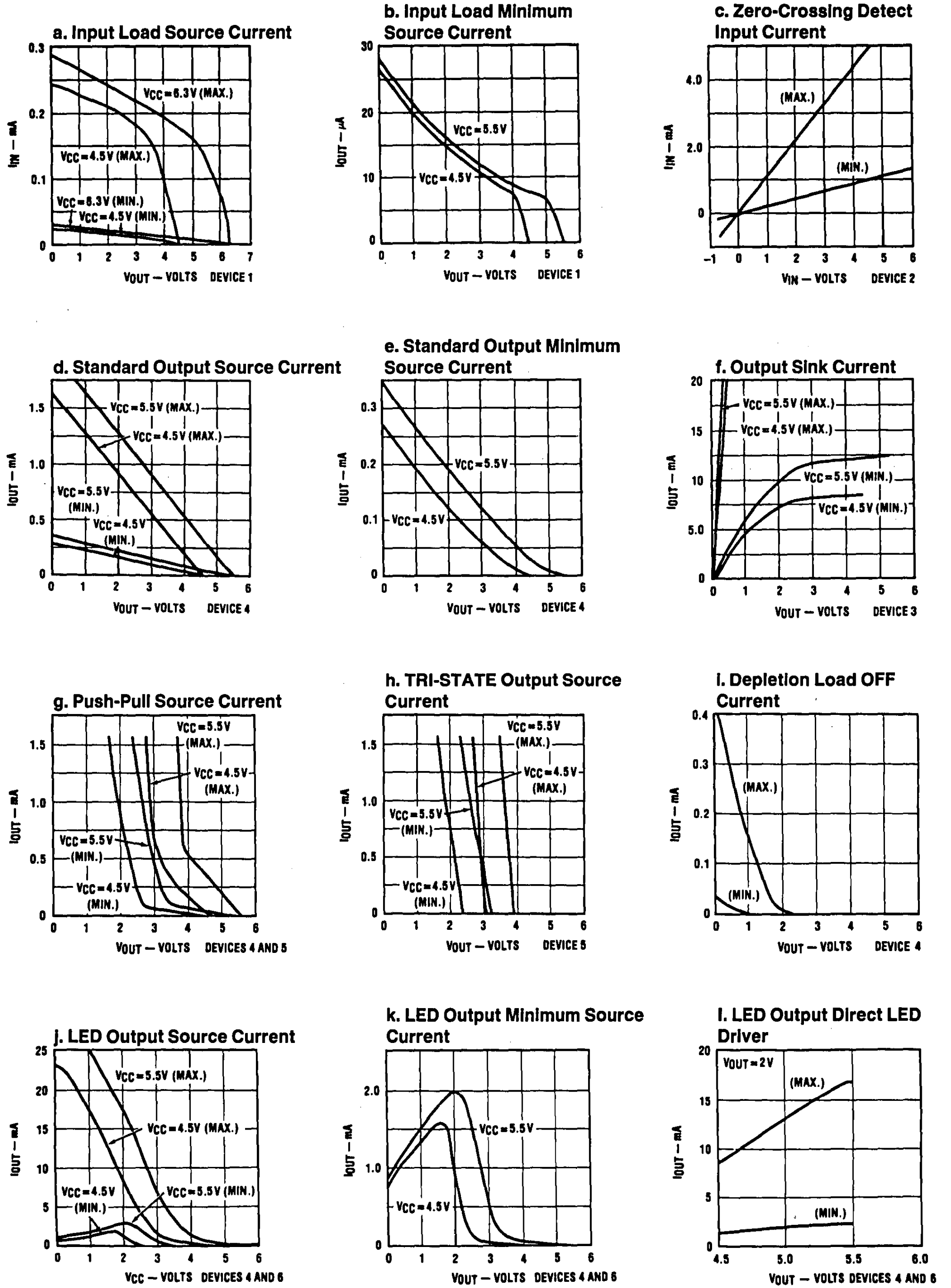
k. LED Output Minimum Source Current

l. LED Output Direct LED Drive

TL/DD/6926–27

FIGURE 11. COP440/441/442 I/O Characteristics

Typical Performance Characteristics (Continued)


a. Input Load Source Current

b. Input Load Minimum Source Current

c. Zero-Crossing Detect Input Current

d. Standard Output Source Current

e. Standard Output Minimum Source Current

f. Output Sink Current

g. Push-Pull Source Current

h. TRI-STATE Output Source Current

i. Depletion Load OFF Current

j. LED Output Source Current

k. LED Output Minimum Source Current

l. LED Output Direct LED Driver

TL/DD/6926-28

FIGURE 12. CCOP340/341/342 I/O Characteristics

Power Dissipation

In order not to damage the device by exceeding the absolute maximum power dissipation rating, the amount of power dissipated inside the chip must be carefully controlled. As an example, an application uses a COP440 in room temperature (25°C) environment with a V_{CC} power supply of 6V; IN and SI inputs have internal loads; G and D ports drive loads that may sink up to 2 mA into the chip; H port with standard output option reads switches; L port with the LED option drives a multiplexed seven-segment display; R, SO and SK drive MOS inputs that do not source or sink any current.

a. At 25°C, maximum power dissipation allowed = 750 mW

b. Power dissipation by chip except

$$I/O = I_{CC} \times V_{CC} = 35\text{ mA} \times 6V = 210\text{ mW}$$

c. Maximum power dissipation by IN,

$$SI = 5 \times 0.3\text{ mA} \times 6V = 9\text{ mW}$$

d. G and D ports are sinking current from external loads; maximum output voltage with 2 mA sink current is less than 0.4V. Power dissipation by G and D ports =

$$2\text{ mA} \times 0.4V \times 8 = 6.4\text{ mW}$$

e. Maximum power dissipation by H port =

$$4 \times 1.5\text{ mA} \times 6V = 36\text{ mW}$$

f. When the seven segments of the LED are turned on, the output voltage is about 2V, so that the segment current is 17 mA. Power dissipation by L port =

$$7 \times 17\text{ mA} \times (6V - 2V) = 476\text{ mW}$$

This power dissipation caused by driving LEDs is usually the highest among the various sources.

g. R, SO, and SK do not dissipate any significant amount of power because they do not need to source or sink any current.

Total power dissipation (TPD) inside the device is the sum of items b through g above.

$$TPD = 210 + 9 + 6 + 36 + 476\text{ mW} = 737\text{ mW}$$

This is within the 750 mW limit at room temperature. If this application has to operate at 70°C, then the power dissipation must be reduced to meet the limit at that temperature. Some ways to achieve this would be to limit the LED current or to use an external LED driver.

At 70°C the absolute maximum power dissipation rating drops to 400 mW. The user must be careful not to exceed this value.

COP440 SERIES DEVICES

If the COP440 is bonded as a 28- or 24-pin device, it becomes the COP441 or COP442, respectively, as illustrated in *Figure 3*. Note that the COP441 and COP442 do not include H and R ports. In addition, the COP442 does not include IN inputs; use of this option precludes the use of the IN options, the interrupt feature with IN as input, the zero-crossing detect option, IN_2 external event counter input, and the MICROBUS option. All other options are available.

COP340, COP341, and COP342 are extended temperature versions of the COP440, COP441, and COP442, respectively.

COP440 Series Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP440 series instruction set.

TABLE II. COP440 Series Instruction Set Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	8-bit RAM Address Register
Br	Upper 4 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	8-bit Enable Register
G	4-bit Register to latch data for G I/O Port
H	4-bit Register to latch data for H I/O Port
IL	Two 1-bit Latches associated with the IN_3 or IN_0 Inputs
IN	4-bit Input Port
IN_1Z	Zero-Crossing Input
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
N	2-bit subroutine return address stack pointer
PC	11-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
R	8-bit Register to latch data for R TRI-STATE I/O Port
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output
T	8-bit Binary Counter Register

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	4-bit Operand Field, 0–9 binary (RAM Register Select)
a	11-bit Operand Field, 0–2047 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Content of RAM location addressed by s
RAM_N	Content of RAM location addressed by stack pointer N
ROM(t)	Content of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values
V	OR

Instruction Set

TABLE III. COP440 Series Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC/LOGIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
ADT		4A	0100 1010	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	0101 y	A + y → A	Carry	Add immediate, Skip on Carry ($y \neq 0$)
CASC		10	0001 0000	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
OR		33 1A	0011 0011 0001 1010	A ∨ M → A	None	OR RAM with A
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{10:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– – –	0110 0 $a_{10:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	– – – –	1 $a_{6:0}$ (pages 2,3 only) or 11 $a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 4)
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → RAM_N N + 1 → N 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	0110 1 $a_{10:8}$ $a_{7:0}$	PC + 1 → RAM_N N + 1 → N a → PC	None	Jump to Subroutine
RET		48	0100 1000	N – 1 → N RAM_N → PC	None	Return from Subroutine
RETSK		49	0100 1001	N – 1 → N RAM_N → PC	Always Skip on Return	Return from Subroutine then Skip

Instruction Set (Continued)

TABLE III. COP440 Series Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS						
CAME		33 1F	0011 0011 0001 1111	$A \rightarrow EN_{7:4}$ $RAM(B) \rightarrow EN_{3:0}$	None	Copy A, RAM to EN
CAMQ		33 3C	0011 0011 0011 1100	$A \rightarrow Q_{7:4}$ $RAM(B) \rightarrow Q_{3:0}$	None	Copy A, RAM to Q
CAMT		33 3F	0011 0011 0011 1111	$A \rightarrow T_{7:4}$ $RAM(B) \rightarrow T_{3:0}$	None	Copy A, RAM to T
CEMA		33 0F	0011 0011 0000 1111	$EN_{7:4} \rightarrow RAM(B)$ $EN_{3:0} \rightarrow A$	None	Copy EN to RAM, A
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4} \rightarrow RAM(B)$ $Q_{3:0} \rightarrow A$	None	Copy Q to RAM, A
CTMA		33 2F	0011 0011 0010 1111	$T_{7:4} \rightarrow RAM(B)$ $T_{3:0} \rightarrow A$	None	Copy T to RAM, A
LD	r	-5	00 r 0101 r = 0:3	$RAM(B) \rightarrow A$ $Br \oplus r \rightarrow Br$	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 --	00 10 0011 0 r d r = 0:7	$RAM(r,d) \rightarrow A$	None	Load A with RAM pointed to directly by r,d
LID		33 19	0011 0011 0001 1001	$ROM (PC_{10:8}, A, M) \rightarrow M, A$	None	Load RAM, A Indirect
LQID		BF	1011 1111	$ROM(PC_{10:8},A,M) \rightarrow Q$	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	$0 \rightarrow RAM(B)_0$ $0 \rightarrow RAM(B)_1$ $0 \rightarrow RAM(B)_2$ $0 \rightarrow RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	$1 \rightarrow RAM(B)_0$ $1 \rightarrow RAM(B)_1$ $1 \rightarrow RAM(B)_2$ $1 \rightarrow RAM(B)_3$	None	Set RAM Bit
STII	y	7-	0111 y	$y \rightarrow RAM(B)$ $Bd + 1 \rightarrow Bd$	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110 r = 0:3	$RAM(B) \longleftrightarrow A$ $Br \oplus r \rightarrow Br$	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 0011 1 r d r = 0:7	$RAM(r,d) \longleftrightarrow A$	None	Exchange A with RAM pointed to directly by r,d
XDS	r	-7	00 r 0111 r = 0:3	$RAM(B) \longleftrightarrow A$ $Bd - 1 \rightarrow Bd$ $Br \oplus r \rightarrow Br$	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	00 r 0100 r = 0:3	$RAM(B) \longleftrightarrow A$ $Bd + 1 \rightarrow Bd$ $Br \oplus r \rightarrow Br$	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

Instruction Set (Continued)

TABLE III. COP440 Series Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	$A \rightarrow Bd$	None	Copy A to Bd
CBA		4E	|0100|1110|	$Bd \rightarrow A$	None	Copy Bd to A
LBI	r,d	--	|00|r|(d − 1)| r = 0:3, d = 0,9:15 or	$r,d \rightarrow B$	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33 --	|0011|0011| |1|r|d| r = 0:7, any d			
LEI	y	33 6–	|0011|0011| |0110|y|	$y \rightarrow EN_{3:0}$	None	Load lower half of EN Immediate
XABR		12	|0001|0010|	$A \longleftrightarrow Br$	None	Exchange A with Br
XAN		33 0B	|0011|0011| |0000|1011|	$A \longleftrightarrow N(0,0 \rightarrow A_3, A_2)$	None	Exchange A with N
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C = "1"	Skip if C is True
SKE		21	|0010|0001|		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	|0011|0011| |0010|0001|		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|	2nd byte	$G_0 = 0$	
	1	11	|0001|0001|		$G_1 = 0$	
	2	03	|0000|0011|		$G_2 = 0$	
	3	13	|0001|0011|		$G_3 = 0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1 = 0$	
	2	03	|0000|0011|		$RAM(B)_2 = 0$	
	3	13	|0001|0011|		$RAM(B)_3 = 0$	
SKSZ		33 1C	|0011|0011| |0001|1100|		SIO = 0	Skip if SIO is Zero
SKT		41	|0100|0001|		T counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP440 Series Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
CAMR		33 3D	0011 0011 0011 1101	$A \rightarrow R_{7:4}$ $RAM(B) \rightarrow R_{3:0}$	None	Output A, RAM to R Port
ING		33 2A	0011 0011 0010 1010	$G \rightarrow A$	None	Input G Port to A
INH		33 2B	0011 0011 0010 1011	$H \rightarrow A$	None	Input H Port to A
ININ		33 28	0011 0011 0010 1000	$IN \rightarrow A$	None	Input IN Inputs to A (Note 2)
INIL		33 29	0011 0011 0010 1001	IL_3, CKO, IN_1Z, $IL_0 \rightarrow A$	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 0011 0010 1110	$L_{7:4} \rightarrow RAM(B)$ $L_{3:0} \rightarrow A$	None	Input L Port to RAM, A
INR		33 2D	0011 0011 0010 1101	$R_{7:4} \rightarrow RAM(B)$ $R_{3:0} \rightarrow A$	None	Input R Port to RAM,A
OBD		33 3E	0011 0011 0011 1110	$Bd \rightarrow D$	None	Output Bd to D Port
OGI	y	33 5-	0011 0011 0101 y	$y \rightarrow G$	None	Output to G Port Immediate
OMG		33 3A	0011 0011 0011 1010	$RAM(B) \rightarrow G$	None	Output RAM to G Port
OMH		33 3B	0011 0011 0011 1011	$RAM(B) \rightarrow H$	None	Output RAM to H Port
XAS		4F	0100 1111	$A \longleftrightarrow SIO, C \rightarrow SKL$	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin COP442/COP342 since this device does not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (00010 is loaded into the upper 5 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP440 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN register, above). If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

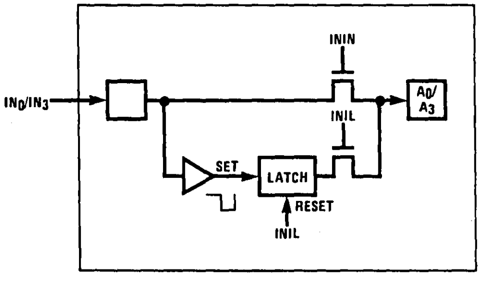
JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, $PC_{10:8}$, A, M. PC_{10}, PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles if executed, 1 instruction cycle time if skipped.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0, CKO and IN_1 into A (see *Figure 13*). The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask-programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. Unlike the COP420/420C/420L/444L series, INIL will input IN_1 into A1.



TL/DD/6926-29

FIGURE 13. INIL Hardware Implementation

If zero-crossing detect is selected, the IN_1 input will go through the detection logic, thus allowing the user to interrogate the input, sending a "1" if the input is above 0V and a "0" if it is below 0V. INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction. It is also useful in checking the status of the zero-crossing detect input. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction, and the IN_1 input does not go through zero-crossing logic so that it has the same logic level as the other IN inputs for the ININ instruction (see *Figure 9*).

Note: IL latches are cleared on reset. This is different from the COP420/420C/420L/444L series.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC_{10}:PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. Note that LQID takes two instruction cycles if executed and one instruction cycle if skipped. Unlike most other COPS processors, this instruction does not push the stack.

LID INSTRUCTION

LID (Load Indirect) loads M and A with the contents of ROM pointed to by the 11-bit word PC_{10}:PC_8, A, M. Note that LID takes three instruction cycles if executed and two if skipped.

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter (see internal logic, above) overflow latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal.

INSTRUCTION SET NOTES

a. The first word of a COP440 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from program memory. Thus program paths take the same number of cycle times whether instructions are skipped or executed, except for LID, LQID, and JID.

c. The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, LQID, or LID instruction is the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23, 27, or 31 will access data in the next group of four pages.

Option List

The COP440 mask-programmable options are assigned numbers which correspond with the COP440 pins.

Option 1: L_1 I/O Port (see note below)
= 0: Standard output
= 1: Open-drain output
= 2: LED direct drive output
= 3: TRI-STATE output
= 4: same as 0 with extra load device to V_{CC}
= 5: same as 1 with extra load device to V_{CC}
= 6: same as 2 with extra load device to V_{CC}
= 7: same as 3 with extra load device to V_{CC}

Option 2: L_0 I/O Port
(same as Option 1)

Option 3: SI Input
= 0: Input with load device to V_{CC}
= 1: Hi-Z Input

Option 4: SO Output
= 0: Standard output
= 1: Open-drain output
= 2: Push-pull output

Option 5: SK Output
(same as Option 4)

Option 6: IN_0 Input
(same as Option 3)

Option 7: IN_3 Input
(same as Option 3)

Option 8: G_0 I/O Port
= 0: Standard output
= 1: Open-drain output

Option 9, G_1 I/O Port
(same as Option 8)

Option 10: G_2 I/O Port
(same as Option 8)

Option 11: G_3 I/O Port
(same as Option 8)

Option 12: H_0 I/O Port
(same as Option 8)

Option 13: H_1 I/O Port
(same as Option 8)

Option 14: H_2 I/O Port
(same as Option 8)

Option 15: H_3 I/O Port
(same as Option 8)

Option 16: D_3 Output
(same as Option 8)

Option 17: D_2 Output
(same as Option 8)

Option 18: D_1 Output
(same as Option 8)

Option 19: D_0 Output
(same as Option 8)

Option 20: GND—No options available

Option 21: CKO Pin
= 0: Oscillator output
= 1: RAM power supply (V_R) input
= 2: General purpose input with load device to V_{CC}
= 3: General purpose Hi-Z input

Option 22: CKI Input
= 0: Crystal input divided by 16
= 1: Crystal input divided by 8
= 2: Single-pin RC controlled oscillator (÷4)
= 3: Schmitt trigger clock input (÷4)

Option 23: $\overline{\text{RESET}}$ Input
(same as Option 3)

Option 24: R_7 I/O Port (see note below)
= 0: Standard output
= 1: Open-drain output
= 2: Push-pull output
= 3: TRI-STATE output
= 4: same as 0 with extra load device to V_{CC}
= 5: same as 1 with extra load device to V_{CC}
= 6: same as 2 with extra load device to V_{CC}
= 7: same as 3 with extra load device to V_{CC}

Option 25: R_6 I/O Port
(same as Option 24)

Option 26: R_5 I/O Port
(same as Option 24)

Option 27: R_4 I/O Port
(same as Option 24)

Option 28: R_3 I/O Port
(same as Option 24)

Option 29: R_2 I/O Port
(same as Option 24)

Option 30: R_1 I/O Port
(same as Option 24)

Option 31: R_0 I/O Port
(same as Option 24)

Option 32: L_7 I/O Port
(same as Option 1)

Option 33: L_6 I/O Port
(same as Option 1)

Option 34: L_5 I/O Port
(same as Option 1)

Option 35: L_4 I/O Port
(same as Option 1)

Option 36: IN_1 Input
= 0: Input with load device to V_{CC}
= 1: Hi-Z Input
= 2: Zero-crossing detect input (Option 41 = 0)

Option 37: IN_2 Input
(same as Option 3)

Option 38: L_3 I/O Port
(same as Option 1)

Option 39: L_2 I/O Port
(same as Option 1)

Option 40: V_{CC}—no options available

Option List (Continued)

Option 41: COP Function
= 0: Normal
= 1: MICROBUS option

Option 42: IN Input Levels
= 0: Standard TTL input levels ("0" = 0.8V, "1" = 2.0V)
= 1: Higher voltage input levels ("0" = 1.2V, "1" = 3.6V)

Option 43: G Input Levels
(same as Option 42)

Option 44: L Input Levels
(same as Option 42)

Option 45: CKO Input Levels
(same as Option 42)

Option 46: SI Input Levels
(same as Option 42)

Option 47: R Input Levels
(same as Option 42)

Option 48: H Input Levels
(same as Option 42)

Option 49: No option available

Option 50: COP Bonding
= 0: COP440 (40-pin device)
= 1: COP441 (28-pin device)
= 2: COP442 (24-pin device)
= 3: COP440 and COP441
= 4: COP440 and COP442
= 5: COP440, COP441, and COP442
= 6: COP441 and COP442

COP440 Option Table

The following options information is to be sent to National along with the EPROM.

OPTION 1 VALUE = ________ IS: L_1 I/O PORT
OPTION 2 VALUE = ________ IS: L_0 I/O PORT
OPTION 3 VALUE = ________ IS: SI INPUT
OPTION 4 VALUE = ________ IS: SO OUTPUT
OPTION 5 VALUE = ________ IS: SK OUTPUT
OPTION 6 VALUE = ________ IS: IN_0 INPUT
OPTION 7 VALUE = ________ IS: IN_3 INPUT
OPTION 8 VALUE = ________ IS: G_0 I/O PORT
OPTION 9 VALUE = ________ IS: G_1 I/O PORT
OPTION 10 VALUE = ________ IS: G_2 I/O PORT
OPTION 11 VALUE = ________ IS: G_3 I/O PORT
OPTION 12 VALUE = ________ IS: H_0 I/O PORT
OPTION 13 VALUE = ________ IS: H_1 I/O PORT
OPTION 14 VALUE = ________ IS: H_2 I/O PORT
OPTION 15 VALUE = ________ IS: H_3 I/O PORT
OPTION 16 VALUE = ________ IS: D_3 OUTPUT
OPTION 17 VALUE = ________ IS: D_2 OUTPUT
OPTION 18 VALUE = ________ IS: D_1 OUTPUT
OPTION 19 VALUE = ________ IS: D_0 OUTPUT
OPTION 20 VALUE = ___0___ IS: GROUND PIN
OPTION 21 VALUE = ________ IS: CKO PIN
OPTION 22 VALUE = ________ IS: CKI INPUT
OPTION 23 VALUE = ________ IS: RESET INPUT
OPTION 24 VALUE = ________ IS: R_7 I/O PORT
OPTION 25 VALUE = ________ IS: R_6 I/O PORT
OPTION 26 VALUE = ________ IS: R_5 I/O PORT
OPTION 27 VALUE = ________ IS: R_4 I/O PORT
OPTION 28 VALUE = ________ IS: R_3 I/O PORT
OPTION 29 VALUE = ________ IS: R_2 I/O PORT
OPTION 30 VALUE = ________ IS: R_1 I/O PORT
OPTION 31 VALUE = ________ IS: R_0 I/O PORT
OPTION 32 VALUE = ________ IS: L_7 I/O PORT
OPTION 33 VALUE = ________ IS: L_6 I/O PORT
OPTION 34 VALUE = ________ IS: L_5 I/O PORT
OPTION 35 VALUE = ________ IS: L_4 I/O PORT
OPTION 36 VALUE = ________ IS: IN_1 INPUT
OPTION 37 VALUE = ________ IS: IN_2 INPUT
OPTION 38 VALUE = ________ IS: L_3 I/O PORT
OPTION 39 VALUE = ________ IS: L_2 I/O PORT
OPTION 40 VALUE = ___0___ IS: V_{CC}
OPTION 41 VALUE = ________ IS: COP FUNCTION
OPTION 42 VALUE = ________ IS: IN INPUT LEVELS
OPTION 43 VALUE = ________ IS: G INPUT LEVELS
OPTION 44 VALUE = ________ IS: L INPUT LEVELS
OPTION 45 VALUE = ________ IS: CKO INPUT LEVELS
OPTION 46 VALUE = ________ IS: SI INPUT LEVELS
OPTION 47 VALUE = ________ IS: R INPUT LEVELS
OPTION 48 VALUE = ________ IS: H INPUT LEVELS
OPTION 49 VALUE = ________ IS: NO OPTION
OPTION 50 VALUE = ________ IS: COP BONDING

Note on L and R I/O Port Options

If L and R I/O Ports are used as inputs, the following must be observed:

a. Open-Drain output (selection 1) is allowed only if external pull-up is provided.

b. If L and R output ports are disabled when reading, an external pull-up is required unless selections 4, 5, 6, or 7 are chosen.

c. If L output port is enabled, selections 3 and 7 are not allowed.

d. If R output port is enabled, selections 2, 3, 6, and 7 are not allowed.

Test Mode (Non-Standard Operation)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP440. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

National Semiconductor

COP444L/COP445L/COP344L/COP345L Single-Chip N-Channel Microcontrollers

General Description

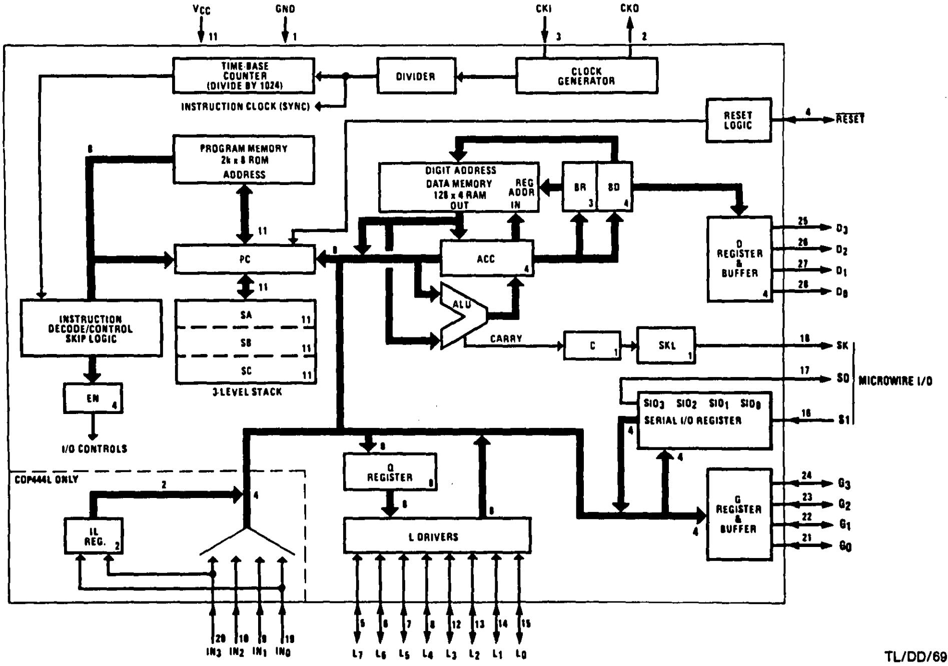
The COP444L, COP445L, COP344L, and COP345L Single-Chip N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. These controller oriented processors are complete microcomputers containing all system timing, internal logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include single supply operation, a variety of output configuration options, with an instruction set, internal architecture and I/O scheme designed to facilitate keyboard input, display output and BCD data manipulation. The COP445L is identical to the COP444L, but with 19 I/O lines instead of 23. They are an appropriate choice for use in numerous human interface control environments. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a customized controller oriented processor at a low end-product cost.

The COP344L and COP345L are exact functional equivalents, but extended temperature range versions of the COP444L and COP445L respectively.

Features

- Low cost
- Powerful instruction set
- 2k x 8 ROM, 128 x 4 RAM
- 23 I/O lines (COP444L)
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 15 μs instruction time
- Single supply operation (4.5–6.3V)
- Low current drain (11 mA max.)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Extended temperature range devices COP344L/COP345L (−40°C to +85°C)

Block Diagram



TL/DD/6928-1

FIGURE 1

COP444L/COP445L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	0.75 Watt at 25°C
	0.4 Watt at 70°C
Total Source Current	120 mA
Total Sink current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolulte maximum ratings.

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$, $4.5V \leq V_{CC} \leq 6.3V$ unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	6.3	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		13	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input (÷32, ÷16, ÷8)				
Logic High (V_{IH})	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0	0.4	V
Logic Low (V_{IL})		−0.3		
Schmitt Trigger Input (÷4)				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.6	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
SO Input Level (Test Mode)	(Note 3)	2.0	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	With TTL Trip Level Options	2.0		V
Logic Low	Selected, V_{CC} = 5V ±10%	−0.3	0.8	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±5%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)				
Logic High	I_{OH} = −10 μA	V_{CC}−1		V
Logic Low	I_{OL} = +10μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.8V for normal operation.

COP444L/COP445L (Continued)

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$, $4.5V \leq V_{CC} \leq 6.3V$ unless otherwise noted. (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{CC} = 6.3V, V_{OL} = 0.4V	1.2		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L_0–L_7 Outputs and Standard	V_{CC} = 6.3V, V_{OL} = 0.4V	0.4		mA
G_0–G_3, D_0–D_3 Outputs (I_{OL})	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	V_{CC} = 6.3V, V_{OL} = 1.0V	11		mA
High Current Options (I_{OL})	V_{CC} = 4.5V, V_{OL} = 1.0V	7.5		mA
G_0–G_3 and D_0–D_3 Outputs with	V_{CC} = 6.3V, V_{OL} = 1.0V	22		mA
Very High Current Options (I_{OL})	V_{CC} = 4.5V, V_{OL} = 1.0V	15		mA
CKI (Single-pin RC oscillator)	V_{CC} = 4.5V, V_{IH} = 3.5V	2		mA
CKO	V_{CC} = 4.5V, V_{OL} = 0.4V	0.2		mA
Output Source Current				
Standard Configuration,	V_{CC} = 6.3V, V_{OH} = 2.0V	−75	−480	μA
All Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.0V	−30	−250	μA
Push-Pull Configuration	V_{CC} = 9.5V, V_{OH} = 4.75V	−1.4		mA
SO and SK Outputs (I_{OH})	V_{CC} = 6.3V, V_{OH} = 2.4V	−1.4		mA
	V_{CC} = 4.5V, V_{OH} = 1.0V	−1.2		mA
LED Configuration, L_0–L_7				
Outputs, Low Current				
Drivers Option (I_{OH})	V_{CC} = 6.0V, V_{OH} = 2.0V	−1.5	−13	mA
LED Configuration, L_0–L_7				
Outputs, High Current				
Driver Option (I_{OH})	V_{CC} = 6.0V, V_{OH} = 2.0V	−3.0	−25	mA
TRI-STATE Configuration,				
L_0–L_7 Outputs, Low	V_{CC} = 6.3V, V_{OH} = 3.2V	−0.8		mA
Current Driver Option (I_{OH})	V_{CC} = 4.5V, V_{OH} = 1.5V	−0.9		mA
TRI-STATE Configuration,				
L_0–L_7 Outputs, High	V_{CC} = 6.3V, V_{OH} = 3.2V	−1.6		mA
Current Driver Option (I_{OH})	V_{CC} = 4.5V, V_{OH} = 1.5V	−1.8		mA
Input Load Source Current	V_{CC} = 5.0V	−10	−140	μA
CKO Output				
RAM Power Supply Option				
Power Requirement	V_R = 3.3V		3.0	mA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3-L0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

COP344L/COP345L

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	−40°C to +85°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C
Power Dissipation	0.75 Watt at 25°C 0.25 Watt at 85°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		15	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})	V_{CC} = Max.	3.0		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.2	0.3	V
Logic Low (V_{IL})		−0.3		
Schmitt Trigger Input				
Logic High (V_{IH})		0.7 V_{CC}		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.4	V
SO Input Level (Test Mode)		2.2	2.5	V
All Other Inputs				
Logic High	V_{CC} = Max.	3.0		V
Logic High	With TTL Trip Level Options	2.2		V
Logic Low	Selected, V_{CC} = 5V ±5%	−0.3	0.6	V
Logic High	With High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Hi-Z Input Leakage		−2	+2	μA
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −20 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
CMOS Operation (Note 2)				
Logic High	I_{OH} = −10 μA	V_{CC}−1		V
Logic Low	I_{OL} = +10 μA		0.2	V

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 2: TRI-STATE and LED configurations are excluded.

Note 3: SO output "0" level must be less than 0.6V for normal operation.

COP344L/COP345L (Continued)

DC Electrical Characteristics

$-40°C \leq T_A \leq +85°C$, $4.5V \leq V_{CC} \leq 5.5V$ unless otherwise noted. (Continued)

Parameter	Conditions	Min	Max	Units
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	$V_{CC} = 5.5V, V_{OL} = 0.4V$	1.0		mA
	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.8		mA
L_0–L_7 Outputs, and Standard	$V_{CC} = 5.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3, D_0–D_3 Outputs (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.4		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 5.5V, V_{OL} = 1.0V$	9		mA
High Current Options (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 1.0V$	7		mA
G_0–G_3 and D_0–D_3 Outputs with	$V_{CC} = 5.5V, V_{OL} = 1.0V$	18		mA
Very High Current Options (I_{OL})	$V_{CC} = 4.5V, V_{OL} = 1.0V$	14		mA
CKI (Single-Pin RC Oscillator)	$V_{CC} = 4.5V, V_{IH} = 3.5V$	2		mA
CKO	$V_{CC} = 4.5V, V_{OL} = 0.4V$	0.2		mA
Output Source Current				
Standard Configuration,	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−55	−600	µA
All Outputs (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−28	−350	µA
Push-Pull Configuration	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.1		mA
SO and SK Outputs (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
LED Configuration, L_0–L_7				
Outputs, Low Current	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−1.4	−17	mA
Driver Option(I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−0.7	−15	mA
LED Configuration, L_0–L_7				
Outputs, High Current	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−2.7	−34	mA
Driver Option (I_{OH})	$V_{CC} = 5.5V, V_{OH} = 2.0V$	−1.4	−30	mA
TRI-STATE Configuration,				
L_0–L_7 Outputs, Low	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−0.6		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−0.9		mA
TRI-STATE Configuration,				
L_0–L_7 Outputs, High	$V_{CC} = 5.5V, V_{OH} = 2.7V$	−1.2		mA
Current Driver Option (I_{OH})	$V_{CC} = 4.5V, V_{OH} = 1.5V$	−1.8		mA
Input Load Source Current	$V_{CC} = 5.0V$	−10	−200	µA
CKO Output				
RAM Power Supply Option	$V_R = 3.3V$		4.0	mA
Power Requirement				
TRI-STATE Output Leakage Current		−5	+5	µA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.5	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics

COP444L/445L: 0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted.

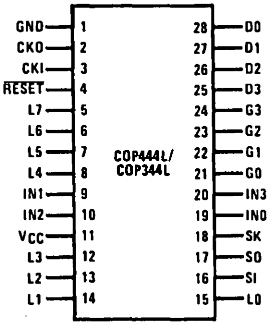
COP344L/345L: −40°C ≤ T_A ≤ +85°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time—t_C		16	40	μs
CKI				
Input Frequency—f_I	÷32 Mode	0.8	2.0	MHz
	÷16 Mode	0.4	1.0	MHz
	÷8 Mode	0.2	0.5	MHz
	÷4 Mode	0.1	0.25	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 2 MHz		120	ns
Fall Time			80	ns
CKI Using RC (÷4)	R = 56 kΩ ±5%			
	C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
CKO as SYNC Input				
t_{SYNC}		400		ns
INPUTS:				
IN_3–IN_0, G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
OUTPUT PROPAGATION DELAY	Test Condition:			
	C_L = 50 pF, R_L = 20 kΩ, V_{OUT} = 1.5V			
SO, SK Outputs				
t_{pd1}, t_{pd0}			4.0	μs
All Other Outputs				
t_{pd1}, t_{pd0}			5.6	μs

Note 1: Variation due to the device included.

Connection Diagrams

Dual-In-Line

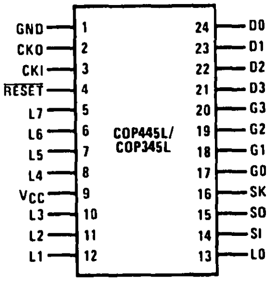


TL/DD/6928-2

Top View

Order Number COP444L-XXX/N or COP344L-XXX/N
See NS Package Number N28B

Dual-In-Line



TL/DD/6928-3

Top View

Order Number COP445L-XXX/N or COP345L-XXX/N
See NS Package Number N24A

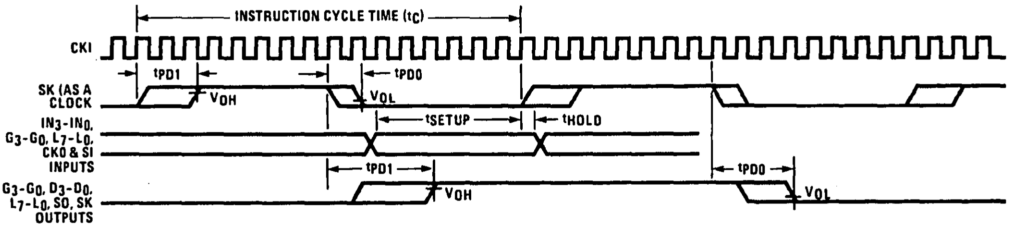
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidriectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs (COP444L only)
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)

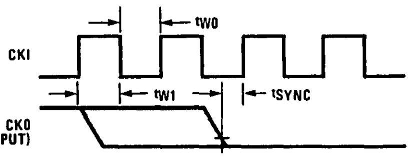
Pin	Description
CKI	System oscillator input
CKO	System oscillator output (or general purpose input, RAM power supply, or SYNC input)
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground

Timing Diagrams


INSTRUCTION CYCLE TIME (tC)
CKI
tPD1
tPD0
SK (AS A CLOCK
VOH
VOL
tSETUP
tHOLD
IN3-IN0, G3-G0, L7-L0, CKO & SI INPUTS
tPD1
tPD0
G3-G0, D3-D0, L7-L0, SO, SK OUTPUTS
VOH
VOL

TL/DD/6928-4

FIGURE 3a. Input/Output Timing Diagrams (Crystal Divide-by-16 Mode)



TL/DD/6928-5

FIGURE 3b. Synchronization Timing

Functional Description

A block diagram of the COP444L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

All functional references to the COP444L/COP445L also apply to the COP344L/COP345L.

PROGRAM MEMORY

Program Memory consists of a 2048 byte ROM. As can be seen by an examination of the COP444L/445L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID, and LQID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by a 11-bit PC register. Its binary value selects one of the 2048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value. Three levels of subroutine nesting are implemented by the 11-bit subroutine save registers, SA, SB, and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register descriptor, below.)

Four general-purpose inputs, IN_3–IN_0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". The table below provides a summary of the modes associated with EN_3 and EN_0.

Functional Description (Continued)

Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:

1. EN_1 has been set.
2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
3. A currently executing instruction has been completed
4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

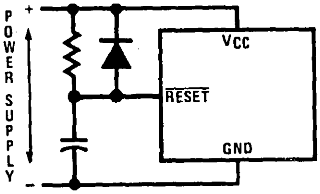
c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user use provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below. If the RC network is not used, the $\overline{\text{RESET}}$ pin must be pulled up to V_{CC} either by the internal load or by an external resistor (≥40 kΩ) to V_{CC}. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/6928-6

RC ≥ 5 x Power Supply Rise Time (R ≥ 40k)

Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instuction at address 0 must be a CLRA.

OSCILLATOR

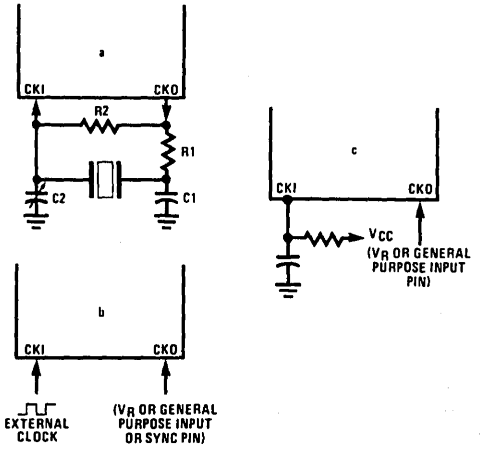
There are four basic clock oscillator configurations available as shown by *Figure 4*.

a. **Crystal Controlled Oscillator.** CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32 (optional by 16 or 8).

b. **External Oscillator.** CKI is an external clock input signal. The external frequency is divided by 32 (optional by 16 or 8) to give the instruction cycle time. CKO is now available to be used as the RAM power supply (V_R), as a general purpose input.

c. **RC Controlled Oscillator.** CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO is available as the RAM power supply (V_R) or as a general purpose input.

Functional Description (Continued)


a
CKI
CKO
R2
R1
C2
C1
b
CKI
CKO
EXTERNAL CLOCK
(VR OR GENERAL PURPOSE INPUT OR SYNC PIN)
c
CKI
CKO
VCC
(VR OR GENERAL PURPOSE INPUT PIN)

TL/DD/6928-7

Crystal Oscillator

Crystal Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220
2.097 MHz	1k	1M	30	6–36

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (μs)
51	100	19 ± 15%
82	56	19 ± 13%

NOTE: 200 kΩ ≥ R ≥ 25 kΩ

360 pF ≥ C ≥ 50 pF

FIGURE 4. COP444L/445L Oscillator

CKO PIN OPTIONS

In a crystal controlled oscillator system, CKO is used as an output to the crystal network. As an option CKO can be a general purpose input, read into bit 2 of A (accumulator) upon execution of an INIL instruction. As another option, CKO can be a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. Using either option is appropriate in applications where the COP444L/445L system timing configuration does not require use of the CKO pin.

I/O OPTIONS

COP444L/445L outputs have the following optional configurations, illustrated in *Figure 5.*

a. Standard—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. Available on SO, SK, and all D and G outputs.

b. Open-Drain—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. Available on SO, SK, and all D and G outputs.

c. Push-Pull—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. Available on SO and SK outputs only.

d. Standard L—same as **a.**, but may be disabled. Available on L outputs only.

e. Open Drain L—same as **b.**, but may be disabled. Available on L outputs only.

f. LED Direct Drive—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high impedance state to provide required LED segment blanking for a multiplexed display. Available on L outputs only.

Note: Series current limiting resistors have to be used if the higher operating voltage option is selected and LEDs are driven directly.

g. TRI-STATE Push-Pull—an enhancement-mode device to ground and V_{CC}. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers. Available on L outputs only.

COP444L/COP445L inputs have the following optional configurations:

h. An on-chip depletion load device to V_{CC}.

i. A Hi-Z input which must be driven to a "1" or "0" by external components.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT} curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

The SO, SK outputs can be configured as shown in **a., b.,** or **c.** The D and G outputs can be configured as shown in **a.** or **b.** Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs can be configured in **d., e., f.** or **g.**

An important point to remember if using configuration **d.** or **f.** with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to logic "1".

RAM KEEP-ALIVE OPTION

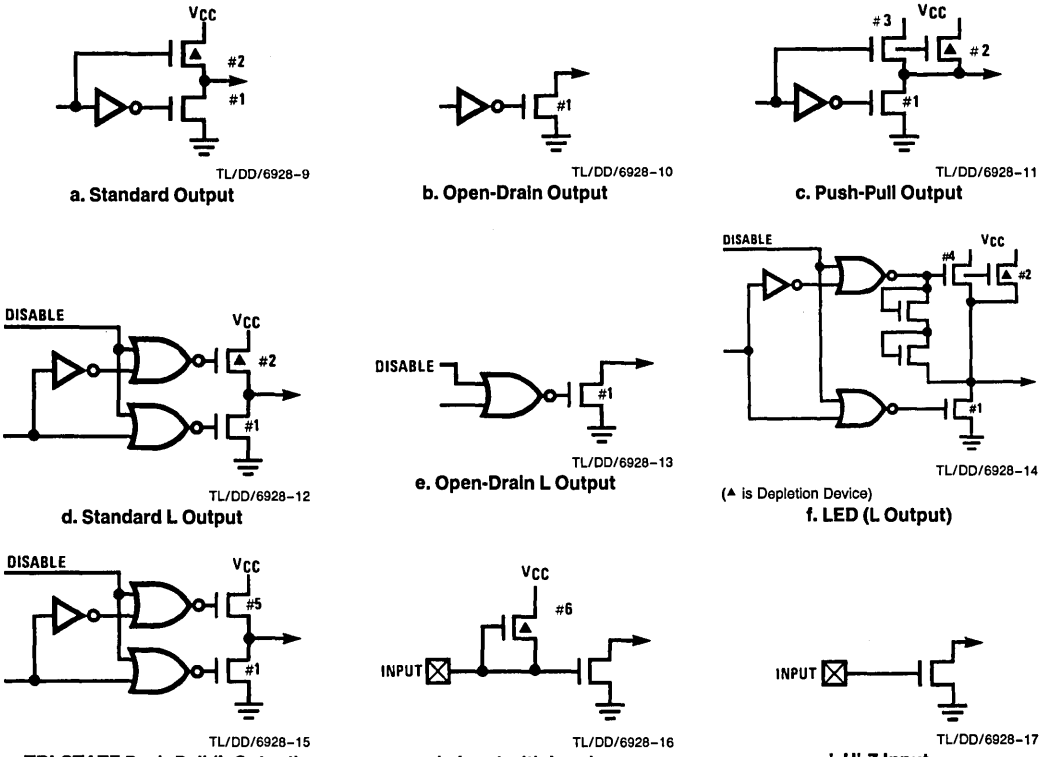
Selecting CKO as the RAM power supply (V_R) allows the user to shut off the chip power supply (V_{CC}) and maintain data in the lower four (Br = 0, 1, 2, 3) registers of RAM. To insure that RAM data integrity is maintained, the following conditions *must* be met:

1. $\overline{RESET}$ must go low before V_{CC} goes low during power off; V_{CC} must go high before $\overline{RESET}$ goes high on power-up.
2. V_R must be within the operating range of the chip, and equal to V_{CC} ±1V during normal operation.
3. V_R must be ≥ 3.3V with V_{CC} off.

Functional Description (Continued)

COP445L

If the COP444L is bonded as a 24-pin device, it becomes the COP455L, illustrated in *Figure 2*, COP444L/445L Connection Diagrams. Note that the COP445L does not contain the four general purpose IN inputs (IN_3–IN_0). Use of this option precludes, of course, use of the IN options and the interrupt feature, which uses IN_1. All other options are available for the COP445L.



TL/DD/6928-9
a. Standard Output

TL/DD/6928-10
b. Open-Drain Output

TL/DD/6928-11
c. Push-Pull Output

TL/DD/6928-12
d. Standard L Output

TL/DD/6928-13
e. Open-Drain L Output

TL/DD/6928-14
(▲ is Depletion Device)
f. LED (L Output)

TL/DD/6928-15
g. TRI-STATE Push-Pull (L Output)

TL/DD/6928-16
h. Input with Load

TL/DD/6928-17
i. Hi-Z Input

FIGURE 5. Output Configuration

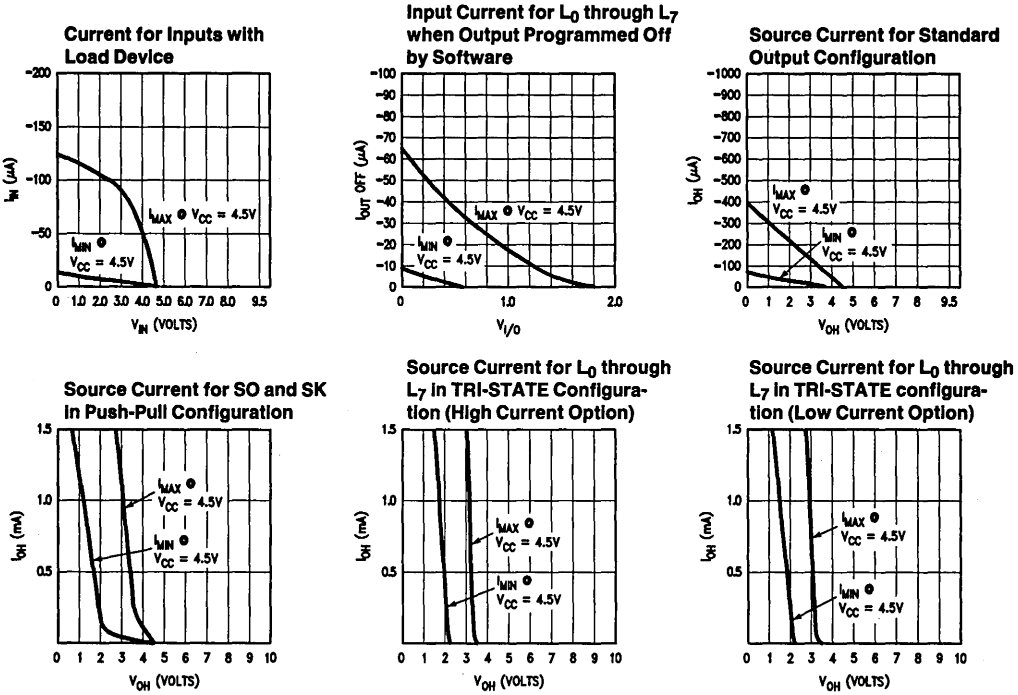
L-Bus Considerations

False states may be generated on L_0–L_7 during the execution of the CAMQ instruction. The L-Ports should not be used as clocks for edge sensitive devices such as flip-flops, counters, shift registers, etc. The following short program illustrates this situation.

```
START:
        CLRA              ;ENABLE THE Q
        LEI   4           ;REGISTER TO L LINES
        LBI   TEST
        STII  3
        AISC  12
LOOP:
        LBI   TEST        ;LOAD Q WITH X'C3
        CAMQ
        JP    LOOP
```

In this program the internal Q register is enabled onto the L lines and a steady bit pattern of logic highs is output on L_0, L_1, L_6, L_7, and logic lows on L_2–L_5 via the two-byte CAMQ instruction. Timing constraints on the device are such that the Q register may be temporarily loaded with the second byte of the CAMQ opcode (X'3C) prior to receiving the valid data pattern. If this occurs, the opcode will ripple onto the L lines and cause negative-going glitches on L_0, L_1, L_6, L_7, and positive glitches on L_2–L_5. Glitch durations are under 2 μs, although the exact value may vary due to data patterns, processing parameters, and the L line loading. These false states are peculiar only to the CAMQ instruction and the L lines.

Typical Performance Characteristics



Current for Inputs with Load Device

Input Current for L_0 through L_7 when Output Programmed Off by Software

Source Current for Standard Output Configuration

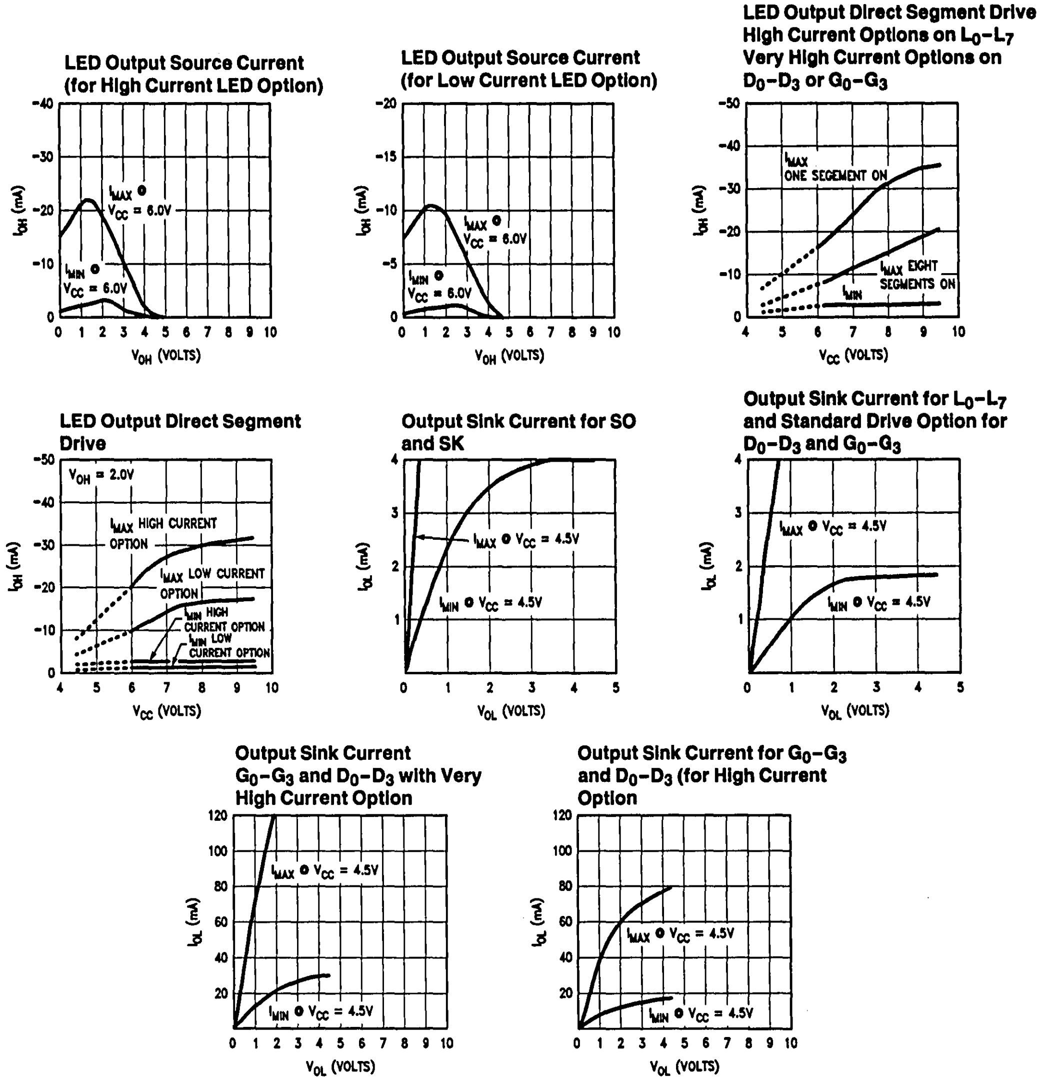
Source Current for SO and SK in Push-Pull Configuration

Source Current for L_0 through L_7 in TRI-STATE Configuration (High Current Option)

Source Current for L_0 through L_7 in TRI-STATE configuration (Low Current Option)

TL/DD/6928-18

Typical Performance Characteristics (Continued)


LED Output Source Current (for High Current LED Option)

LED Output Source Current (for Low Current LED Option)

LED Output Direct Segment Drive High Current Options on L_0–L_7 Very High Current Options on D_0–D_3 or G_0–G_3

LED Output Direct Segment Drive

Output Sink Current for SO and SK

Output Sink Current for L_0–L_7 and Standard Drive Option for D_0–D_3 and G_0–G_3

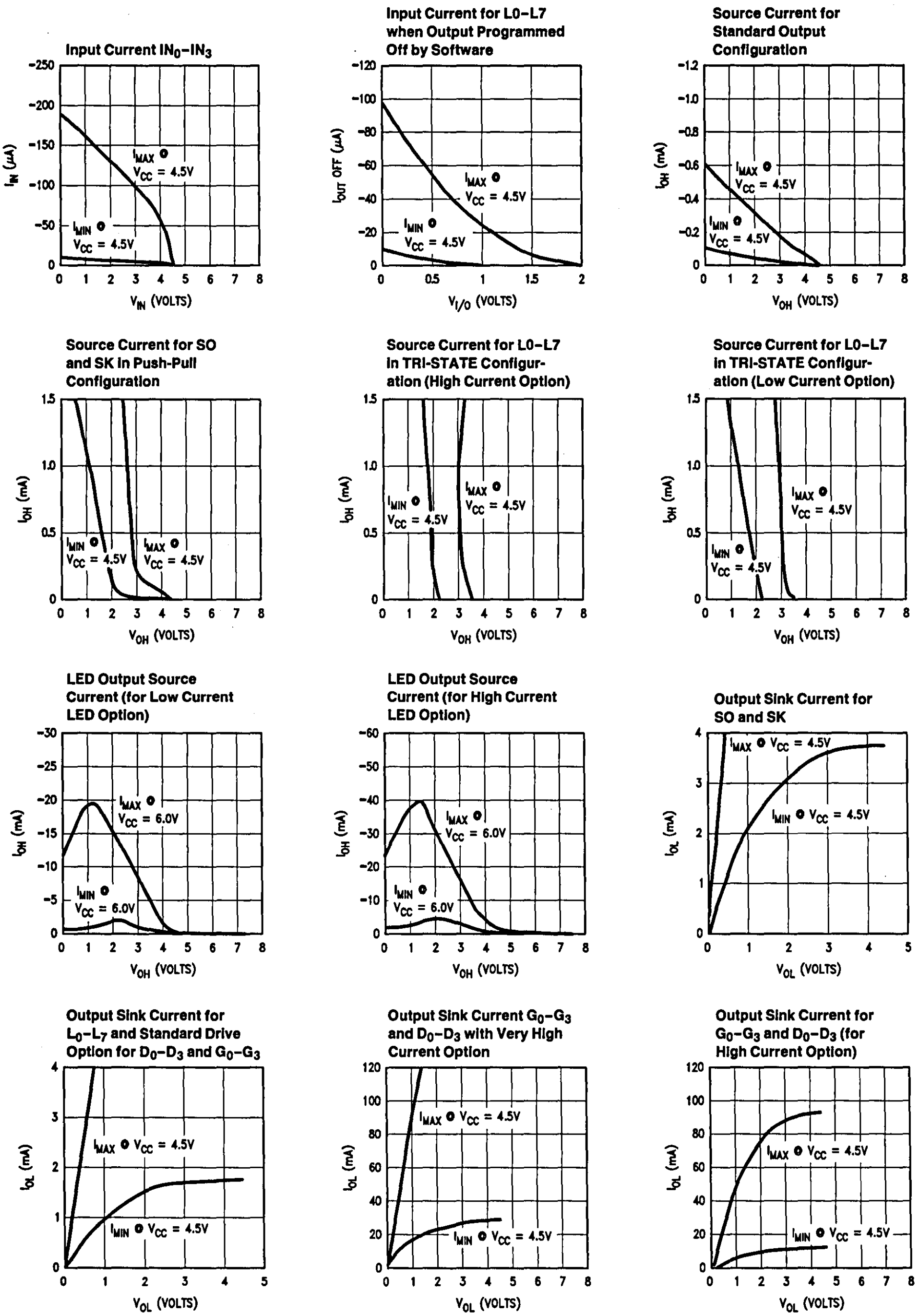
Output Sink Current G_0–G_3 and D_0–D_3 with Very High Current Option

Output Sink Current for G_0–G_3 and D_0–D_3 (for High Current Option)

TL/DD/6928-19

FIGURE 6a. COP444L/COP445L Input/Output Characteristics

Typical Performance Characteristics (Continued)



TL/DD/6928-20

FIGURE 6b. COP344L/COP345L Input/Output Characteristics

COP444L/COP445L/COP344L/COP345L Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP444L/445L instruction set.

TABLE I. COP444L/445L/344L/345L Instruction Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	11-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	3-bit Operand Field, 0-7 binary (RAM Register Select)
a	11-bit Operand Field, 0-2047 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE II. COP444L/445L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
ADT		4A	0100 1010	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	0101 y	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	0001 0000	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
JID		FF	1111 1111	ROM ($PC_{10:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6- --	0110 0 $a_{10:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	--	1 $a_{6:0}$ (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 4)
		--	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	--	10 $a_{5:0}$	PC + 1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6- --	0110 1 $a_{10:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	-5	00 r 0101 (r = 0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A Exclusive-OR Br with r
LDD	r,d	23 --	0010 0011 0 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 1101 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7-	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	-6	00 r 0110 (r = 0:3)	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 0011 1 r d	RAM(r,d) ←→ A	None	Exchange A with RAM pointed to directly by r,d

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	00 r 0111 (r = 0:3)	RAM(B) ←→ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	00 r 0100 (r = 0:3)	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	- -	00 r (d - 1) (r = 0:3; d = 0, 9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33	0011 0011			
		- -	1 r d any r, any d)			
LEI	y	33	0001 0011	y → EN	None	Load EN Immediate (Note 7)
		6-	0110 y			
XABR		12	0001 0010	A ←→ Br (0 → A_3)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
		21	0010 0001			
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0 = 0$	
	1	11	0001 0001		$G_1 = 0$	
	2	03	0000 0011		$G_2 = 0$	
	3	13	0001 0011		$G_3 = 0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1 = 0$	
	2	03	0000 0011		$RAM(B)_2 = 0$	
	3	13	0001 0011		$RAM(B)_3 = 0$	
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
ININ		33	0011 0011	IN → A	None	Input IN Inputs to A (Note 2)
		28	0010 1000			
INIL		33	0011 0011	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
		29A	0010 1001			

Instruction Set (Continued)

TABLE II. COP444L/445L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS (Continued)						
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5-	0011 0011 0101 t	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: The ININ instruction is not available on the 24-pin COP445L or COP345L since these devices do not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14 or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP444L/445L programs.

SOFTWARE AND OPCODE DIFFERENCES IN THE COP444L INSTRUCTION SET

The COP444L is essentially a COP420L with a double RAM and ROM. Because of this increased memory space certain instructions have expanded capability in the COP444L. Note that there are no new instructions in the COP444L and that all instructions perform the same operations in the COP444L as they did in the COP420L. The expanded capability is merely to allow appropriate handling of the increased memory space. The affected instructions are:

JMP	a	(a = address)
JSR	a	(a = address)
LDD	r,d	(r,d = RAM address Br,Bd)
XAD	r,d	(r,d = RAM address Br,Bd)
LBI	r,d	(r,d = RAM address Br,Bd; only two byte form of the instruction affected)
XABR		

The JMP and JSR instructions are modified in that the address a may be anywhere within the 2048 words of ROM space. The opcodes are as follows:

JMP	0110 0 $a_{10:9:8}$ $a_{7:0}$	JSR	0110 1 $a_{10:9:8}$ $a_{7:0}$

The LDD, XAD, and two byte LBI are modified so that they may address the entire RAM space. The opcodes are as follows:

LDD	0110 0011 0 r d	XAD	0010 0011 1 r d
LBI	0011 0011 1 r d		

The XABR instruction change is transparent to the user. The opcode is not changed nor is the function of the instruction. The change is that values of 0 through 7 in A will address registers in the COP444L (i.e., the lower three bits of A become the Br value following the instruction). In the COP420L, the lower two bits of A became the Br value following an XABR instruction.

Note that those instructions which have an exclusive-or argument (LD, X, XIS, XDS) are not affected. The argument is still two bits of the opcode. This means that the exclusive-or aspect of these instructions works within blocks of four registers. It is not possible to toggle Br from a value between 0 and 3 to a value between 4 and 7 by means of these instructions.

Description of Selected Instructions (Continued)

There are no other software or opcode differences between the COP444L and the COP420L. Examination of the above changes indicates that the existing opcodes for those instructions have merely been extended. There is no fundamental change.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, $PC_{10:8}$, A, M. PC_{10}, PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

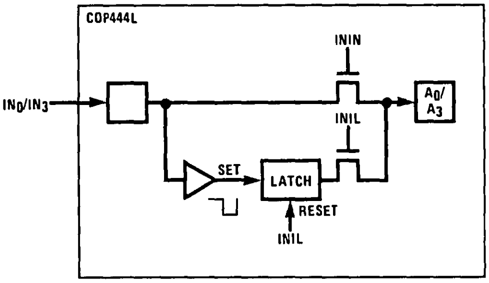
INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 7*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred or the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset; IL_3–IL_0 *not* input on 445L

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC_{10}, PC_9, PC_8 A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_{10}, PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.



TL/DD/6928-21

FIGURE 7. INIL Hardware Implementation

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP444L/445L to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

Description of Selected Instructions (Continued)

INSTRUCTION SET NOTES

a. The first word of a COP444L/445L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last work of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23 or 27 will access data in the next group of four pages.

Option List

The COP444L/445L mask-programmable options are assigned numbers which correspond with the COP*444L* pins.

The following is a list of COP444L options. When specifying a COP445L chip, Options 9, 10, 19, and 20 must all be set to zero. The options are programmed at the same time as the ROM pattern to provide the user with the hardware flexibility to interface to various I/O components using little or no external circuitry.

Option 1 = 0: Ground Pin—no options available

Option 2: CKO Output
= 0: clock generator ouput to crystal/resonator
(0 not allowable value if option 3 = 3)
= 1: pin is RAM power supply (V_R) input
= 2: general purpose input, load device to V_{CC}
= 3: general purpose input, Hi-Z

Option 3: CKI Input
= 0: oscillator input divided by 32 (2 MHz max.)
= 1: oscillator input divided by 16 (1 MHz max.)
= 2: oscillator input divided by 8 (500 kHz max.)
= 3: single-pin RC controlled oscillator divided by 4
= 4: oscillator input divided by 4 (Schmitt)

Option 4: $\overline{\text{RESET}}$ Input
= 0: load device to V_{CC}
= 1: Hi-Z input

Option 5: L_7 Driver
= 0: Standard output
= 1: Open-drain output
= 2: High current LED direct segment drive output
= 3: High current TRI-STATE push-pull output
= 4: Low-current LED direct segment drive output
= 5: Low-current TRI-STATE push-pull output

Option 6: L_6 Driver
same as Option 5

Option 7: L_5 Driver
same as Option 5

Option 8: L_4 Driver
same as Option 5

Option 9: IN_1 Input
= 0: load device to V_{CC}
= 1: Hi-Z input

Option 10: IN_2 Input
same as Option 9

Option 11: V_{CC} pin Operating Voltage

	COP44XL	COP34XL
= 0:	+4.5V to +6.3V	+4.5V to +5.5V

Option 12: L_3 Driver
same as Option 5

Option 13: L_2 Driver
same as Option 5

Option 14: L_1 Driver
same as Option 5

Option 15: L_0 Driver
same as Option 5

Option 16: SI Input
same as Option 9

Option List (Continued)

Option 17: SO Driver
= 0: standard output
= 1: open-drain output
= 2: push-pull output

Option 18: SK Driver
same as Option 17

Option 19: IN_0 Input
same as Option 9

Option 20: IN_3 Input
same as Option 9

Option 21: G_0 I/O Port
= 0: very-high current standard output
= 1: very-high current open-drain output
= 2: high current standard output
= 3: high current open-drain output
= 4: standard LSTTL output (fanout = 1)
= 5: open-drain LSTTL output (fanout = 1)

Option 22: G_1 I/O Port
same as Option 21

Option 23: G_2 I/O Port
same as Option 21

Option 24: G_3 I/O Port
same as Option 21

Option 25: D_3 Output
same as Option 21

Option 26: D_2 Output
same as Option 21

Option 27: D_1 Output
same as Option 21

Option 28: D_0 Output
same as Option 21

Option 29: L Input Levels
= 0: standard TTL input levels
("0" = 0.8V, "1" = 2.0V)
= 1: higher voltage input levels
("0" = 1.2V, "1" = 3.6V)

Option 30: IN Input Levels
same as Option 29

Option 31: G Input Levels
same as Option 29

Option 32: SI Input Levels
same as Option 29

Option 33: RESET Input
= 0: Schmitt trigger input levels
= 1: standard TTL input levels
= 2: higher voltage input levels

Option 34: CKO Input Levels (CKO = input; Option 2 = 2, 3)
same as Option 29

Option 35: COP Bonding
= 0: COP444L (28-pin device)
= 1: COP445L (24-pin device)
= 2: both 28- and 24-pin versions

Option 36: Internal Initialization Logic
= 0: normal operation
= 1: no internal initialization logic

COP444L Option Table

The following option information is to be sent to National along with the EPROM.

OPTION DATA

OPTION 1 VALUE= ____0____ IS: GROUND PIN
OPTION 2 VALUE= ________ IS: CKO PIN
OPTION 3 VALUE= ________ IS: CKI PIN
OPTION 4 VALUE= ________ IS: RESET INPUT
OPTION 5 VALUE= ________ IS: L(7) DRIVER
OPTION 6 VALUE= ________ IS: L(6) DRIVER
OPTION 7 VALUE= ________ IS: L(5) DRIVER
OPTION 8 VALUE= ________ IS: L(4) DRIVER
OPTION 9 VALUE= ________ IS: IN1 INPUT
OPTION 10 VALUE= ________ IS: IN2 INPUT
OPTION 11 VALUE= ____0____ IS: VCC PIN
OPTION 12 VALUE= ________ IS: L(3) DRIVER
OPTION 13 VALUE= ________ IS: L(2) DRIVER
OPTION 14 VALUE= ________ IS: L(1) DRIVER
OPTION 15 VALUE= ________ IS: L(0) DRIVER
OPTION 16 VALUE= ________ IS: SI INPUT
OPTION 17 VALUE= ________ IS: SO DRIVER
OPTION 18 VALUE= ________ IS: SK DRIVER
OPTION 19 VALUE= ________ IS: IN0 INPUT
OPTION 20 VALUE= ________ IS: IN3 INPUT

OPTION DATA

OPTION 21 VALUE= ________ IS: G0 I/O PORT
OPTION 22 VALUE= ________ IS: G1 I/O PORT
OPTION 23 VALUE= ________ IS: G2 I/O PORT
OPTION 24 VALUE= ________ IS: G3 I/O PORT
OPTION 25 VALUE= ________ IS: D3 OUTPUT
OPTION 26 VALUE= ________ IS: D2 OUTPUT
OPTION 27 VALUE= ________ IS: D1 OUTPUT
OPTION 28 VALUE= ________ IS: D0 OUTPUT
OPTION 29 VALUE= ________ IS: L INPUT LEVELS
OPTION 30 VALUE= ________ IS: IN INPUT LEVELS
OPTION 31 VALUE= ________ IS: G INPUT LEVELS
OPTION 32 VALUE= ________ IS: SI INPUT LEVELS
OPTION 33 VALUE= ________ IS: RESET INPUT
OPTION 34 VALUE= ________ IS: CKO INPUT LEVELS
OPTION 35 VALUE= ________ IS: COP BONDING
OPTION 36 VALUE= ________ IS: INTERNAL INITIALIZATION LOGIC

Typical Applications

TEST MODE (NON-STANDARD OPERATION)

The SO output has been configured to provide for standard test procedures for the custom-programmed COP444L. With SO forced to logic "1", two test modes are provided, depending upon the value of SI:

a. RAM and Internal Logic Test Mode (SI = 1)

b. ROM Test Mode (SI = 0)

These special test modes should not be employed by the user; they are intended for manufacturing test only.

APPLICATION #1: COP444L GENERAL CONTROLLER

Figure 8 shows an interconnect diagram for a COP444L used as a general controller. Operation of the system is as follows:

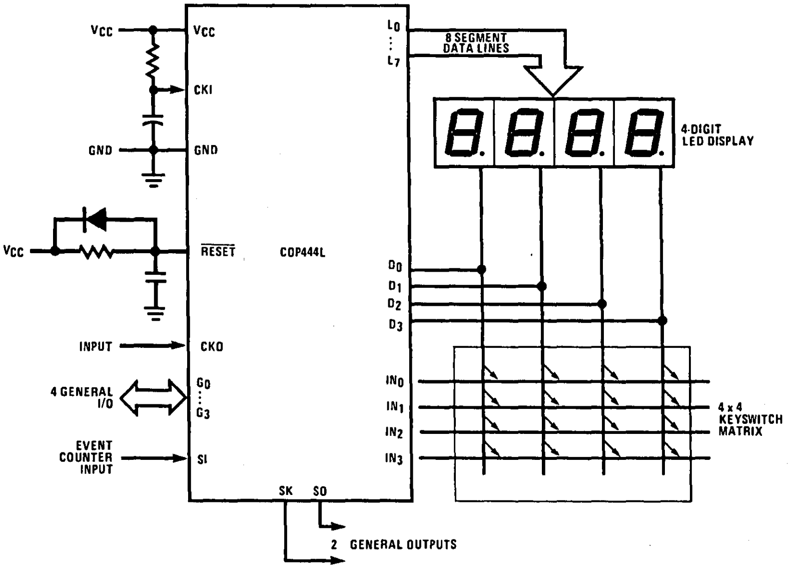
1. The L_7–L_0 outputs are configured as LED Direct Drive outputs, allowing direct connection to the segments of the display
2. The D_3–D_0 outputs drive the digits of the multiplexed display directly and scan the columns of the 4 x 4 keyboard matrix.
3. The IN_3–IN_0 inputs are used to input the 4 rows of the keyboard matrix. Reading the IN lines in conjunction with the current value of the D outputs allows detection, debouncing, and decoding of any one of the 16 keyswitches.
4. CKI is configured as a single-pin oscillator input allowing system timing to be controlled by a single-pin RC network. CKO is therefore available for use as a general-purpose input.
5. SI is selected as the input to a binary counter input. With SIO used as a binary counter, SO and SK can be used as general purpose outputs.
6. The 4 bidirectional G I/O ports (G_3–G_0) are available for use as required by the user's application.
7. Normal reset operation is selected.

COP444L EVALUATION (See COP Note 4)

The 444L-EVAL is a pre-programmed COP444L, containing several routines which facilitate user familiarization and evaluation of the COP444L operating characteristics. It may be used as an up/down counter or timer, interfacing to any combination of (1) an LED digit or lamps, (2) 4-digit LED Display Controller, (3) a 4-digit VF Display Controller, and/or (4) a 4-digit LCD Display Controller. Alternatively, it may be used as a simple music synthesizer.

SAMPLE CIRCUITS

1. By making only the oscillator, power supply and "L7" connections, *(Figure 9)* an approximate 1 Hz square wave will be produced at output "D1." This output may be observed with an oscilloscope, or connected to additional TTL or CMOS circuitry.
2. By making the indicated connections to a small LED digit (NSA1541A, NSA1166, or equiv.—larger digits will be proportionately dimmer), the counter actions may be observed. Place the "up/down" switch in the "up" (open) position and apply a TTL-compatible signal at the "counter-input." Placing the "up/down" switch in the "down" (closed) position causes the count to decrement on each high-to-low input transition.
3. All 4 digits of the counter may be displayed by connecting a standard display controller (COP472 for LCD, MM5450 for LED) as shown in *Figure 9.*



TL/DD/6928-22

FIGURE 8. COP444L Keyboard/Display Interface

Typical Applications (Continued)

Any combination of the single LED digit and display controllers may be used simultaneously, and will display the same data.

4. The simple counter described above becomes a timer when the 1 Hz output is connected to the "counter input." Up or down counting may be used with input frequencies up to 1 kHz. Improved timing accuracies may be obtained by subsituting the 2.097 MHz crystal oscillator circuit of *Figure 4a* for the RC network shown in *Figure 9*, or by connecting a more stable external frequency to the "counter input" in place of the 1 Hz signal.

5. An "entertaining" use of the 444L-EVAL is as a simple music synthesizer (or electronic organ). By attaching a simple switch matrix (or keyboard), a speaker or piezo-ceramic transducer, and grounding "L7", the user can play "music" *(Figure 10)*. Three modes of operation are available: Play a note, play one of four stored tunes, or record a tune for subsequent replay.

   a. *Play A Note*

   Twelve keys, representing the 12 notes in one octave, are labeled "C" through "B"; depressing a key causes a square wave of the corresponding frequency to be outputted to the speaker. Depressing "LShift" or "UShift" causes the next note to be shifted to the next lower octave (one-half frequency) or the next upper octave (double frequency), respectively.

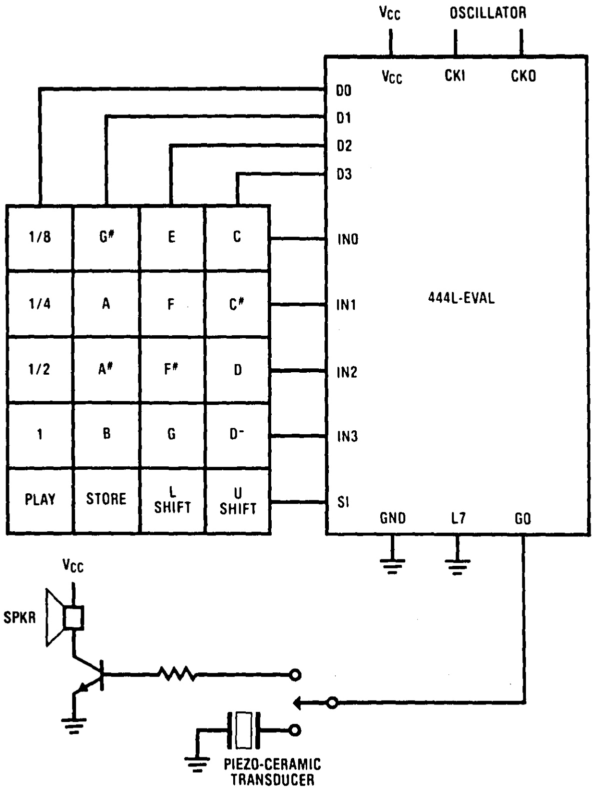
   b. *Play Stored Tune*

   Depressing "Play" followed by "1/8", "1/4", "1/2", or "1" will cause one of 4 stored tunes to be played.

   c. *Record Tune*

   Any combination of notes and rests up to a total of 48 may be stored in RAM for later replay. To store a note, press the appropriate note key, followed by the duration of the note (1/8-note, 1/4-note, 1/2-note, whole (1)-note, followed by "Store"; a rest is stored by selecting the duration and pressing "Store." When the tune is complete, press "Play" followed by "Store"; the tune will be played for immediate audition. Subsequent depression of "Play" and "Store" will replay the last stored tune.

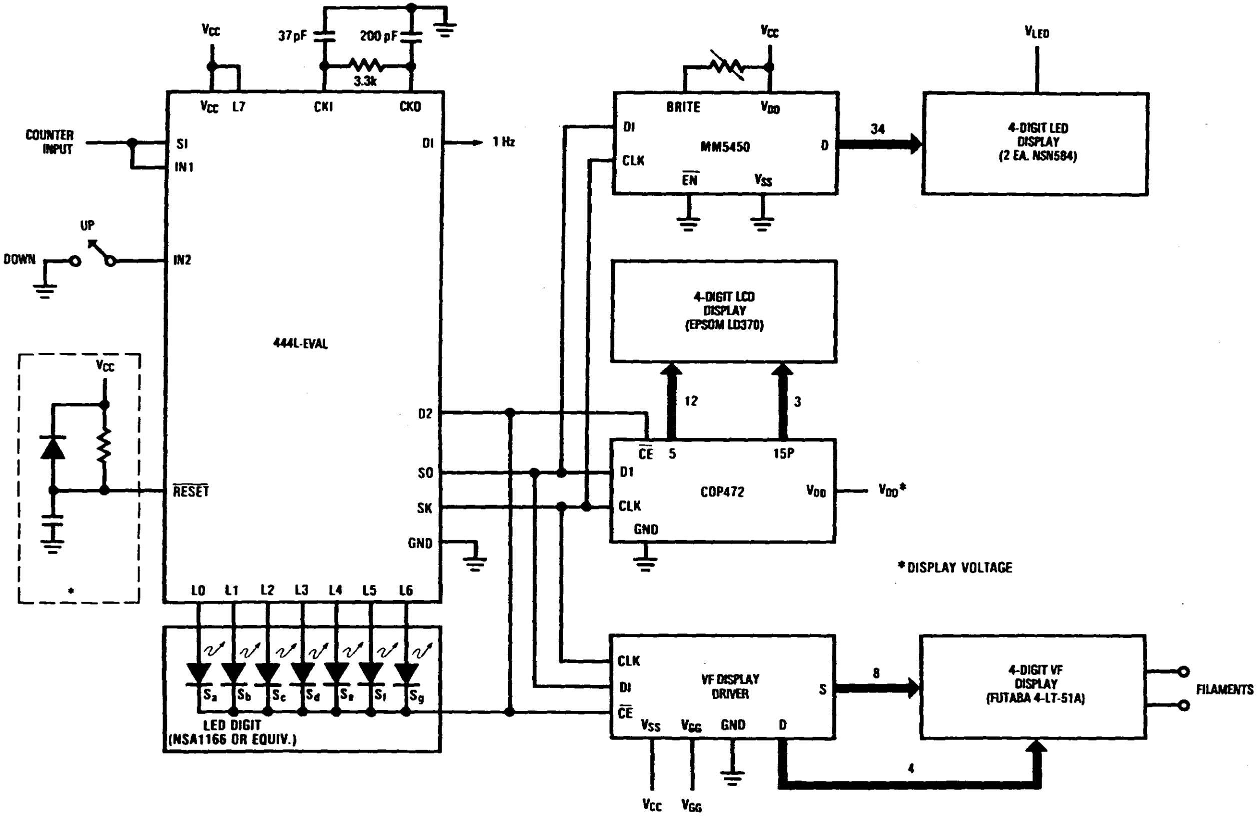
Note: The accuracy of the tones produced is a function of the oscillator accuracy and stability; the crystal oscillator is recommended.



TL/DD/6928-23

FIGURE 9. Counter/Timer

Typical Applications (Continued)



TL/DD/6928–24

* See "Initialization"

FIGURE 10. Music Synthesizer

National Semiconductor

COP401L ROMless N-Channel Microcontroller

General Description

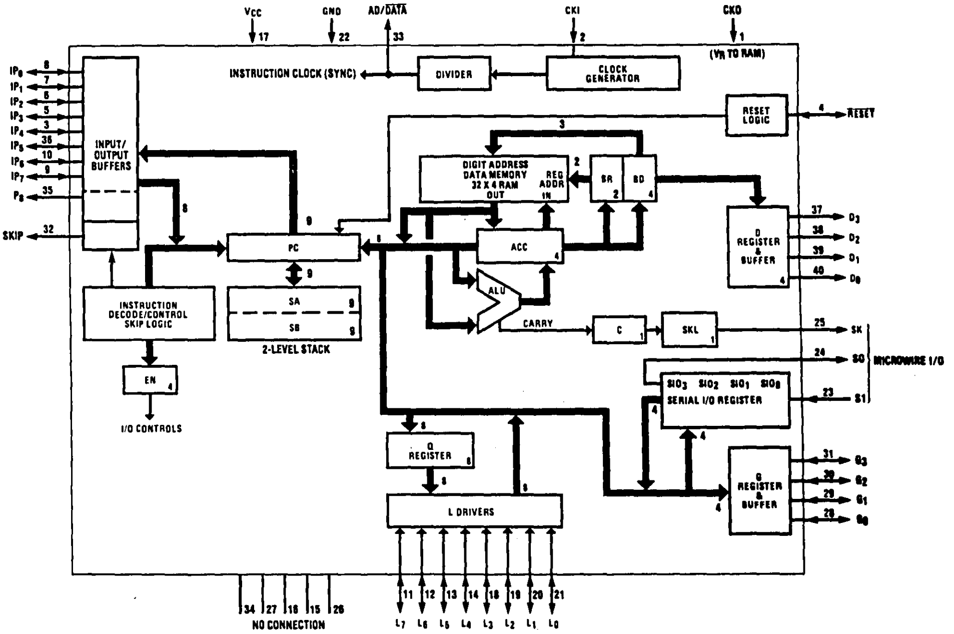
The COP401L ROMless Microcontroller is a member of the COPS™ family of microcontrollers, fabricated using N-channel, silicon gate MOS technology. The COP401L contains CPU, RAM, I/O and is identical to a COP410L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP401L will perform exactly as the COP410L. This important benefit facilitates development and debug of a COP program prior to masking the final part.

The COP401L is intended for emulation only, not intended for volume production. Use COP402 or COP404L for volume production.

Features

- Circuit equivalent of COP410L
- Low cost
- Powerful instruction set
- 512 x 8 ROM, 32 x 4 RAM
- Separate RAM power supply pin for RAM keep-alive applications
- Two-level subroutine stack
- 15 μs instruction time
- Single supply operation (4.5–9.5V)
- Low current drain (8 mA max.)
- Internal binary counter register with serial I/O
- MICROWIRE™ compatible serial I/O
- General purpose outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family
- Pin-for-pin compatible with COP402 and COP404L

Block Diagram



TL/DD/6913-1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	120 mA
Total Sink Current	120 mA

Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 9.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 2)	4.5	9.5	V
Power Supply Ripple	Peal to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})		2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
IP0–IP7 Input Levels				
Logic High	V_{CC} = 9.5V	2.4		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
All Other Inputs				
Logic High	V_{CC} = 9.5V	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
IP0–IP7, P8, SKIP	(Note 1)			
Logic Low	I_{OL} = 1.6 mA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{CC} = 9.5V, V_{OL} = 0.4V	1.8		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L_0–L_7 and G_0–G_3 Outputs	V_{CC} = 9.5V, V_{OL} = 0.4V	0.8		mA
	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
D_0–D_3 Outputs	V_{CC} = 9.5V, V_{OL} = 1.0V	30		mA
	V_{CC} = 4.5V, V_{OL} = 1.0V	15		mA
CKO				
RAM Power Supply Input	V_R = 3.3V		1.5	mA

DC Electrical Characteristics $0°C \le T_A \le +70°C, 4.5V \le V_{CC} \le 9.5V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Output Source Current				
D_0–D_3, G_0–G_3 Outputs (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−140	−800	μA
	$V_{CC} = 4.5V, V_{OH} = 2.0V$	−30	−250	μA
SO and SK Outputs (I_{OH})	$V_{CC} = 9.5V, V_{OH} = 4.75V$	−1.4		mA
	$V_{CC} = 4.5V, V_{OH} = 1.0V$	−1.2		mA
L_0–L_7 Outputs	$V_{CC} = 9.5V, V_{OH} = 2.0V$	−3.0	−35	mA
	$V_{CC} = 6.0V, V_{OH} = 2.0V$	−0.3	−25	mA
Input Load Source Current (I_{IL})	$V_{CC} = 5.0V, V_L = 0V$	−10	−140	μA
Total Sink Current Allowed				
All Outputs Combined			120	mA
D Port			100	mA
L_7–L_4, G Port			4	mA
L_3–L_0			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			25	mA
All Other Pins			1.5	mA

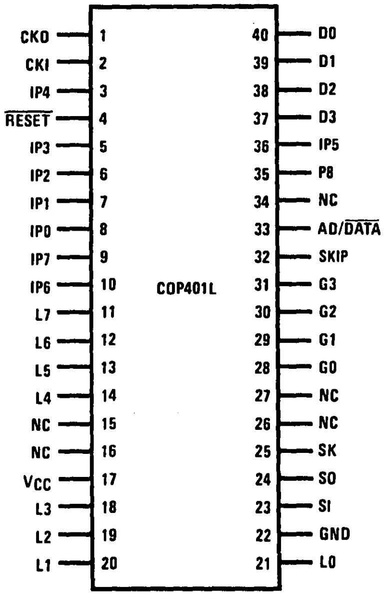
AC Electrical Characteristics $0°C \le T_A \le +70°C, 4.5V \le V_{CC} \le 9.5V$ unless otherwise specified.

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		15	40	μs
CKI				
Input Frequency f_I	(÷32 Mode)	0.8	2.1	MHz
Duty Cycle		30	60	%
Rise Time	$f_I = 2.097$ MHz		120	ns
Fall Time			80	ns
INPUTS:				
SI, IP7–IP0				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
OUTPUT PROPAGATION DELAY	Test Condition: C_L = pF, $V_{OUT} = 1.5V$			
SO, SK Outputs	$R_L = 20$ kΩ			
t_{pd1}, t_{pd0}			4.0	μs
D_3–D_0, G_3–G_0, L_7–L_0	R_L = kΩ			
t_{pd1}, t_{pd0}			5.6	μs
IP7–IP0, P8, SKIP	$R_L = 5$ kΩ			
t_{pd1}, t_{pd0}			7.2	μs

Note 1: Pull-up resistors required.

Note 2: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Connection Diagram



TL/DD/6913-2

Order Number COP401L/N
NS Package Number N40A

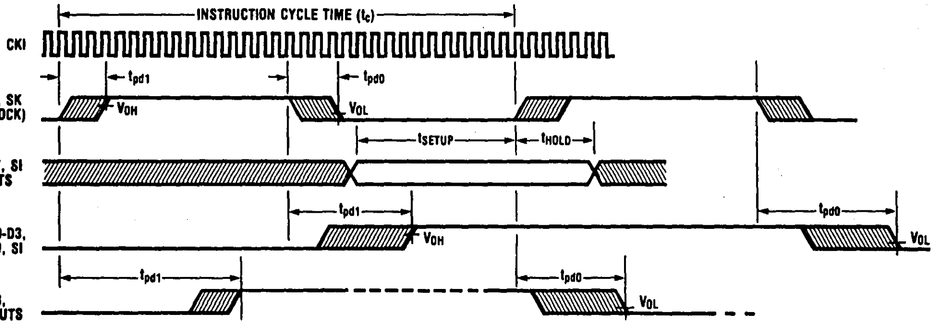
FIGURE 2

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports with LED segment drive
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{\text{DATA}}$	Address Out/data in flag

Pin	Description
CKI	System oscillator input
CKO	RAM power supply input
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8	Most significant ROM address bit output
SKIP	Instruction skip output

Timing Diagram



TL/DD/6913-3

FIGURE 3. Input/Output

Functional Description

A block diagram of the COP401L is given in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

PROGRAM MEMORY

Program Memory consists of a 512-byte external memory. As can be seen by an examination of the COP401L instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

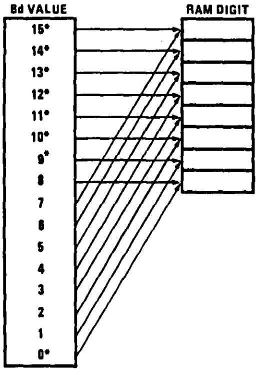
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3, 15 instruction. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table 3).



TL/DD/6913-4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP401L, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift register.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP401L operation.

Functional Description (Continued)

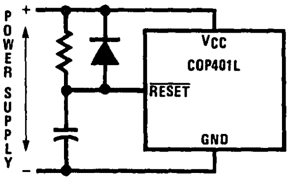
TABLE I. Enable Register Modes—Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table I provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below *(Figure 5)*. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/6913-5

RC ≥ Power Supply Rise Time

FIGURE 5. Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP401L is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE® outputs
3. TTL-compatible inputs
4. access time = 5 µs max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P8 and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P8 is a dedicated address output, and does not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

CKI is an external clock input signal. The external frequency is divided by 32 to give the instruction cycle time. The divide-by-32 configuration was chosen to make the COP 401L compatible with the COP404L and the COPS™ Development System. However, the ÷32 configuration is not available on the COP410L/COP411L. It is therefore possible to exactly emulate the system **speed** (cycle time), but **not** possible to drive the 401L with the system clock during emulation.

Functional Description (Continued)

CKO (RAM POWER)

CKO is configured as a RAM power supply pin (V_R), allowing its connection to a standby/backup power supply to maintain the integrity of RAM data with minimum power drain when the main supply is inoperative or shut down to conserve power. This pin must be connected to V_{CC} if the power backup feature is not used. To insure that RAM integrity is maintained, the following conditions must be met:

1. $\overline{\text{RESET}}$ must go low before V_{CC} goes below spec during power-off; V_{CC} must be within spec before $\overline{\text{RESET}}$ goes high on power-up.
2. During normal operation, V_R must be within the operating range of the chip with $(V_{CC}-1) \le V_R \le V_{CC}$.
3. V_R must be $\ge$ 3.3V with V_{CC} off.

INPUT/OUTPUT CONFIGURATIONS

COP401L outputs have the following configurations, illustrated in *Figure 6*:

a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. (Used on D and G outputs.)

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application. (Used on IP, P and SKIP outputs.)

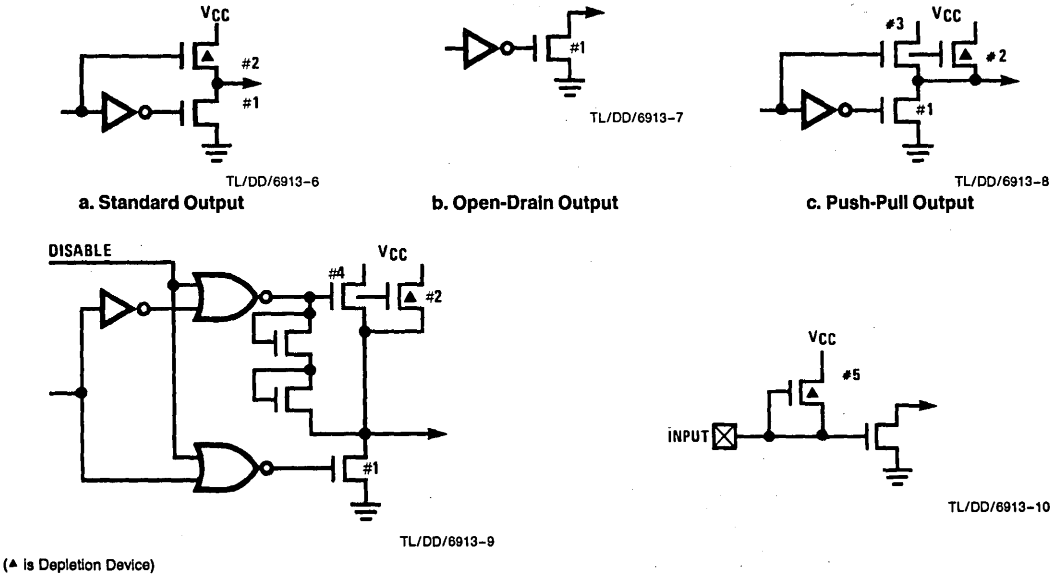
c. **Push-Pull**—An enhancement-mode device to ground in conjunction with a depletion-mode device paralleled enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads. (Used on SO and SK outputs.)

d. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (See Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. (Used on L outputs.)

COP401L inputs have an on-chip depletion load device to V_{CC}.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of five devices (numbered 1–5, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 7* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 7,* Device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to a logic "1".



TL/DD/6913-6

a. Standard Output

TL/DD/6913-7

b. Open-Drain Output

TL/DD/6913-8

c. Push-Pull Output

TL/DD/6913-9

TL/DD/6913-10

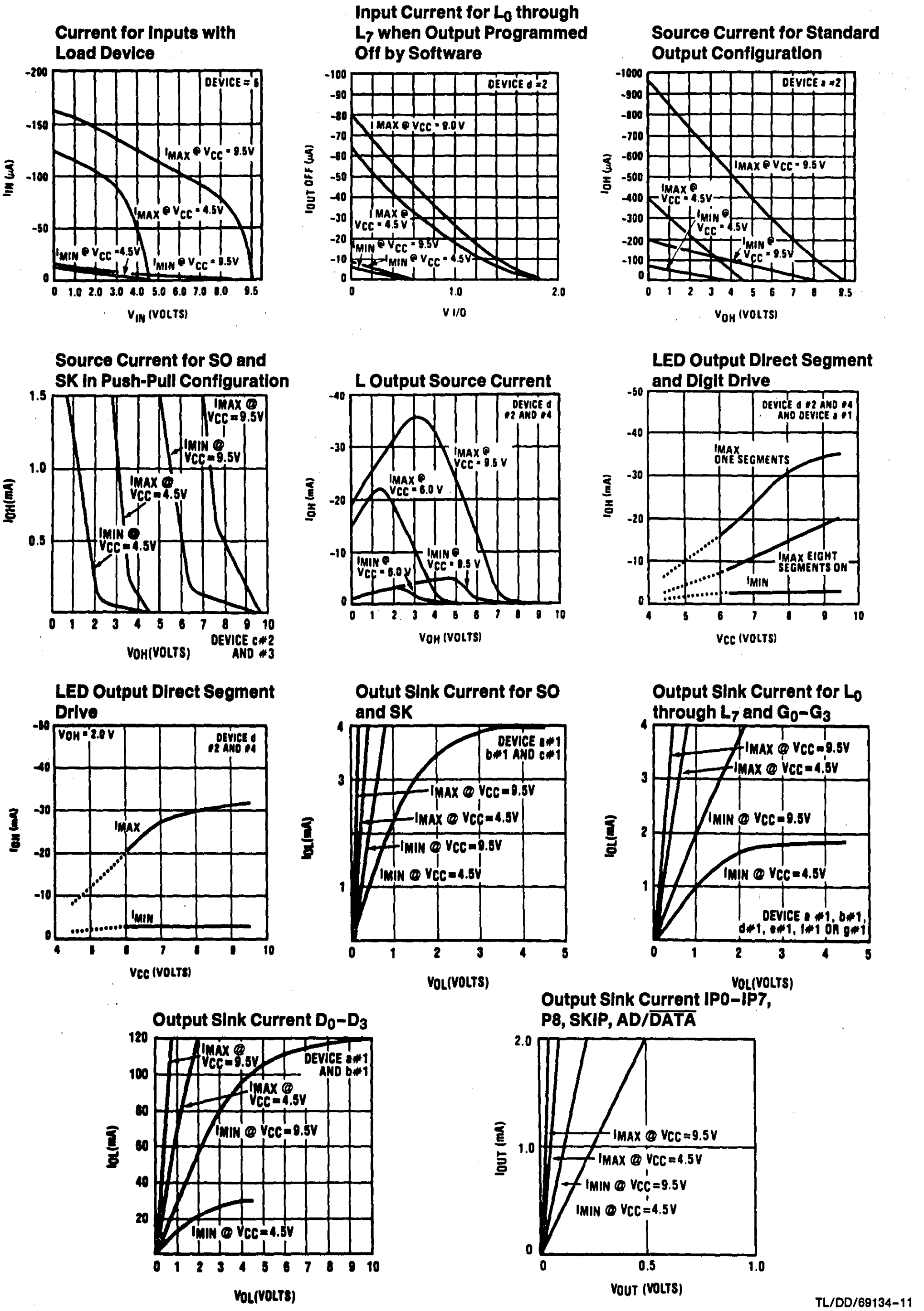
(▲ is Depletion Device)

d. L Output (LED)

e. Input with Load

FIGURE 6. Output Configurations

Typical Performance Characteristics



Current for Inputs with Load Device

Input Current for L_0 through L_7 when Output Programmed Off by Software

Source Current for Standard Output Configuration

Source Current for SO and SK in Push-Pull Configuration

L Output Source Current

LED Output Direct Segment and Digit Drive

LED Output Direct Segment Drive

Outut Sink Current for SO and SK

Output Sink Current for L_0 through L_7 and G_0–G_3

Output Sink Current D_0–D_3

Output Sink Current IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$

TL/DD/69134-11

FIGURE 7. I/O Characteristics

COP401L Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP401L instruction set.

TABLE II. COP401L Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0-15 binary (RAM Digit Select)
r	2-bit Operand Field, 0-3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0-15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP401L Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A + RAM(B) → A	None	Add RAM to A
AISC	y	5–	|0101| y |	A + y → A	Carry	Add immediate, Skip on Carry (y ≠ 0)
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

COP410L Instruction Set (Continued)

TABLE III. COP401L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM (PC_8, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– – –	0110 000 a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	– – – –	1 $a_{6:0}$ (pages 2,3 only) or 11 $a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 3)
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– – –	0110 100 a_8 $a_{7:0}$	PC + 1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100 1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
LD	r	–5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011 1111	ROM(PC_8, A, M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110	RAM(B) ⟷ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	3,15	23 BF	0010 0011 1011 1111	RAM(3,15) ⟷ A	None	Exchange A with RAM (3,15)
XDS	r	–7	00 r 0111	RAM(B) ⟷ A Bd – 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	–4	00 r 0100	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r

COP410L Instruction Set (Continued)

TABLE III. COP401L Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r, d	-	00 r (d - 1) (d = 0, 9:15)	r, d → B	Skip until not a LBI	Load B Immediate with r, d (Note 5)
LEI	y	33	0011 0011	y → EN	None	Load EN Immediate
		6–	0110 y			(Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		$G_{3:0} = 0$	Skip if G is Zero
		21	0010 0001			(all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0 = 0$	
	1	11	0001 0001		$G_1 = 0$	
	2	03	0000 0011		$G_2 = 0$	
	3	13	0001 0011		$G_3 = 0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1 = 0$	
	2	03	0000 0011		$RAM(B)_2 = 0$	
	3	13	0001 0011		$RAM(B)_3 = 0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 1110	$L_{3:0}$ → A		
OBD		33	0011 0011	Bd → D	None	Output Bd to D Outputs
		3E	0011 1110			
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ↔ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (010 is loaded into the upper 3 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus* 1, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP401L programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP401L program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or 7 will access data in the next group of 4 pages.

Typical Applications

PROM-BASED SYSTEM

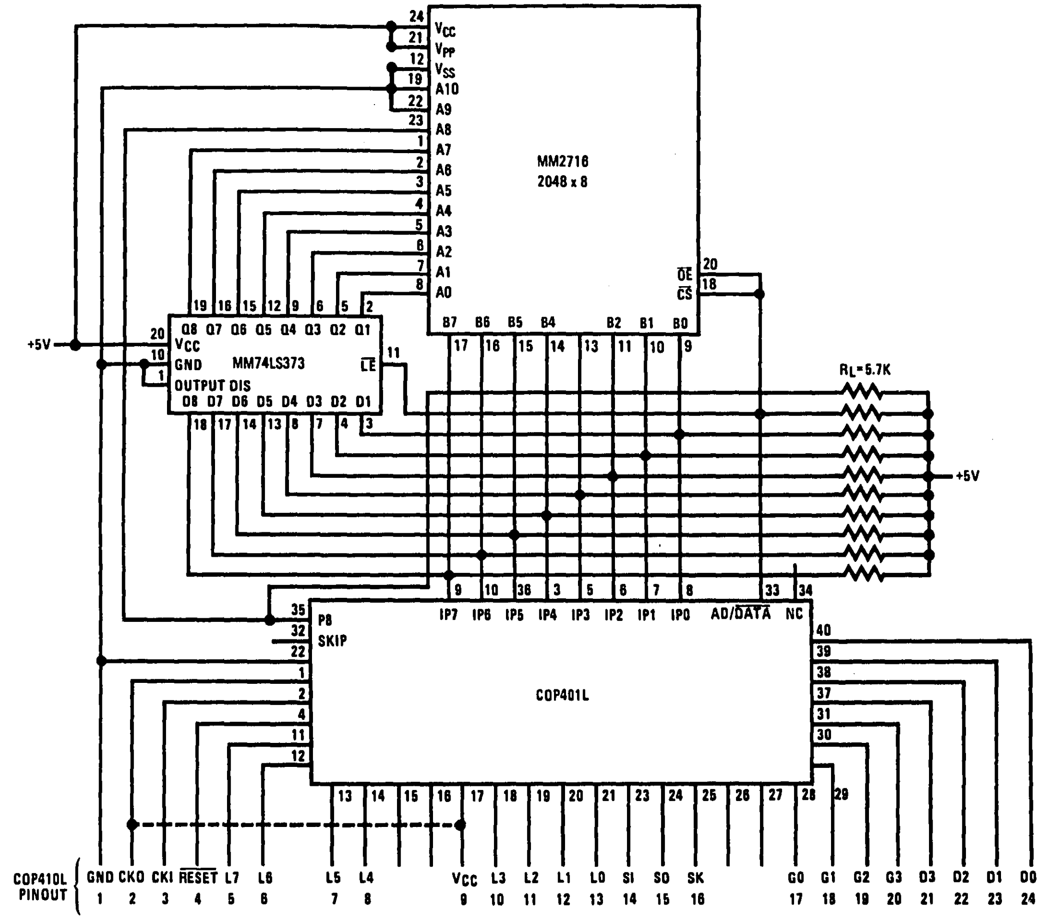
The COP401L may be used to emulate the COP410L. *Figure 8* shows the interconnect to implement a COP401L hardware emulation. This connection uses one MM5204 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP_7–IP_0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP_7–IP_0 output addresses. The 8-bit latch (MM74C373) latches the address to drive the memory.

When AD/$\overline{\text{DATA}}$ turns off, the EPROM is enabled and the IP_7–IP_0 pins will input the memory data. P8 outputs the most significant address bit to the memory. (SKIP output may be used for program debug if needed.)

24 of the COP401L pins may be configured exactly the same as a COP410L.

Typical Applications (Continued)



TL/DD/6913-12

FIGURE 8. COP401L Used to Emulate a COP410L

Option Table

COP401L MASK OPTIONS

The following COP410L options have been implemented in this basic version of the COP401L.

Option Value	Comment
Option 1 = 0	Ground—no option
Option 2 = 1	CKO is RAM power supply input
Option 3 = N/A	CKI is external clock divide-by-32 (not available on COP410L)
Option 4 = 0	Reset has load to V_{CC}
Option 5 = 2	L outputs are LED direct-drive
Option 6 = 2	
Option 7 = 2	
Option 8 = 2	
Option 9 = 1	V_{CC} pin 4.5V to 9.5V operation
Option 10 = 2	L outputs are LED direct-drive
Option 11 = 2	
Option 12 = 2	
Option 13 = 2	

Option Value	Comment
Option 14 = 0	SI has load to V_{CC}
Option 15 = 2	SO is push-pull output
Option 16 = 2	SK is push-pull output
Option 17 = 0	G outputs are standard
Option 18 = 0	
Option 19 = 0	
Option 20 = 0	
Option 21 = 0	
Option 22 = 0	D outputs are standard
Option 23 = 0	very high current
Option 24 = 0	
Option 25 = 0	L
Option 26 = 0	G Have standard TTL input levels
Option 27 = 0	SI
Option 28 = N/A	40-pin package

National Semiconductor

COP401L-X13/COP401L-R13 ROMless N-Channel Microcontroller

General Description

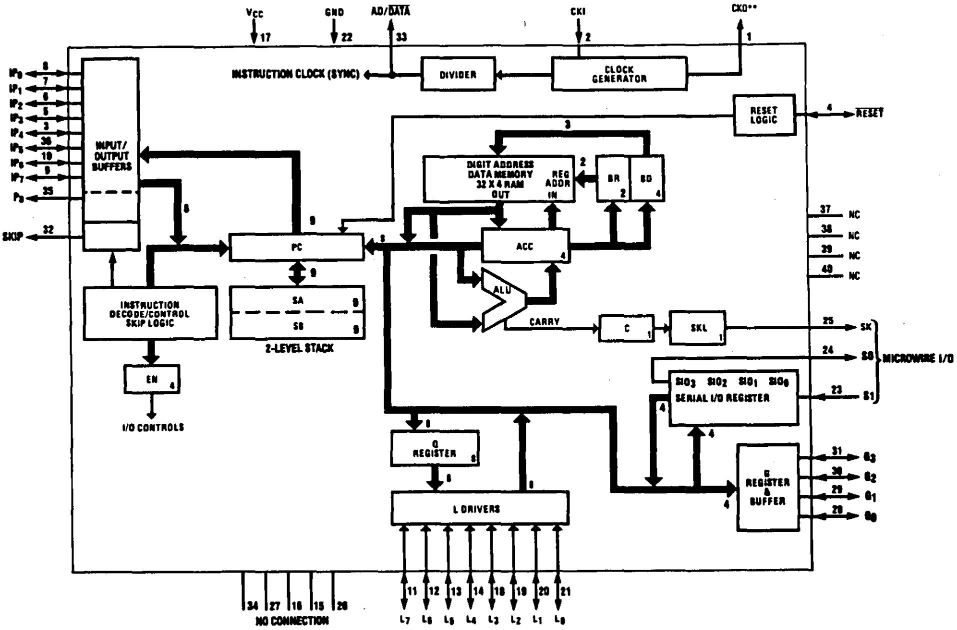
The COP401L-X13/COP401L-R13 ROMless Microcontrollers are members of the COPS™ family of microcontrollers, fabricated using N-channel, silicon gate MOS technology. The COP401L-X13/COP401L-R13 contain CPU, RAM, I/O and are identical to a COP413L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP401L-X13/COP401L-R13 will perform exactly as the COP413L. This important benefit facilitates development and debug of a COP program prior to masking the final part.

There are two clock oscillator configurations available. The crystal oscillator configuration is called COP401L-X13 and the RC oscillator configuration is called COP401L-R13.

Features

- Circuit equivalent of COP413L
- Low cost
- Powerful instruction set
- 512 × 8 ROM, 32 × 4 RAM
- Two-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5–5.5V)
- Low current drain (8 mA max)
- Internal binary counter register with serial I/O
- MICROWIRE™ compatible serial I/O
- General purpose outputs
- Software/hardware compatible with other members of COP400 family
- Pin-for-pin compatible with COP402 and COP404L
- High noise immunity inputs (V_{IL} = 1.2V, V_{IH} = 3.6V)

Block Diagram



TL/DD/8528–1

**COP401L-X13 only

FIGURE 1

COP401L-X13/COP401L-R13 Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.3 to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Power Dissipation COP413L	0.3 Watt at 70°C
Total Source Current	25 mA
Total Sink Current	40 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted.

Parameter	Conditions	Min	Max	Units
Standard Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.4	V
Operating Supply Current	All Inputs and Outputs Open		8	mA
Input Voltage Levels				
CKI Input Levels				
Ceramic Resonator Input (÷8)				
Logic High (V_{IH})		3.0		V
Logic Low (V_{IL})			0.4	V
CKI (RC), Reset Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low			0.6	V
SO Input Level (Test Mode)	(Note 2)	2.5		V
IP0–IP7, SI Input Level				
Logic High	(TTL Level)	2.0		V
Logic Low			0.8	V
L, G Inputs				
Logic High	(High Trip Levels)	3.6		V
Logic Low			1.2	V
Input Capacitance			7	pF
Reset Input Leakage		−1	+1	μA
Output Current Levels				
Output Sink Current (I_{OL})				
SO and SK Outputs	V_{OL} = 0.4V	0.9		mA
L0–L7 Outputs, G0–G3	V_{OL} = 0.4V	0.4		mA
CKO	V_{OL} = 0.4V	0.2		mA
IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$	V_{OL} = 0.4V	1.6		mA
Output Source Current (I_{OH})				
L0–L7 G0–G3, SO, SK	V_{OH} = 2.4V	−25		μA
IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$	V_{OH} = 2.4V	−25		μA
SO, SK	V_{OH} = 1.0V	−1.2		mA
IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$	V_{OH} = 1.0V	−1.2		mA
SI Input Load Source Current	V_{IL} = 0V	−10	−140	μA
Total Sink Current Allowed				
L7–L4, G Port			4	mA
L3–L0			4	mA
Any Other Pin			2.0	mA
Total Source Current Allowed				
Each Pin			1.5	mA

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

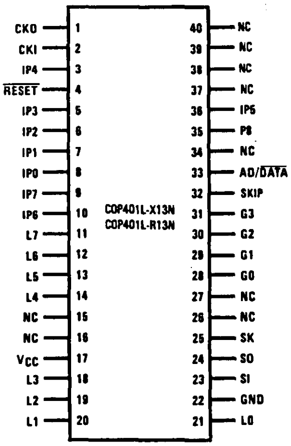
Note 2: SO output "0" level must be less than 0.8V for normal operation.

AC Electrical Characteristics $0°C \le T_A \le 70°C, 4.5V \le V_{CC} \le 5.5V$

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time - t_c		16	40	μs
CKI				
Input Frequency - fi	÷ 8 Mode	0.2	0.5	MHz
Duty Cycle		30	60	%
Rise Time	fi = 0.5 MHz		500	ns
Fall Time			200	ns
CKI Using RC (÷ 4)	R = 56 kΩ ±5%			
	C = 100 pF ±10%			
Instruction Cycle Time (Note 1)		16	28	μs
Inputs:				
G3–G0, L7–L0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
SI, IP0–IP7				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
Output Propagation Delay	Test Condition:			
	C_L = 50 pF, V_{OUT} = 1.5V			
SO, SK Outputs	R_L = 20 kΩ			
tpd1, tpd0			4.0	μs
L, G Outputs	R_L = 20 kΩ			
tpd1, tpd0			5.6	μs
IP0–IP7, P8, SKIP	R_L = 5 kΩ			
tpd1, tpd0			7.2	μs

Note 1: Variation due to the device included.

Connection Diagram



TL/DD/8528-2

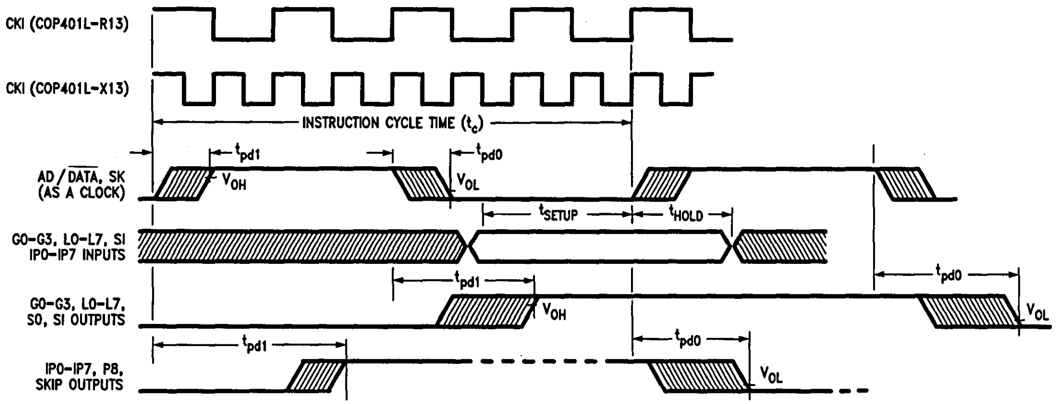
FIGURE 2

Order Number COP401L-X13N or COP401L-R13N
See NS Package Number N40A

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirectional I/O ports
G_3–G_0	4 bidirectional I/O ports
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{DATA}$	Address out/data in flag
CKI	System oscillator input
CKO	System oscillator output or NC
$\overline{RESET}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8	Most significant ROM address bit output
SKIP	Instruction skip output

Timing Waveform



TL/DD/8528-3

FIGURE 3. Input/Output Timing Diagram

Development Support

The MOLE (Microcontroller On Line Emulator) is a low cost development system and real time emulator for COP's products. They also include TMP, 8050, and the new 16-bit HPC Microcontroller Family. The MOLE provides effective support for the development of both software and hardware in the user's application.

The purpose of the MOLE is to provide a tool to write and assemble code, emulate code for the target microcontroller and assist in debugging of the system.

The MOLE can be connected to various hosts, IBM PC STARPLEX™, Kaypro, Apple, and Intel Systems, via RS-232 port. This link facilitate the up loading/down loading of code, supports host assembly and mass storage.

The MOLE consists of three parts; brain, personality and optional host software.

The brain board is the computing engine of the system. It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, from programming and diagnostic operation. It has three serial ports which can be connected to a terminal, host system, printer, modem or to other MOLE's in a multi-MOLE environment.

The personality board contains the necessary hardware and firmware needed to emulate the target microcontroller. The emulation cable which replaces the target controller attaches to this board. The software contains a cross assembler and communications program for up loading and down loading code from the MOLE.

MOLE Ordering Information

P/N	**Description**
MOLE-BRAIN	MOLE Computer Board
MOLE-COPS-PB1	COPS' Personality Board
MOLE-XXX-YYY	Optional Software

Where XXX = COPS, TMP, 8050, or HPC

YYY = Host System, IBM, APPLE, KAY (Kaypro), CP/M

Functional Description

A block diagram of the COP401L-X13/COP401L-R13 is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2 volts). When a bit is reset, it is a logic "0" (less than 0.8 volts).

PROGRAM MEMORY

Program Memory consists of a 512-byte external memory. As can be seen by an examination of the COP401L-X13/ COP401L-R13 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 8 pages of 64 words each.

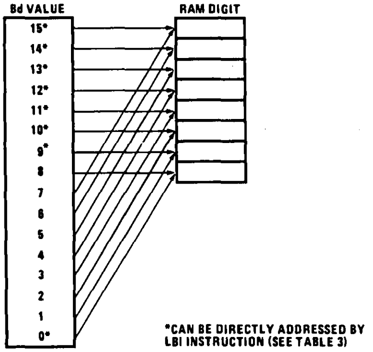
ROM addressing is accomplished by a 9-bit PC register. Its binary value selects one of the 512 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 9-bit binary count value. Two levels of subroutine nesting are implemented by the 9-bit subroutine save registers, SA and SB, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 128-bit RAM, organized as 4 data registers of 8 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 3 bits of the 4-bit Bd select 1 of 8 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the XAD 3,15 instruction.

The most significant bit of Bd is not used to select a RAM digit. Hence each physical digit of RAM may be selected by two different values of Bd as shown in *Figure 4* below. The skip condition for XIS and XDS instructions will be true if Bd changes between 0 and 15, but NOT between 7 and 8 (see Table 3).



TL/DD/8528-4

FIGURE 4. RAM Digit Address to Physical RAM Digit Mapping

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Bd portion of the B register, to load 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions of the COP401L-X13/COP401L-R13, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

The G register contents are outputs to 4 general-purpose bidirectional I/O ports.

The Q register is an internal, latched, 8-bit register, used to hold data loaded from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M.

The SIO register functions as a 4-bit serial-in-/serial-out shift register or as a binary counter depending on the contents of the EN Register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL Latch. In the counter mode, SK is the output of SKL in the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN registers (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO Register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI Input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO Output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data pesent at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. EN_1 is not used. It has no effect on COP401L-X13/ COP401L-R13 operation.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high impedance input state.

Functional Description (Continued)

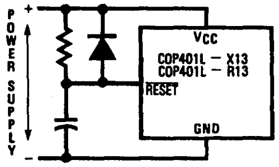
TABLE I. Enable Register Modes - Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = Clock If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = Clock If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0". Table 1 provides a summary of the modes associated with EN_3 and EN_0.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{\text{RESET}}$ pin as shown below (*Figure 5*). The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, provided it stays low for at least three instruction cycle times.



RC > 5 x POWER SUPPLY RISE TIME

TL/DD/8528-5

Figure 5. Power-Up Clear Circuit

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP401L-X13/COP401L-R13 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE® outputs
3. TTL-compatible inputs
4. access time = 5 µs max.

Typically these requirements are met using bipolar or MOS PROMs.

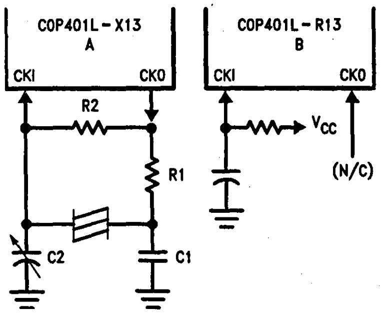
During operation, the address of the next instruction is sent out on P8 and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P8 is a dedicated address output, and does not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

There are two basic clock oscillator configurations available as shown by *Figure 6*.

a. The COP401L-X13 is a Resonator Controlled Oscillator. CKI and CKO are connected to an external ceramic resonator. The instruction cycle frequency equals the resonator frequency divided by 8.

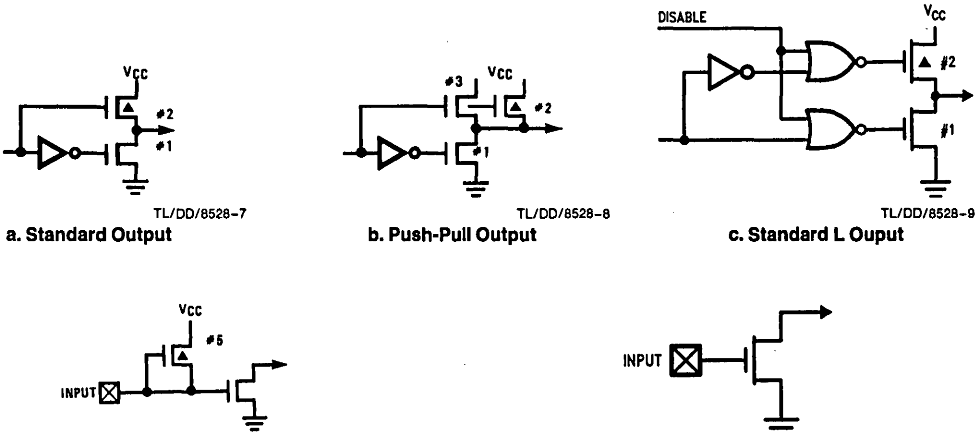
b. The COP401L-R13 is a RC Controlled Oscillator. CKI is configured as a single pin RC controlled Schmitt trigger oscillator. The instruction cycle equals the oscillation frequency divided by 4. CKO becomes no connection.



TL/DD/8528-6

FIGURE 6. COP401L-X13/COP401L-R13 Oscillator

Functional Description (Continued)



TL/DD/8528-7

a. Standard Output

TL/DD/8528-8

b. Push-Pull Output

TL/DD/8528-9

c. Standard L Ouput

TL/DD/8528-10

d. Input With Load

TL/DD/8528-11

e. Hi-Z Input

FIGURE 7. Input and Output Configurations

Ceramic Resonator Oscillator

Resonator Value	Component Values			
	R1 (Ω)	R2 (Ω)	C1 (pF)	C2 (pF)
455 kHz	4.7k	1M	220	220

RC Controlled Oscillator

R (kΩ)	C (pF)	Instruction Cycle Time (in μs)
51	100	19 ± 15%
82	56	19 ± 13%

Note: 200 kΩ≥R≥25 kΩ
220 pF≥C≥50 pF

I/O CONFIGURATIONS

COP401L-X13/COP401L-R13 inputs and outputs have the following configurations, illustrated in *Figure 7.*

a. G0–G3—an enhancement mode device to ground in conjunction with depletion-mode device to V_{CC}.

b. SO, SK, IP0–IP7, P8, SKIP, AD/$\overline{\text{DATA}}$—an enhancement mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

c. L0–L7—same as a, but may be disabled.

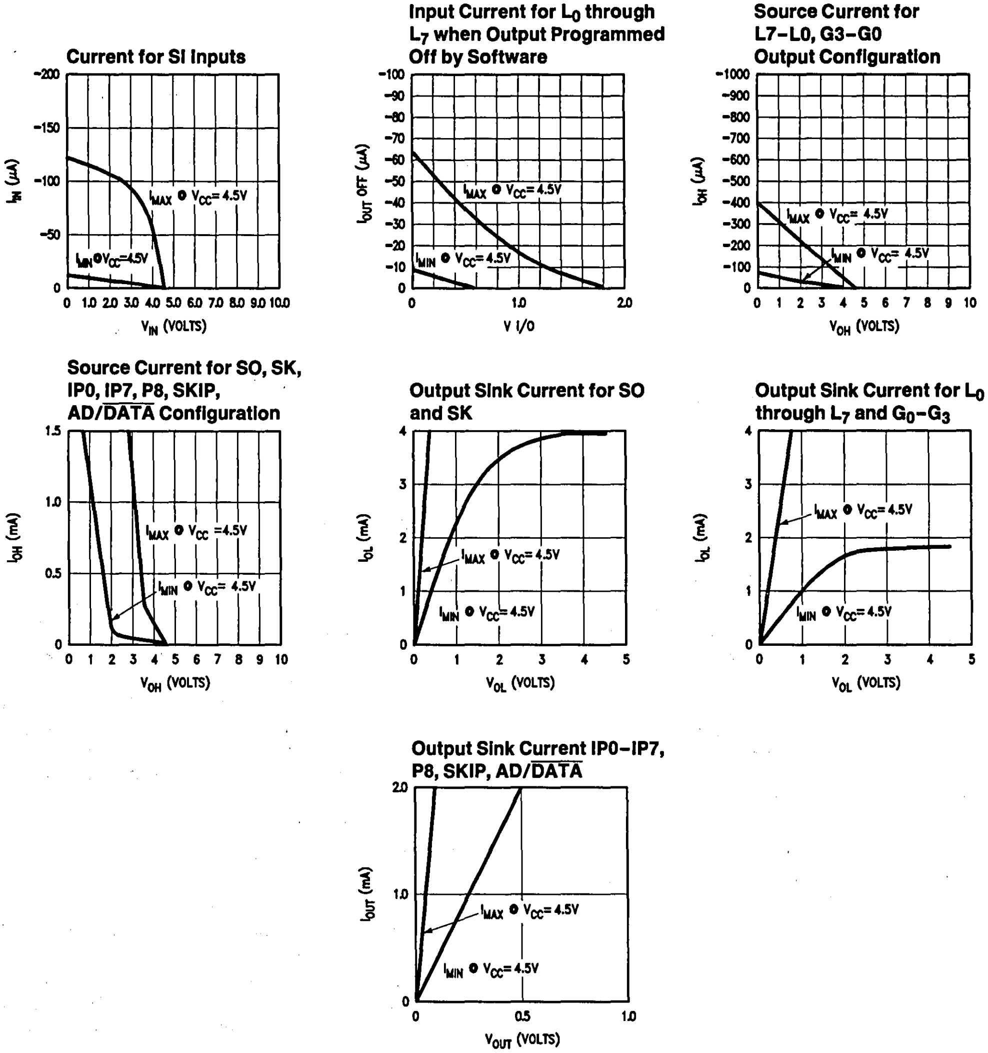
d. SI has on-chip depletion load device to V_{CC}.

e. $\overline{\text{RESET}}$ has a Hi-Z input which must be driven to a "1" or "0" by external components.

Curves are given in *Figure 8* to allow the designer to effectively use the I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current, however, when the L lines are used as inputs, the disabled depletion device can not be relied on to source sufficient current to pull an input to a logic "1".

Typical Performance Characteristics



Current for SI Inputs

Input Current for L_0 through L_7 when Output Programmed Off by Software

Source Current for L7-L0, G3-G0 Output Configuration

Source Current for SO, SK, IP0, IP7, P8, SKIP, AD/$\overline{\text{DATA}}$ Configuration

Output Sink Current for SO and SK

Output Sink Current for L_0 through L_7 and G_0-G_3

Output Sink Current IP0-IP7, P8, SKIP, AD/$\overline{\text{DATA}}$

TL/DD/8528-12

FIGURE 8. I/O Characteristics

COP401L-X13/COP401L-R13 Instruction Set

Table II is a symbol table providing internal architecture, Instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP401L-X13/COP401L-R13 instruction set.

TABLE II. COP401L-X13/COP401L-R13 Instruction Set Table Symbols

Symbol	Definition
Internal Architecture Symbols	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
PC	9-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	9-bit Subroutine Save Register A
SB	9-bit Subroutine Save Register B
SIO	4-bit Shift Register and Counter
SK	Logic Controlled Clock Output
Instruction Operand Symbols	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0-511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
Operational Symbols	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP401L-X13/COP401L-R13 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011|0000	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011|0001	A+RAM(B) → A	None	Add RAM to A
AISC	y	5–	0101| y	A+y → A	Carry	Add Immediate, Skip on Carry ($y \neq 0$)
CLRA		00	0000|0000	0 → A	None	Clear A
COMP		40	0100|0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100|0100	None	None	No Operation
RC		32	0011|0010	"0" → C	None	Reset C
SC		22	0010|0010	"1" → C	None	Set C
XOR		02	0000|0010	A⊕RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111|1111	ROM(PC_8,A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– –	0110|000|a_8 $a_{7:0}$	a → PC	None	Jump
JP	a	– –	1|$a_{6:0}$ (pages 2, 3 only) or 11|$a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within-Page (Note 3)
JSRP	a	–	10|$a_{5:0}$	PC+1 → SA → SB 010 → $PC_{8:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6– –	0110|100|a_8 $a_{7:0}$	PC+1 → SA → SB a → PC	None	Jump to Subroutine
RET		48	0100|1000	SB → SA → PC	None	Return from Subroutine
RETSK		49	0100|1001	SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011|0011 0011|1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
LD	r	–5	00|r|0101	RAM(B) → A Br⊕r → Br	None	Load RAM into A, Exclusive-OR Br with r
LQID		BF	1011|1111	ROM(PC_8, A,M) → Q SA → SB	None	Load Q Indirect (Note 2)
RMB	0	4C	0100|1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100|0101	0 → $RAM(B)_1$		
	2	42	0100|0010	0 → $RAM(B)_2$		
	3	43	0100|0011	0 → $RAM(B)_3$		
SMB	0	4D	0100|1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100|0111	1 → $RAM(B)_1$		
	2	46	0100|0110	1 → $RAM(B)_2$		
	3	4B	0100|1011	1 → $RAM(B)_3$		
STII	y	7–	0111| y	y → RAM(B) Bd+1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00|r|0110	RAM(B) ←→ A Br⊕r → Br	None	Exchange RAM with A, Exclusive-OR Br with r

TABLE III. COP401L-X13/COP401L-R13 Instruction Set (Continued)

			Machine			
		Hex	Language Code			
Mnemonic	**Operand**	**Code**	**(Binary)**	**Data Flow**	**Skip Conditions**	**Description**
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XAD	3,15	23	0010 0011	RAM(3,15) ←→ A	None	Exchange A with RAM
		BF	1011 1111			(3,15)
XDS	r	−7	00 r 0111	RAM(B) ←→ A	Bd decrements past 0	Exchange RAM with A
				Bd−1 → Bd		and Decrement Bd,
				Br⊕r → Br		Exclusive-OR Br with r
XIS	r	−4	00 r 0100	RAM(B) ←→ A	Bd increments past 15	Exchange RAM with A
				Bd+1 → Bd		and Increment Bd,
				Br⊕r → Br		Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	–	00 r (d−1)	r,d → B	Skip until not a LBI	Load B immediate with
			(d=0,9:15)			r,d (Note 5)
LEI	y	33	0011 0011	y → EN	None	Load EN Immediate
		6–	0110 y			(Note 6)
TEST INSTRUCTIONS						
SKC		20	0010 0000		C="1"	Skip if C is True
SKE		21	0010 0001		A=RAM(B)	Skip if A Equals RAM
SKGZ		33	0011 0011		$G_{3:0}=0$	Skip if G is Zero
		21	0010 0001			(all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001		$G_0=0$	
	1	11	0001 0001	2nd byte	$G_1=0$	
	2	03	0000 0011		$G_2=0$	
	3	13	0001 0011		$G_3=0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1=0$	
	2	03	0000 0011		$RAM(B)_2=0$	
	3	13	0001 0011		$RAM(B)_3=0$	
INPUT/OUTPUT INSTRUCTIONS						
ING		33	0011 0011	G → A	None	Input G Ports to A
		2A	0010 1010			
INL		33	0011 0011	$L_{7:4}$ → RAM(B)	None	Input L Ports to RAM, A
		2E	0010 1110	$L_{3:0}$ → A		
OMG		33	0011 0011	RAM(B) → G	None	Output RAM to G Ports
		3A	0011 1010			
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO
						(Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined) Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, and LQID instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: The machine code for the lower 4 bits of the LBI instruction equals the binary value of the "d" data *minus 1* e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP401L-X13/COP401L-R13 programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 9-bit word, PC_8, A, M. PC_8 is not affected by this instruction.

Note that JID requires 2 instruction cycles to execute.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 9-bit word PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM (B) → $PC_{3:0}$, leaving PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SA → SB, the previous contents of SB are lost. Also, when LQID pops the stack, the previously pushed contents of SA are left in SB. The net result is that the contents of SA are placed in SB (SA → SB). Note that LQID takes two instruction cycle times to execute.

INSTRUCTION SET NOTES

a. The first word of a COP401L-X13/COP401L-R13 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 8 pages of 64 words each. The Program Counter is a 9-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3 or will access data in the next group of 4 pages.

COPS Programming Manual

For detailed information on writing COPS programs, the COPS Programming Manual 424410284-001 provides an in-depth discussion of the COPS architecture, instruction set and general techniques of COPS programming. This manual is written with the programmer in mind.

Typical Applications

PROM-Based System

The COP401L-X13/COP401L-R13 may be used to emulate the COP413L. *Figure 9* shows the interconnect to implement a COP401L-X13/COP401L-R13 hardware emulation. This connection uses one MM2716 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP_7–IP_0 are bidirectional inputs and outputs. When the AD/$\overline{DATA}$ clocking output turns on, the EPROM drivers are disabled and IP_7–IP_0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

When AD/$\overline{DATA}$ turns off, the EPROM is enabled and the IP_7–IP_0 pins will input the memory data. P8 outputs the most significant address bit to the memory. (SKIP output may be used for program debug if needed.)

Twenty of the COP401L-X13/COP401L-R13 pins may be configured exactly the same as the COP413L. Selection of the COP401L-X13 or COP401L-R13 depends upon which oscillator is selected for the COP413L.

			Order
Oscillator Requirement			**ROMless**
COP413L	Option 1 = 0	Ceramic Resonator or external input frequency divided by 8. CKO is oscillator out.	COP401L-X13
	Option 1 = 1	Single Pin RC controlled oscillator divided by 4. CKO is no connection.	COP401L-R13

Typical Applications (Continued)

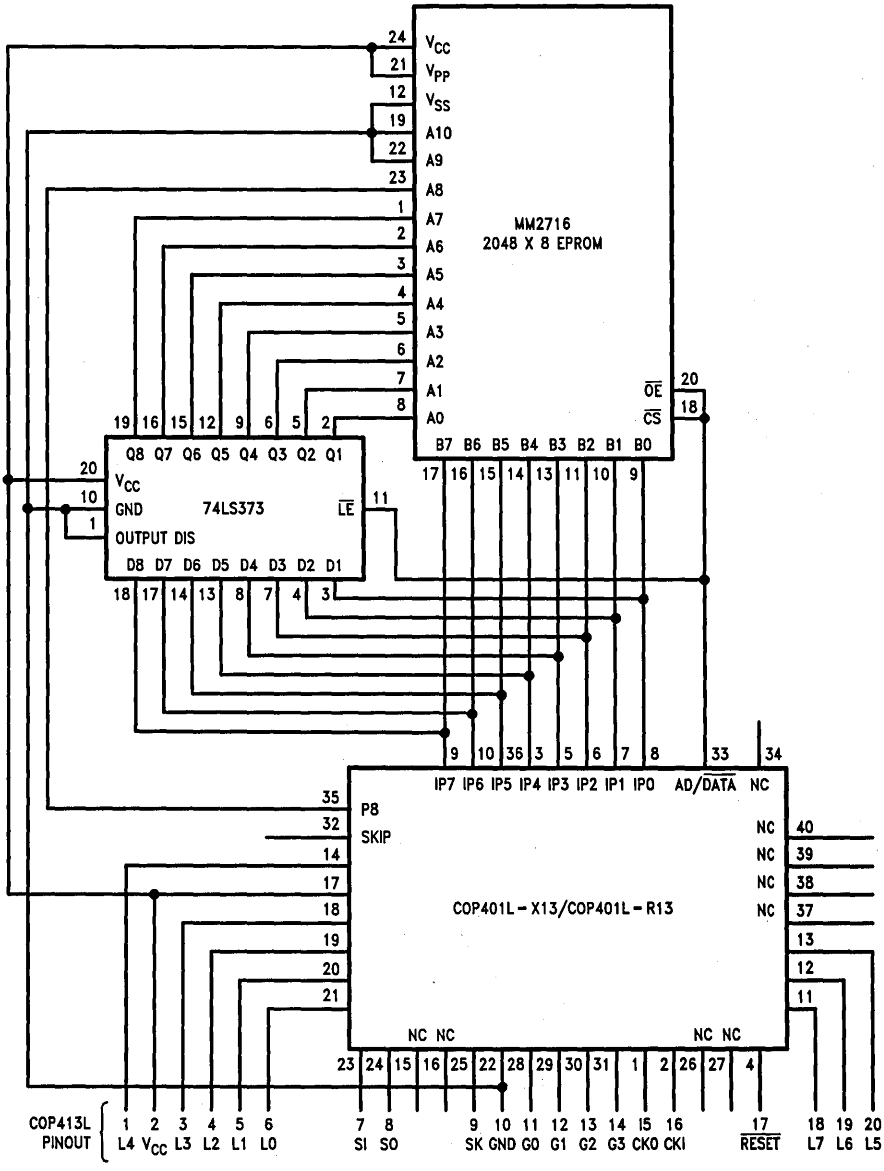


FIGURE 9. COP401L-X13/COP401L-R13 Used to Emulate a COP413L

COP402/COP402M

National Semiconductor

COP402/COP402M ROMless N-Channel Microcontrollers

General Description

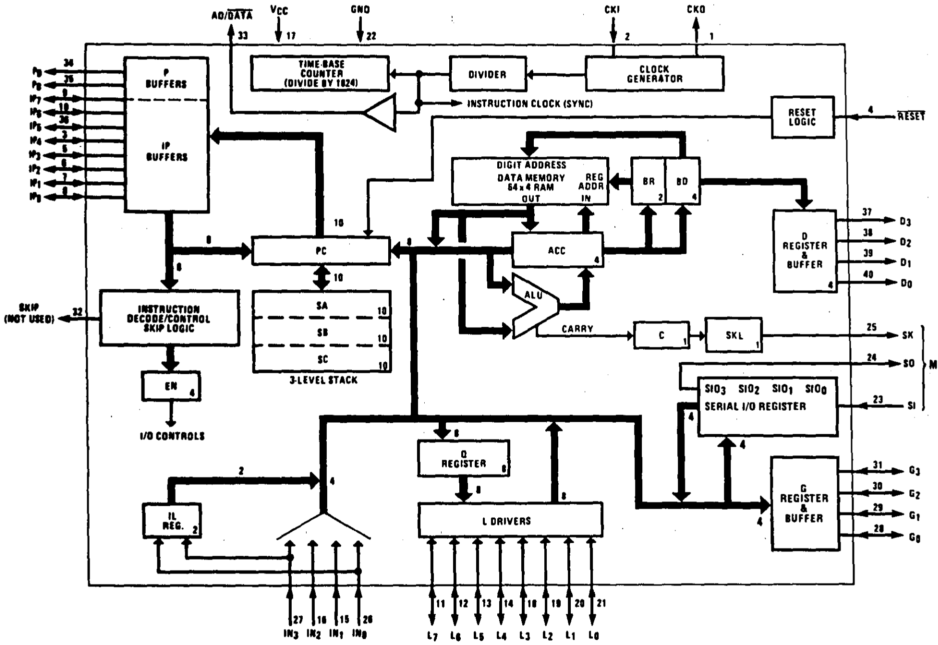
The COP402/COP402M ROMless Microcontrollers are members of the COPS™ family, fabricated using N-channel silicon gate MOS technology. Each part contains CPU, RAM, and I/O, and is identical to a COP420 device, except the ROM has been removed; pins have been added to output the ROM address and to input ROM data. In a system, the COP402 or 402M will perform exactly as the COP420; this important benefit facilitates development and debug of a COP420; this important benefit facilitates development and debug of a COP420 program prior to masking the final part. These devices are also appropriate in low volume applications, or when the program may require changing. The COP402M is identical to the COP402, except the MICROBUS™ interface option has been implemented.

The COP402 may also be used to emulate the COP410L, 411L, or 420L by appropriately reducing the clock frequency.

Features

- Extended temperature (−40°C to +85°C) COP302/COP302M, available as special order
- Low cost
- Exact circuit equivalent of COP420
- Standard 40-pin dual-in-line package
- Interfaces with standard PROM or ROM
- 64 x 4 RAM, addresses up to 1k x 8 ROM
- MICROBUS compatible (COP402M)
- Powerful instruction set
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 4.0 μs instruction time
- Single supply operation (4.5V to 6.3V)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ serial I/O capability
- Software/hardware compatible with other members of COP400 family

Block Diagram



TL/DD/6915-1

FIGURE 1

COP402/COP402M and COP302/COP302M

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin	−0.3V to +7V
Operating Temperature Range	
COP402/COP402M	0°C to 70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 sec.)	300°C
Package Power Dissipation	750 mW at 25°C 400 mW at 70°C 250 mW at 85°C
Total Sink Current	50 mA
Total Source Current	70 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

COP402/COP402M DC Electrical Characteristics

$0°C \le T_A \le 70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	6.3	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	All Outputs Open V_{CC} = 5V		40	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High		2.4		V
Logic Low		−0.3	0.4	V
Schmitt Trigger Input				
RESET				
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
Input Load Source Current	V_{CC} = 5V, V_{IN} = 0V	−100	−800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage	V_{CC} = 5V	−1	+1	μA
Output Voltage Levels				
D, G, L, SK, SO Outputs				
TTL Operation	V_{CC} = 5V ±10%			
Logic High	I_{OH} = −100 μA	2.4		V
Logic Low	I_{OL} = 1.6 mA	−0.3	0.4	V
IP0–IP7, P8, P9, SKIP, CKO, AD/$\overline{\text{DATA}}$				
Logic High	I_{OH} = −75 μA	2.4		V
Logic Low	I_{OL} = 400 μA	−0.3	0.4	V
CMOS Operation (Note 1)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = 10 μA	−0.3	0.2	V
Output Current Levels				
LED Direct Drive (COP402)	V_{CC} = 6V			
Logic High	V_{OH} = 2.0V	2.5	14	mA
TRI-STATE® (COP402M) Leakage Current	V_{CC} = 5V	−50	+50	μA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

Note 1: TRI-STATE and LED configurations are excluded.

COP402/COP402M AC Electrical Characteristics $0°C \le T_A \le 70°C, 4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	μs
Operating CKI Frequency	÷16 Mode	1.6	4.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time	Frequency = 4 MHz		60	ns
Fall Time	Frequency = 4 MHz		40	ns
Inputs:				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
Output Propagation Delay	Test Conditions: $R_L = 5k, C_L = 50\ pF, V_{OUT} = 1.5V$			
SO and SK				
t_{pd1}			1.0	μs
t_{pd0}			1.0	μs
CKO				
t_{pd1}			0.25	μs
t_{pd0}			0.25	μs
AD/$\overline{DATA}$, SKIP				
t_{pd1}			0.6	μs
t_{pd0}			0.6	μs
All Other Outputs				
t_{pd1}			1.4	μs
t_{pd0}			1.4	μs
MICROBUS Timing	$C_L = 100\ pF, V_{CC} = 5V \pm 5\%$			
Read Operation *(Figure 4)*				
Chip Select Stable before $\overline{RD}$—t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$—t_{RCS}		20		ns
$\overline{RD}$ Pulse Width—t_{RR}		400		ns
Data Delay from $\overline{RD}$—t_{RD}			375	ns
$\overline{RD}$ to Data Floating—t_{DF}			250	ns
Write Operation *(Figure 5)*				
Chip Select Stable before $\overline{WR}$—t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$—t_{WCS}		20		ns
$\overline{WR}$ Pulse Width—t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$—t_{DW}		320		ns
Data Hold Time for $\overline{WR}$—t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$—t_{WI}			700	ns

Note 1: Duty Cycle = $t_{WI}/(t_{W1} + t_{W0})$.

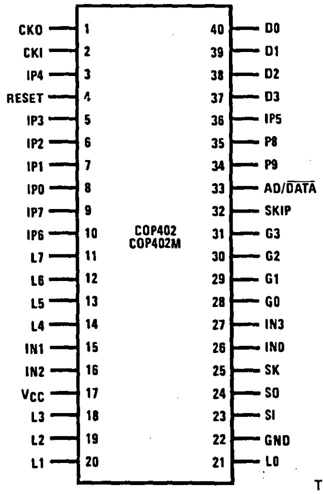
Note 2: See *Figure 9* for additional I/O characteristics.

Note 3: Voltage change must be less than 0.5V in a 1 ms period.

Note 4: Exercise great care not to exceed maximum device power dissipation limits when direct driving LEDs (or sourcing similar loads) at high temperature.

Connection Diagram

Dual-In-Line Package



TL/DD/6915-2

Top View

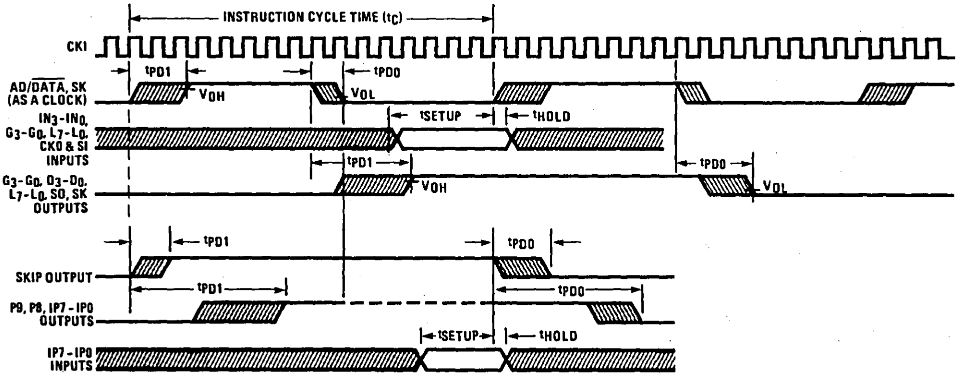
Order Number COP402N or COP402MN
See NS Package Number N40A

FIGURE 2.

Pin Descriptions

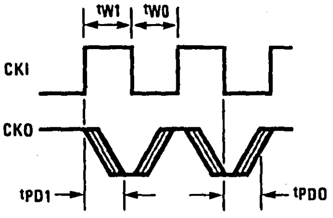
Pin	Description
L_7–L_0	8 bidirectional I/O ports with TRI-STATE
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose inputs
SI	Serial input (or counter input)
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{\text{DATA}}$	Address out/data in flag
SKIP	Instruction skip output
CKI	System oscillator input
CKO	System oscillator output
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8, P9	2 most significant ROM address outputs

Timing Diagrams



TL/DD/6915-3

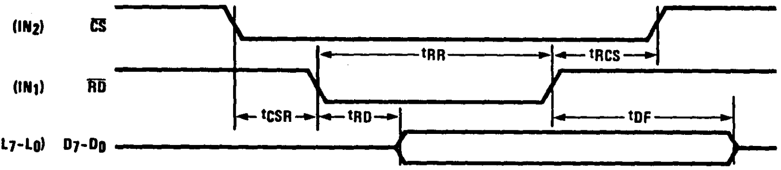
FIGURE 3a. Input/Output Timing Diagrams (Crystal ÷ 16 Mode)



TL/DD/6915-4

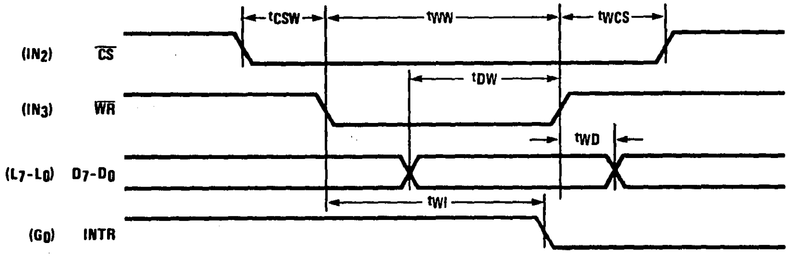
FIGURE 3b. CKO Output Timing

Timing Diagrams (Continued)



TL/DD/6915-5

FIGURE 4. MICROBUS Read Operation Timing



TL/DD/6915-6

FIGURE 5. MICROBUS Write Operation Timing

Functional Description

A block diagram of the COP402 is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 1,024-byte external memory (typically PROM). Words of this memory may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 16 pages of 64 words each.

ROM addressing is accomplished by a 10-bit PC register. Its binay value selects one of the 1,024 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential **10-bit binary count** value. Three levels of subroutine nesting are implemented by the 10-bit subroutine save registers, SA, SB and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 256-bit RAM, organized as 4 data registers of 16 4-bit digits. RAM addressing is implemented by a 6-bit B register whose upper 2 bits (Br) select 1 of 4 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instruction based upon the 6-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit **A register** (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A **4-bit adder** performs the arithmetic and logic functions of the COP402/402M, storing its results in A. It also outputs a carry bit to the 1-bit **C register**, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below.)

Four **general-purpose inputs, IN_3-IN_0**, are provided; IN_1, IN_2, and IN_3 may be selected, by a mask-programmable option, as Read Strobe, Chip Select and Write Strobe inputs, respectively, for use in MICROBUS applications.

The **D register** provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd.

The **G register** contents are outputs to 4 general-purpose bidirectional I/O ports. G_0 may be mask-programmed as a "ready" output for MICROBUS applications.

The **Q register** is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.) With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

Functional Description (Continued)

The **8 L drivers**, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa–Sg and decimal point segments of the display.

The **SIO register** functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The **XAS instruction** copies C into the SKL latch. In the counter mode, SK is the output of SKL. In the shift register mode, SK outputs SKL ANDed with internal instruction cycle clock.

The **EN register** is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3–EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state. If the MICROBUS option is being used, EN_2 does not affect the L drivers.
4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0.

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC + 1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC + 1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:

1. EN_1 has been set.
2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
3. A currently executing instruction has been completed.
4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon the popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. At *this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and the LQID instruction should not be nested within the interrupt servicing routine since their popping of the stack enables any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. An LEI instruction can be put immediately before the RET to re-enable interrupts.

TABLE I. Enable Register Modes—Bits EN_3 and EN_0

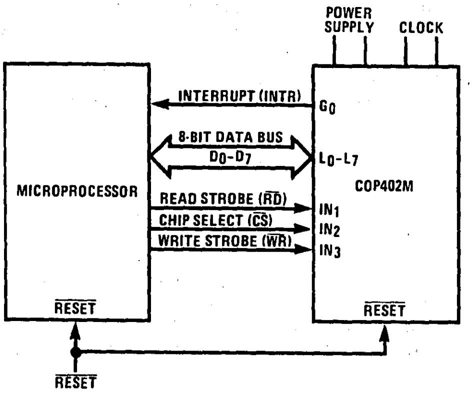
EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = SYNC If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = SYNC If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

MICROBUS INTERFACE

The COP402M can be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN_1, IN_2, and IN_3 general purpose inputs become MICROBUS compatible read-strobe, chip-select, and write-strobe lines, respectively. IN_1 becomes $\overline{RD}$—a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN_2 becomes $\overline{CS}$—a logic "0" on this line selects the COP402M as the μP peripheral device by enabling the operation of the $\overline{RD}$ and $\overline{WR}$ lines and allows for the selection of one of several peripheral components. IN_3 becomes $\overline{WR}$—a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP402M. G_0 becomes INTR, a "ready" output reset by a write pulse from the μP on the $\overline{WR}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP402M.

This option has been designed for compatibility with National's MICROBUS—a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS, National Publication.) The functioning and timing relationships between the COP402M signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection to the MICROBUS is shown in *Figure 6*.



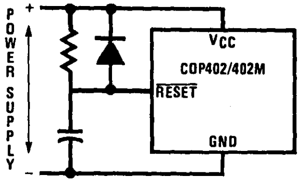
TL/DD/6915-7

FIGURE 6. MICROBUS Option Interconnect

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 μs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{RESET}$ pin as shown below. The $\overline{RESET}$ pin is configured as a Schmitt trigger input. If not used it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{RESET}$ input, provided it stays low for at least two instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, G, and SO are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.



RC ≥ 5 × Power Supply Rise Time TL/DD/6915-8

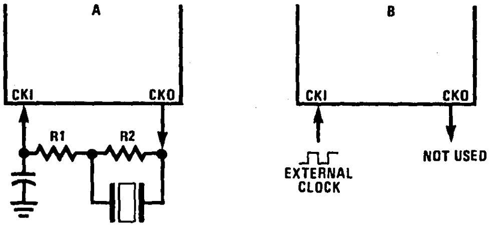
FIGURE 7. Power-Up Clear Circuit

OSCILLATOR

There are two basic clock oscillator configurations available as shown by *Figure 8*.

a. Crystal Controlled Oscillator. CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 16.

b. External Oscillator. CKI is driven by an external clock signal. The instruction cycle time is the clock frequency divided by 16.



TL/DD/6915-9

Crystal Value	Component Values		
	R1	R2	C
4 MHz	1k	1M	27 pF
3.58 MHz	1k	1M	27 pF
2.09 MHz	1k	1M	56 pF

FIGURE 8. COP402/402M Oscillator

EXTERNAL MEMORY INTERFACE

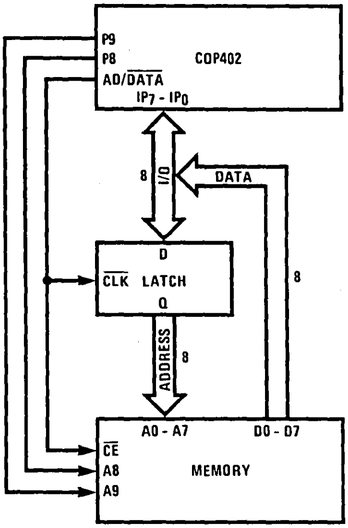
The COP402 and COP402M are designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL = compatible inputs
4. access time = 1.0 μs, max.

Typically these requirements are met using bipolar or MOS PROMs.

Functional Description (Continued)

During operation, the address of the next instruction is sent out on P9, P8, and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{\text{DATA}}$ line; P9 and P8 are dedicated address outputs, and do not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input. A simplified block diagram of the external memory interface is shown in *Figure 9.*



TL/DD/6915-10

FIGURE 9. External Memory Interface to COP402

INPUT/OUTPUT

COP402 outputs have the following configurations, illustrated in *Figure 10.*

a. **Standard**—an enhancement-mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with TTL and CMOS input requirements.

b. **High Drive**—same as **a.** except greater current sourcing capability.

c. **Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

d. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display.

e. **TRI-STATE Push-Pull**—an enhancement-mode device to ground and V_{CC} intended to meet the requirements associated with the MICROBUS option. These outputs are TRI-STATE outputs, allowing for connection of these outputs to a data bus shared by other bus drivers.

f. Inputs have an on-chip depletion load device to V_{CC}, as shown in *Figure 10f.*

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 10* for each of these devices.

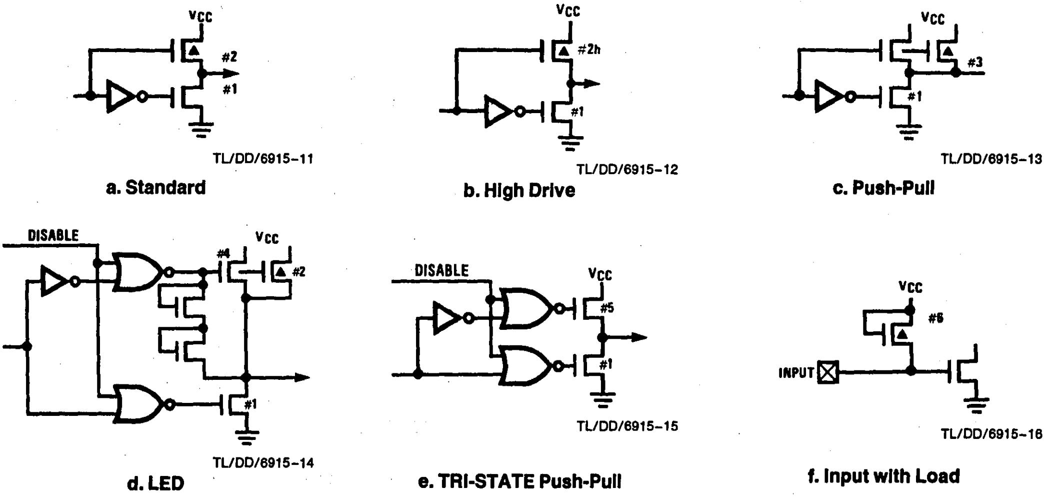
The SO, SK outputs are configured as shown in *Figure 10c.* The D and G outputs are configured as shown in *Figure 10a.*

Functional Description (Continued)

Note that when inputting data to the G ports, the G outputs should be set to "1". The L outputs are configured as in *Figure 10d* on the COP402. On the COP402M the L outputs are as in *Figure 10e*.

An important point to remember if using configuration d with the L drivers is that even when the L drivers are disabled, the depletion load device will source a small amount of current. (See *Figure 11*.)

IP7 through IP0 outputs are configured as shown in *Figure 10c*; P9, P8, SKIP, and AD/$\overline{\text{DATA}}$ are configured as shown in *Figure 10b*.



TL/DD/6915-11

a. Standard

TL/DD/6915-12

b. High Drive

TL/DD/6915-13

c. Push-Pull

TL/DD/6915-14

d. LED

TL/DD/6915-15

e. TRI-STATE Push-Pull

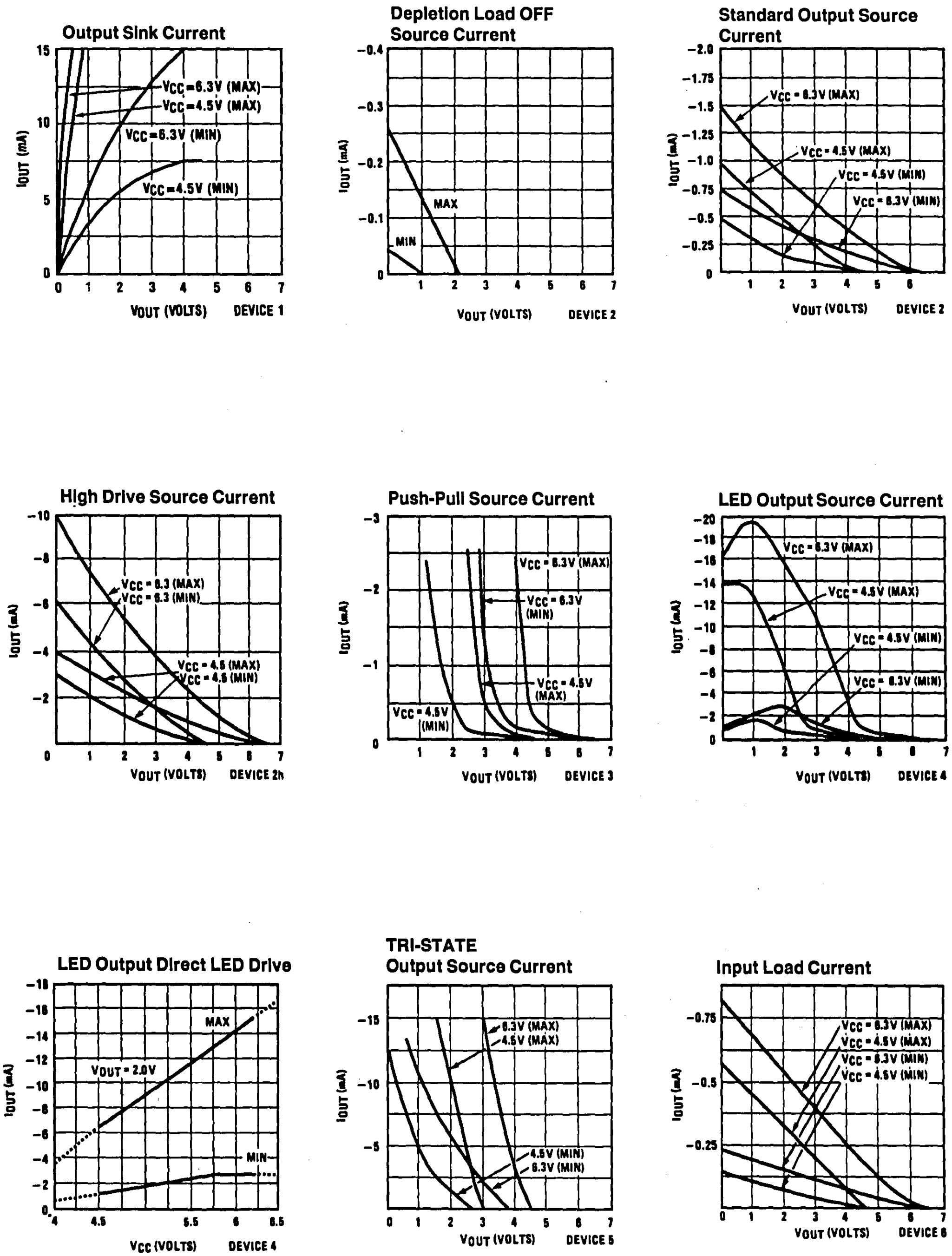
TL/DD/6915-16

f. Input with Load

(▲ is Depletion Device)

FIGURE 10. Input/Output Configurations

Typical Performance Characteristics


Output Sink Current

Depletion Load OFF Source Current

Standard Output Source Current

High Drive Source Current

Push-Pull Source Current

LED Output Source Current

LED Output Direct LED Drive

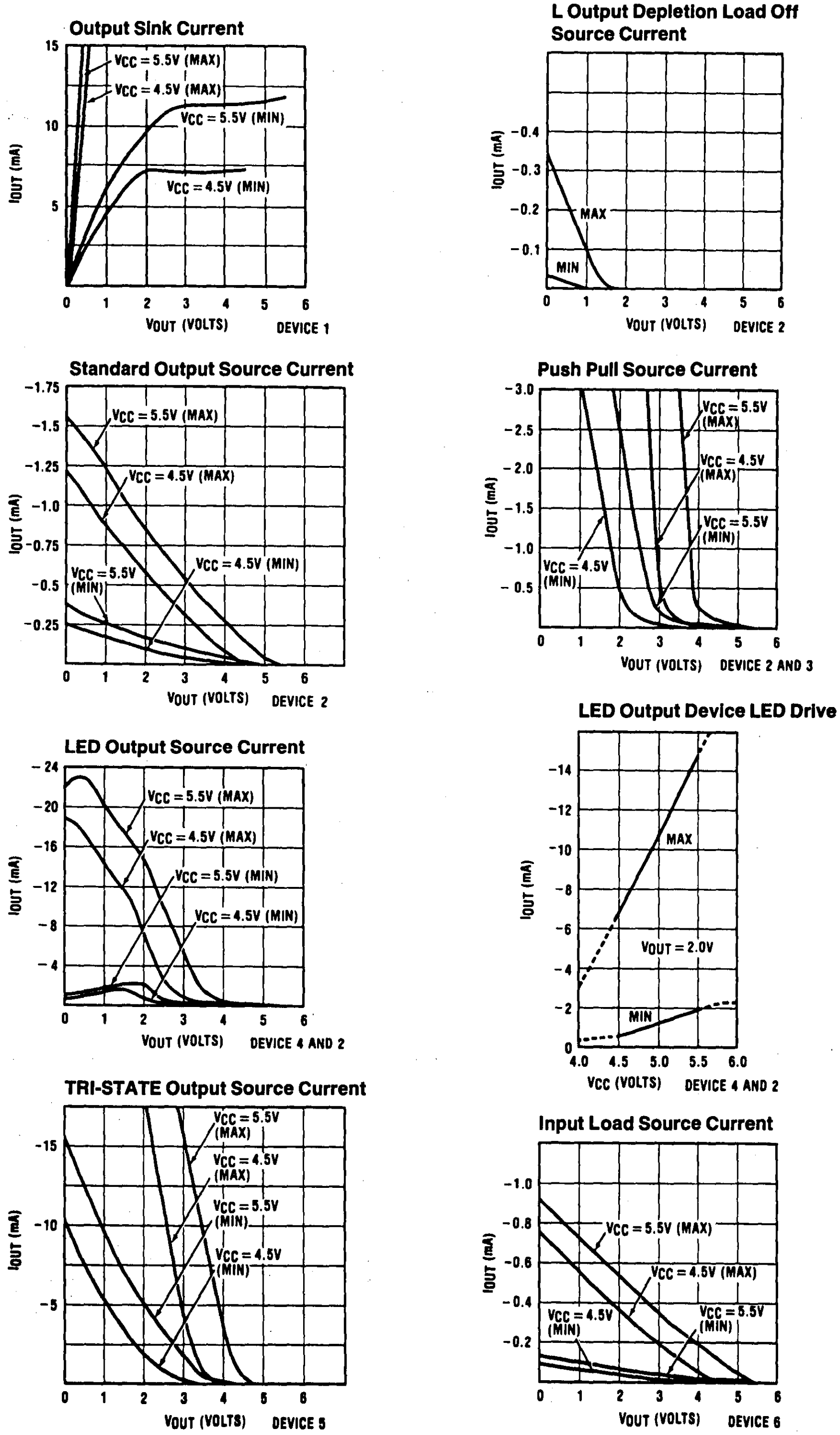
TRI-STATE Output Source Current

Input Load Current

FIGURE 11. COP402/COP402M Input/Output Characteristics

TL/DD/6915-17

Typical Performance Characteristics (Continued)



Output Sink Current

L Output Depletion Load Off Source Current

Standard Output Source Current

Push Pull Source Current

LED Output Source Current

LED Output Device LED Drive

TRI-STATE Output Source Current

Input Load Source Current

TL/DD/6915–18

FIGURE 11a. COP302/COP302M Input/Output Characteristics

Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code, data flow, skip conditions and description associated with each instruction in the COP402/402M instruction set.

TABLE II. COP402/COP402M Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	6-bit RAM Address Register
Br	Upper 2 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit Latches Associated with the IN_3 or IN_0 inputs
IN	4-bit Input port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
P	2-bit ROM Address Port
PC	10-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	10-bit Subroutine Save Register A
SB	10-bit Subroutine Save Register B
SC	10-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	2-bit Operand Field, 0–3 binary (RAM Register Select)
a	9-bit Operand Field, 0–511 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
←→	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE III. COP402/COP402M Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
ADT		4A	0100 1010	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	0101 y	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	0001 0000	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

Instruction Set (Continued)

TABLE III. COP402/COP402M Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{9:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 3)
JMP	a	6– --	0110 00 $a_{9:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	--	1 $a_{6:0}$ (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 4)
		--	11 $a_{5:0}$ (all other pages)	a → $PC_{5:0}$		
JSRP	a	--	10 $a_{5:0}$	PC + 1 → SA → SB → SC 0010 → $PC_{9:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– --	0110 10 $a_{9:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 --	0010 0011 00 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{9:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0	4C	0100 1100	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	0100 0101	0 → $RAM(B)_1$		
	2	42	0100 0010	0 → $RAM(B)_2$		
	3	43	0100 0011	0 → $RAM(B)_3$		
SMB	0	4D	0100 1101	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	0100 0111	1 → $RAM(B)_1$		
	2	46	0100 0110	1 → $RAM(B)_2$		
	3	4B	0100 1011	1 → $RAM(B)_3$		
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110	RAM(B) ↔ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 --	0010 0011 10 r d	RAM(r,d) ↔ A	None	Exchange A with RAM pointed to directly by r,d

Instruction Set (Continued)

TABLE III. COP402/COP402M Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XDS	r	-7	00|r|0111	RAM(B) ←→ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r
XIS	r	-4	00|r|0100	RAM(B) ←→ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101|0000	A → Bd	None	Copy A to Bd
CBA		4E	0100|1110	Bd → A	None	Copy Bd to A
LBI	r,d	--	00|r|(d − 1) (d=0, 9:15) or	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
		33	0011|0011			
		--	10|r|d (any d)			
LEI	y	33	0011|0011	y → EN	None	Load EN Immediate (Note 7)
		6-	0110|y			
XABR		12	0001|0010	A ←→ Br (0,0 → A_3,A_2)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010|0000		C = "1"	Skip if C is True
SKE		21	0010|0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33	0011|0011		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
		21	0010|0001			
SKGBZ		33	0011|0011	1st byte		Skip if G Bit is Zero
	0	01	0000|0001	2nd byte	$G_0 = 0$	
	1	11	0001|0001		$G_1 = 0$	
	2	03	0000|0011		$G_2 = 0$	
	3	13	0001|0011		$G_3 = 0$	
SKMBZ	0	01	0000|0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001|0001		$RAM(B)_1 = 0$	
	2	03	0000|0011		$RAM(B)_2 = 0$	
	3	13	0001|0011		$RAM(B)_3 = 0$	
SKT		41	0100|0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 3)

Instruction Set (Continued)

TABLE III. COP402/COP402M Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN → A	None	Input IN Inputs to A (Notes 2 and 8)
INIL		33 29	0011 0011 0010 1001	IL_3, "0", IL_0 → A	None	Input IL Latches to A (Note 3)
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM,A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5-	0011 0011 0101 y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 0011 0011 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 3)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit register.

Note 2: The ININ instruction is not available on the 24-pin COP421 since this device does not contain the IN inputs.

Note 3: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 4: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 5: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 6: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 7: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 8: The COP402M will always read a "1" into A1 with the ININ instruction.

Description of Selected Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing programs.

XAS INSTRUCTION

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register.

The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register, above.) If SIO is selected as a shift register, an XAS instruction must be performed once evey 4 instruction cycles to effect a continuous data stream.

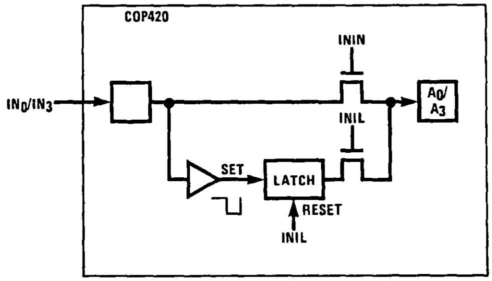
JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 10-bit word, $PC_{9:8}$, A, M. PC_9 and PC_8 are not affected by this instruction.

Note that JID requires 2 instruction cycles.

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 12*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IN_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. If CKO is mask programmed as a general purpose input, an INIL will input the state of CKO into A2. If CKO has not been so programmed, a "1" will be placed in A2. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon the execution of an ININ instruction. (See Table III, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.



TL/DD/6915-19

FIGURE 12. IN_0/IN_3 Latches

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 10-bit word PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC). Note that LQID takes two instruction cycle times to execute.

SKT INSTRUCTION

The SKT (Skip on Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the controller to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz crystal as the time-base to the clock generator, the instruction cycle clock frequency will be 131 kHz (crystal frequency ÷ 16) and the binary counter output pulse frequency will be 128 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 128 ticks.

INSTRUCTION SET NOTES

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths take the same number of cycle times whether instructions are skipped or executed, except JID and LQID. LQID and JID take two cycle times if executed and one if skipped.

c. The ROM is organized into 16 pages of 64 words each. The Program Counter is a 10-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, or 15 will access data in the next group of 4 pages.

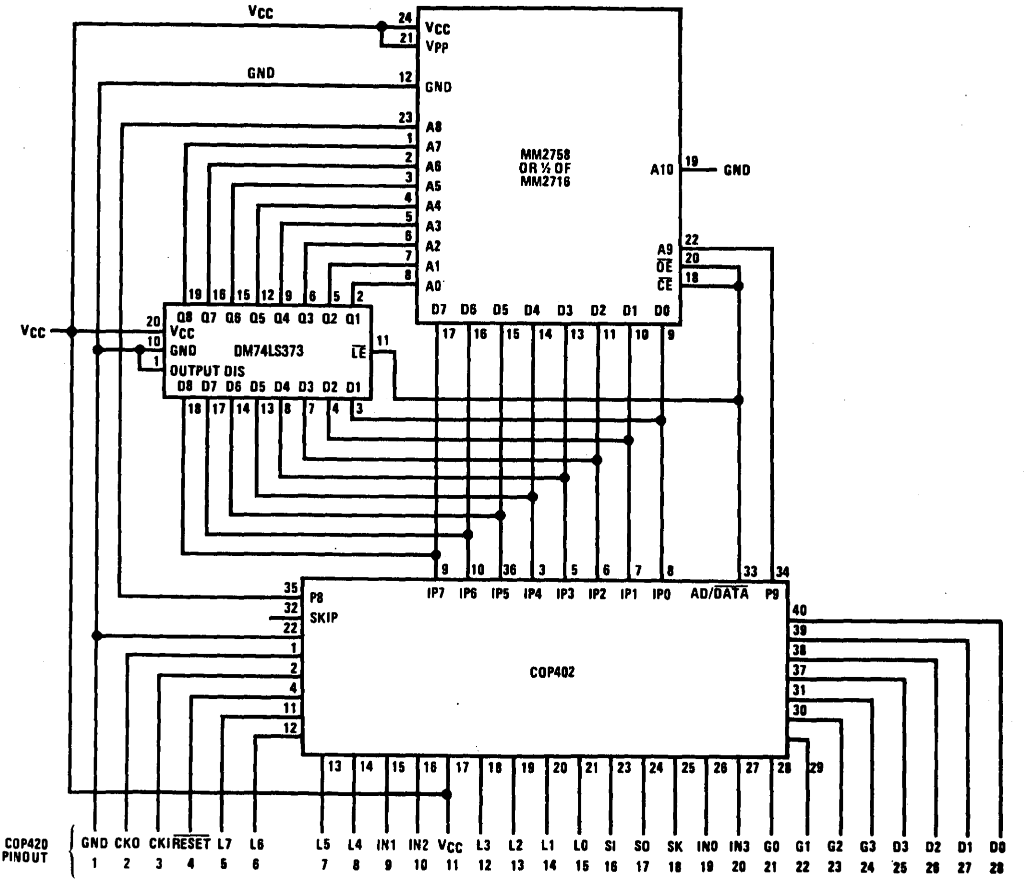
Typical Application: PROM-Based System

The COP402 may be used to exactly emulate the COP420, *Figure 13* shows the interconnect to implement a COP420 hardware emulation. This connection uses two MM5204 EPROMs as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.

When AD/$\overline{\text{DATA}}$ turns off, the EPROMs are enabled and the IP7-IP0 pins will input the memory data. P8 and P9 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

The other 28 pins of the COP402 may be configured exactly the same as a COP420. The COP402M chip can be used if the MICROBUS feature of the COP420 is needed.



TL/DD/6915–20

FIGURE 13. COP402 Used to Emulate a COP420

Option List

COP402 MASK OPTIONS

The following COP420 options have been implemented in this basic version of the COP402. Subsequent versions of the COP402 will implement different combinations of available options; such versions will be identified as COP402-A, COP402-B, etc.

Option Value	Comment
Option 1 = 0	Ground Pin—no option available
Option 2 = 0	CKO is clock generator output to crystal
Option 3 = 0	CKI is crystal input ÷ 16 (may be overridden externally)
Option 4 = 0	RESET pin has load device to V_{CC}
Option 5 = 2 (402) = 3 (402M)	L7 has LED direct-drive output L7 has TRI-STATE push-pull output
Option 6 = 2, 3	L6 same as L7
Option 7 = 2, 3	L5 same as L7
Option 8 = 2, 3	L4 same as L7
Option 9 = 0 (402) = 1 (402M)	IN1 has load device to V_{CC} Hi Z
Option 10 = 0 (402) = 1 (402M)	IN2 has load device to V_{CC} Hi Z
Option 11 = 0	V_{CC} pin—no option available
Option 12 = 2, 3	L3 same as L7
Option 13 = 2, 3	L2 same as L7
Option 14 = 2, 3	L1 same as L7

Option Value	Comment
Option 15 = 2, 3	L0 same as L7
Option 16 = 0	SI has load device to V_{CC}
Option 17 = 2	SO has push-pull output
Option 18 = 2	SK has push-pull output
Option 19 = 0	IN0 has load device to V_{CC}
Option 20 = 0 (402) = 1 (402M)	IN3 has load device to V_{CC} Hi Z
Option 21 = 0	G0 has standard output
Option 22 = 0	G1 same as G0
Option 23 = 0	G2 same as G0
Option 24 = 0	G3 same as G0
Option 25 = 0	D3 has standard output
Option 26 = 0	D2 same as D3
Option 27 = 0	D1 same as D3
Option 28 = 0	D0 same as D3
Option 29 = 0 (402) = 1 (402M)	normal operation MICROBUS operation
Option 30 = N/A	40-pin package

National
Semiconductor

COP404 ROMless N-Channel Microcontroller

General Description

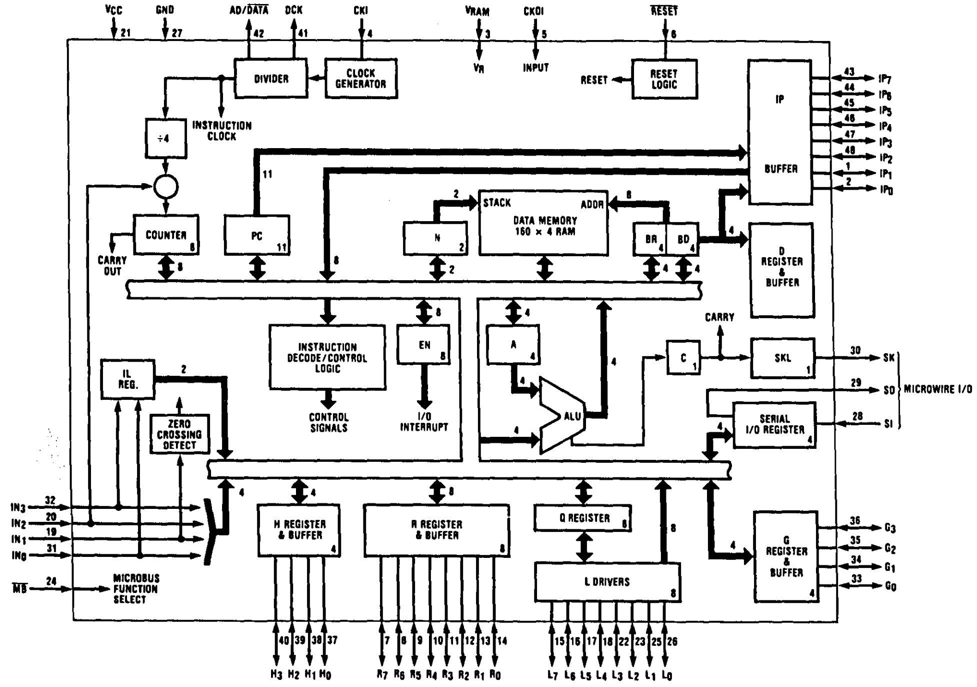
The COP404 ROMless N-Channel Microcontrollers are members of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. Each microcontroller contains all system timing, internal logic, RAM and I/O necessary to implement dedicated control functions in a variety of applications, and is identical to the COP440/COP340 devices, except that the ROM has been removed; pins have been added to output the ROM address and to input ROM data. In a system, the COP404 will perform exactly as the COP440; this important benefit facilitates development and debug of a COP440 program prior to masking the final part. Features include single supply operation, various output configurations, and an instruction set, internal architecture, and I/O scheme designed to facilitate keyboard input, display output and data manipulation. Standard test procedures and reliable high-density fabrication techniques provide the medium to large volume customers with a controller-oriented processor at a low end-product cost.

For extended temperature range (−40°C to +85°C) COP304 available on special order.

Features

- Exact circuit equivalent of COP440
- Standard 48-pin dual-in-line package
- Interfaces with standard PROM or ROM
- Enhanced, more powerful instruction set
- 160 × 4 RAM, addresses up to 2k × 8 ROM
- MICROBUS™ compatible
- Zero-crossing detect circuitry with hysteresis
- True multi-vectored interrupt from four selectable sources (plus restart)
- Four-level subroutine stack (in RAM)
- 4 μs cycle time
- Single supply operation (4.5V–6.3V)
- Programmable time-base counter for real-time processing
- Internal binary counter/register with MICROWIRE™ compatible serial I/O
- General purpose and TRI-STATE® outputs
- TTL/CMOS compatible in and out
- Software/hardware compatible with other members of COP400 family
- Compatible dual CPU device available

Block Diagram



TL/DD/6916–1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Zero-Crossing Detect Pin Relative to GND	−1.2V to +15V
Voltage at Any Other Pin Relative to GND	−0.5V to +7V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	150 mA
Total Sink Current	90 mA

Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics

0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 6.3V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 4)	4.5	6.3	V
Power Supply Ripple	(Peak to Peak)		0.4	V
Operating Supply Current	(All Inputs and Outputs Open)			
	T_A = 0°C		44	mA
	T_A = 25°C		37	mA
	T_A = 70°C		30	mA
V_R RAM Power Supply Current	V_R = 3.3V		3	mA
Input Voltage Levels				
CKI Input Levels (÷16)				
Logic High (V_{IH})	V_{CC} = Max.,	2.5		V
Logic High (V_{IH})	V_{CC} = 5V ±5%	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	(Schmitt Trigger Input)			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
Zero-Crossing Detect Input (IN_1)	Zero-Crossing Interrupt Input; INIL Instruction			
Trip Point		−0.15	0.15	V
Logic High (V_{IH}) Limit			12	V
Logic Low (V_{IL}) Limit		−0.8		V
IN_1				
Logic High	Interrupt Input;			
	ININ Instruction;	3.0		V
Logic Low	MICROBUS Input	−0.3	0.8	V
All Other Inputs				
Logic High	V_{CC} = Max.	2.5		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
IN_1 Input Resistance to Ground	V_{IH} = 1.0V	1.5	4.6	kΩ
Input Load Source Current	V_{IH} = 2.0V, V_{CC} = 4.5V	14	230	μA
Input Capacitance			7.0	pF
Hi-Z Input Leakage		−1.0	+1.0	μA
Output Voltage Levels				
Standard Output				
TTL Operation				
Logic High (V_{OH})	I_{OH} = −100 μA	2.4		V
Logic Low (V_{OL})	I_{OL} = 1.6 mA		0.4	V
CMOS Operation (Note 1)				
Logic High (V_{OH})	I_{OH} = −10 μA	V_{CC} − 0.4		V
Logic Low (V_{OL})	I_{OL} = 10 μA		0.2	V
TRI-STATE Output				
TTL Operation				
Logic High (V_{OH})	I_{OH} = −100 μA	2.4		V
Logic Low (V_{OL})	I_{OL} = 1.6 mA		0.4	V
CMOS Operation (Note 1)	33 kΩ ≥ R_L ≥ 4.7 kΩ			
Logic High (V_{OH})	I_{OH} = −10 μA	V_{CC} − 0.5		V
Logic Low (V_{OL})	I_{OL} = 1.6 mA		0.4	V
Output Current Levels				
Standard Output Source Current	V_{CC} = 4.5V, V_{OH} = 2.4V	−100	−650	μA
TRI-STATE Output Leakage Current		−2.5	+2.5	μA

DC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted (Continued)

Parameter	Conditions	Min	Max	Units
Total Sink Current Allowed				
All I/O Combined			90	mA
Each L, R Port			20	mA
Each D, G, H Port			10	mA
SO, SK			2.5	mA
IP			1.8	mA
Total Source Current Allowed	(Note 5)			
All I/O Combined			150	mA
L Port			120	mA
L_7–L_4			70	mA
L_3–L_0			70	mA
Each L Pin			23	mA
All Other Output Pins			1.6	mA

Note 1: TRI-STATE configuration is excluded.

AC Electrical Characteristics $0°C \le T_A \le +70°C$, $4.5V \le V_{CC} \le 6.3V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time—t_E		4.0	10	μs
CKI Frequency	÷16 Mode	1.6	4.0	MHz
Duty Cycle (Note 2)	f_I = 4 MHz	30	60	%
Rise Time	f_I = 4 MHz		60	ns
Fall Time	f_I = 4 MHz		40	ns
INPUTS: *(Figure 3)*				
SI				
t_{SETUP}		0.3		μs
t_{HOLD}		300		ns
IP				
t_{SETUP}		0.25		μs
t_{HOLD}		250		ns
t_{HOLD}	From AD/$\overline{DATA}$ Rising Edge	0		ns
All Other Inputs				
t_{SETUP}		1.7		μs
t_{HOLD}		300		ns
OUTPUT PROPAGATION DELAY	Test Condition:			
IP	C_L = 50 pF, V_{OUT} = 1.5V			
t_{pd1A}, t_{pd0A}			1.94	μs
t_{pd1B}, t_{pd0B}			0.94	μs
DCK				
t_{pd1}, t_{pd0}			375	ns
AD/$\overline{DATA}$				
t_{pd1}, t_{pd0}			300	ns
SO, SK				
t_{pd1}, t_{pd0}	R_L = 2.4 kΩ		1.0	μs
All Other Outputs	R_L = 5.0 kΩ		1.4	μs
MICROBUS TIMING	C_L = 100 pF, V_{CC} = 5V±5%			
Read Operation	TRI-STATE outputs			
Chip Select Stable Before $\overline{RD}$—t_{CSR}		65		ns
Chip Select Hold Time for $\overline{RD}$—t_{RCS}		20		ns
$\overline{RD}$ Pulse Width—t_{RR}		400		ns
Data Delay from $\overline{RD}$—t_{RD}			375	ns
$\overline{RD}$ to Data Floating—t_{DF}			250	ns
Write Operation				
Chip Select Stable Before $\overline{WR}$—t_{CSW}		65		ns
Chip Select Hold Time for $\overline{WR}$—t_{WCS}		20		ns
$\overline{WR}$ Pulse Width—t_{WW}		400		ns
Data Set-Up Time for $\overline{WR}$—t_{DW}		320		ns
Data Hold Time for $\overline{WR}$—t_{WD}		100		ns
INTR Transition Time from $\overline{WR}$—t_{WI}			700	ns

Note 2: Duty Cycle = $t_{WI}/(t_{WI} + t_{WO})$.

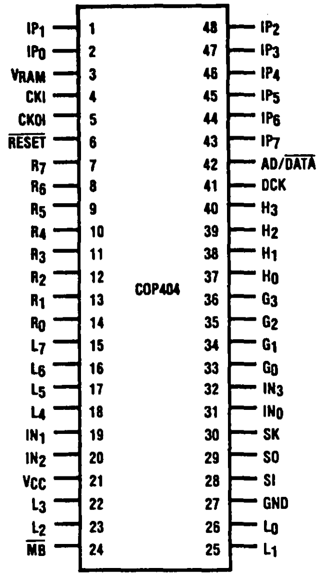
Note 3: See Figure for additional I/O Characteristics.

Note 4: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Note 5: Exercise great care not to exceed maximum device power dissipation limits when direct-driving LEDs (or sourcing similar loads) at high temperature.

Connection Diagram

Dual-In-Line Package



TL/DD/6916-2

Top View

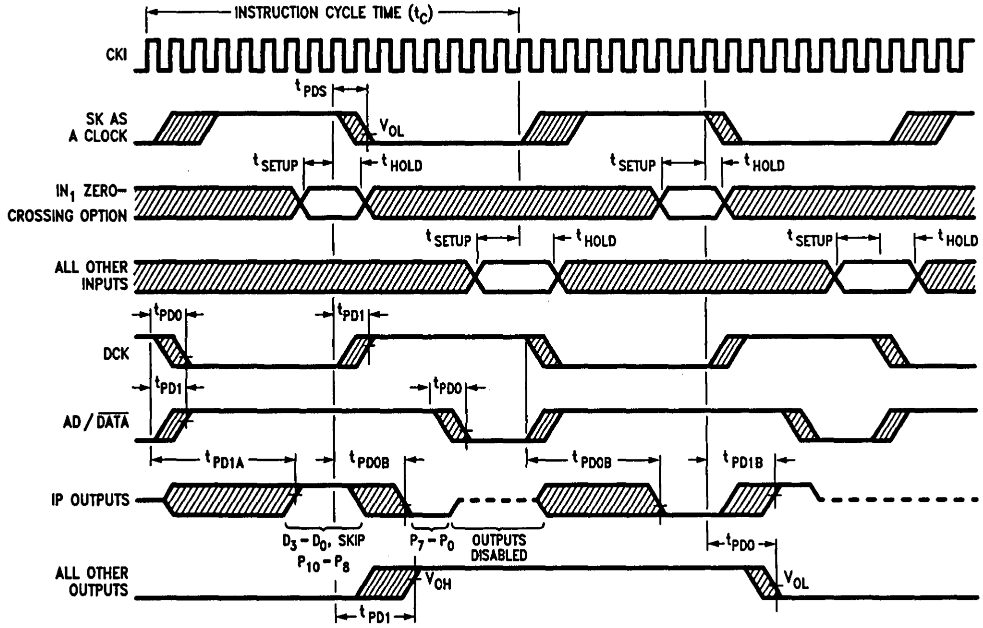
FIGURE 2

Order Number COP404N
See NS Package Number N48A

Pin Descriptions

Pin	Description
L_7–L_0	8-bit bidirectional TRI-STATE I/O port
G_3–G_0	4-bit bidirectional I/O port
IN_3–IN_0	4-bit general purpose input port
H_3–H_0	4-bit bidirectional I/O port.
R_7–R_0	8-bit bidirectional TRI-STATE I/O port
SI	Serial input
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
CKI	System oscillator input
CKOI	General purpose input
V_{RAM}	Power supply to first 4 registers of RAM
$\overline{MB}$	MICROBUS function select
DCK	Clock output to latch D outputs and high order address bits
AD/$\overline{DATA}$	Address out/data in flag
IP_1–IP_0	8-bit bidirectional port for ROM address, ROM data and D outputs
$\overline{RESET}$	System reset input
V_{CC}	Power Supply
GND	Ground

Timing Diagram



TL/DD/6916-3

FIGURE 3. Input/Output Timing Diagrams (÷ 16 Mode)

Functional Description

The COP404 is a ROMless microcontroller for emulating the COP440 or for stand-alone applications. Please refer to the COP440 description for detail functional description. The following describes functions that are unique to the COP404 or are different from those in COP440. *Figures 1* and *2* show the COP404 block diagram and pin-out.

PROGRAM MEMORY

Program memory consists of 2048 bytes of external memory (on-chip in the COP440) that can be accessed through the IP port. See External Memory Interface below.

D PORT

The D3–D0 outputs are missing from this 48-pin package, but may be recovered through the IP port (see External Memory Interface below). Note that the recovered signals have the same timing but different output drive capability as those from the COP440 (see D Port Characteristics below).

MICROBUS AND ZERO-CROSSING DETECT INPUT OPTION

The MICROBUS compatible I/O, selected by a mask option on the COP440, is selected by tying the $\overline{\text{MB}}$ pin directly to ground. When the MICROBUS compatible I/O is not desired, the $\overline{\text{MB}}$ pin should be tied to V_{CC}. Note that none of the IN inputs are Hi-Z. Since zero-crossing detect input (used by INIL instruction and zero-crossing interrupt feature) is chosen for IN1, the IN1 input "1" level for ININ instruction, IN1 interrupt, and MICROBUS input is 3V. Even though the MICROBUS option and zero-crossing detector option appear on the COP404, they are mutually exclusive on the COP440.

OSCILLATOR

CKI is an external clock input signal. The clock frequency is divided by 16 to give the execution frequency.

CKO PIN OPTIONS

Two different CKO functions of the COP440 are available on the COP404. V_{RAM} supplies power to the lower four registers of RAM, and CKOI is an interrupt input or a general purpose input, reading into bit 2 of A (accumulator) through the INIL instruction.

EXTERNAL MEMORY INTERFACE

The COP404 is designed for use with an external program memory. This memory may be implemented using any devices having the following characteristics:

1. Random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL-compatible inputs
4. Access time = 450 ns maximum

Typically these requirements are met using bipolar or MOS PROMs.

Figure 3 shows the timings for IP port and the external memory interface clocks—DCK and AD/$\overline{\text{DATA}}$. While DCK is low, the upper three address bits, P10–P8, of the next instruction to be executed appear at IP2–IP0 respectively; D3–D0 appear at IP7–IP4 and IP3 contains the SKIP output used by the COPS Program Development System (PDS). The rising edge of DCK clocks these data into D flip-flops, e.g., 74LS374. The timing of D port data is then the same for COP404 and COP440. After DCK has risen to a "1" level, the remaining address bits (P7–P0) appear at IP7–IP0. The falling edge of AD/$\overline{\text{DATA}}$ latches these data into flow-through latches, e.g., 74LS373. The latched addresses provide the inputs to the external memory. When AD/$\overline{\text{DATA}}$ goes low, the IP outputs are disabled and the IP lines become program memory inputs from the external memory. Note that DCK has a duty cycle of about 50% and AD/$\overline{\text{DATA}}$ has a duty cycle of about 75%. *Figure 4* shows how to emulate the COP440 using a COP404 and an EPROM as the external memory.

I/O OPTIONS

All inputs except IN1 and CKI have on-chip depletion load devices to V_{CC}. IN1 has a resistive load to GND due to the zero-crossing input. CKI is a Hi-Z input.

G and H ports have standard outputs. L and R ports have TRI-STATE outputs. IP port, DCK, AD/$\overline{\text{DATA}}$, SO and SK have push-pull outputs.

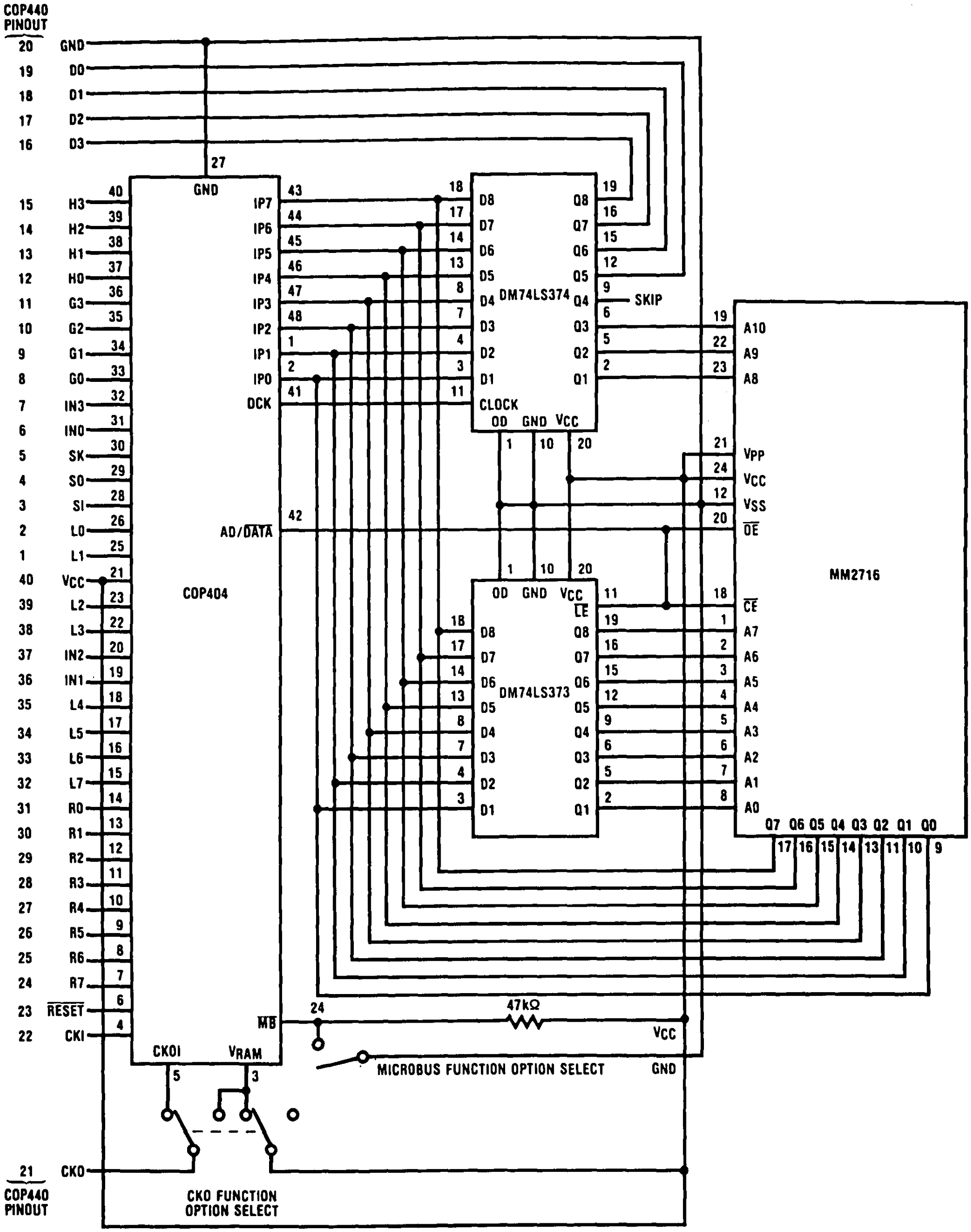
LED DRIVE

The TRI-STATE outputs of L port may be used to drive the segments of an LED display. External current limiting resistors of 100Ω must be connected between the L outputs and the LED segments.

D PORT CHARACTERISTICS

Since the D port is recovered through an external latch, the output drive is that of the latch and not that of COP440. Using the set-up as shown in *Figure 4*, at an output "0" level of 0.4V, the 74LS374 may sink 10 times as much current as the COP440. At an output "1" level of 2.4V, the 74LS374 may source 10 times as much current as the COP440. On the other hand, the output "1" level of 74LS374 latch does not go to V_{CC} without an external pull-up resistor. In order to better approximate the COP440 output characteristics, add a 74C906 buffer to the output of the 74LS374, thus emulating an open drain D output. A pull-up resistor of 10k should be added to the input of the buffer. To emulate the standard output, add a pull-up resistor between 2.7k and 15k to the output of the 74C906.

Functional Description (Continued)



TL/DD/6916-4

FIGURE 4. COP404 Used to Emulate a COP440

Option Table

COP404 MASK OPTIONS

The following COP440 options have been implemented in the COP404.

Option Value		Comment
Option 1–2	= 3	L outputs are TRI-STATE
Option 3	= 0	SI has load to V_{CC}
Option 4	= 2	SO is push-pull output
Option 5	= 2	SK is push-pull output
Option 6	= 0	IN0 has load to V_{CC}
Option 7	= 0	IN3 has load to V_{CC}
Option 8–11	= 0	G outputs are standard
Option 12–15	= 0	H outputs are standard
Option 16–19	= N/A	D outputs are derived from external latch, see *Figure 4*
Option 20	= N/A	GND—No option
Option 21	= 1,2	CKO is replaced by V_{RAM} and CKOI

Option Value		Comment
Option 22	= 0	CKI is input clock divided by 16
Option 23	= 0	$\overline{\text{RESET}}$ has load to V_{CC}
Option 24–31	= 3	R outputs are TRI-STATE
Option 32–35	= 3	L outputs are TRI-STATE
Option 36	= 2	IN1 is zero-crossing detect input
Option 37	= 0	IN2 has load to V_{CC}
Option 38–39	= 3	L outputs are TRI-STATE
Option 40	= N/A	V_{CC}—No option available
Option 41	= 0,1	MICROBUS option is pin selectable
Option 42–48	= 0	Inputs have standard TTL levels
Option 49	= N/A	No option available
Option 50	= N/A	48-pin package

National Semiconductor

COP404C ROMless CMOS Microcontrollers

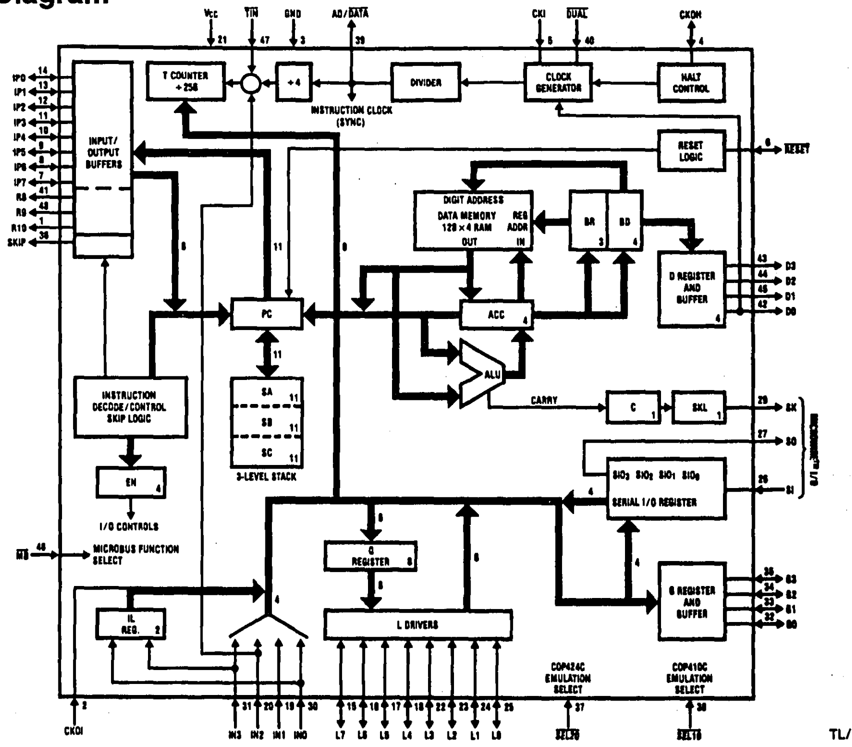
General Description

The COP404C ROMless Microcontroller is a member of the COPS™ family, fabricated using double-poly, silicon gate CMOS (microCMOS) technology. The COP404C contains CPU, RAM, I/O and is identical to a COP444C device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. The COP404C can be configured, by means of external pins, to function as a COP444C, a COP424C, or a COP410C. Pins have been added to allow the user to select the various functional options that are available on the family of mask-programmed CMOS parts. The COP404C is primarily intended for use in the development and debug of a COP program for the COP444C/445C, COP424C/425C, and COP410C/411C devices prior to masking the final part. The COP404C is also appropriate in low volume applications or when the program might be changing.

Features

- Accurate emulation of the COP444C, COP424C and COP410C
- Lowest Power Dissipation (50 μW typical)
- Fully static (can turn off the clock)
- Power saving IDLE state and HALT mode
- 4 μs instruction time, plus software selectable clocks
- 128 × 4 RAM, addresses 2k × 8 ROM
- True vectored interrupt, plus restart
- Three-level subroutine stack
- Single supply operation (2.4V to 5.5V)
- Programmable read/write 8-bit timer/event counter
- Internal binary counter register with MICROWIRE™ serial I/O capability
- General purpose and TRI-STATE® outputs
- LSTTL/CMOS compatible
- MICROBUS™ compatible
- Software/hardware compatible with other members of the COP400 family

Block Diagram



TL/DD/5530-1

FIGURE 1. Block Diagram

Absolute Maximum Ratings

Supply Voltage	6V	Operating temperature range	0° to +70°C
Voltage at any pin	$-0.3V$ to $V_{CC} + 0.3V$	Storage temperature range	−65°C to +150°C
Total Allowable Source Current	25 mA	Lead temperature (soldering, 10 sec.)	300°C
Total Allowable Sink Current	25 mA		

DC Electrical Characteristics $0°C \le T_a \le 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		2.4	5.5	V
Power Supply Ripple	peak to peak		$0.1\ V_{CC}$	V
(Note 5)				
Supply Current	$V_{CC}=2.4V$, $t_c=64\ \mu s$		120	μA
(Note 1)	$V_{CC}=5.0V$, $t_c=16\ \mu s$		700	μA
	$V_{CC}=5.0V$, $t_c=4\ \mu s$		3000	μA
	(T_c is instruction cycle time)			
HALT Mode Current	$V_{CC}=5.0V$, $F_{IN}=0$ kHz, $T_A=25°C$		20	μA
(Note 2)	$V_{CC}=2.4V$, $F_{IN}=0$ kHz, $T_A=25°C$		6	μA
Input Voltage Levels				
$\overline{\text{RESET}}$, D0 (clock input)				
CKI				
Logic High		$0.9\ V_{CC}$		V
Logic Low			$0.1\ V_{CC}$	V
All other inputs (Note 7)				
Logic High		$0.7\ V_{CC}$		V
Logic Low			$0.2\ V_{CC}$	V
Input Pull-up current	$V_{CC}=4.5V$, $V_{IN}=0$	30	330	μA
Hi-Z input leakage		−1	+1	μA
Input capacitance			7	pF
(Note 4)				
Output Voltage Levels	Standard outputs			
LSTTL Operation	$V_{CC}=5.0V \pm 10\%$			
Logic High	$I_{OH}=-100\ \mu A$	2.7		V
Logic Low	$I_{OL}=400\ \mu A$		0.4	V
CMOS Operation				
Logic High	$I_{OH}=-10\ \mu A$	$V_{CC}-0.2$		V
Logic Low	$I_{OL}=10\ \mu A$		0.2	V
Output current levels				
Sink (Note 6)	$V_{CC}=4.5V$, $V_{OUT}=V_{CC}$	1.2		mA
	$V_{CC}=2.4V$, $V_{OUT}=V_{CC}$	0.2		mA
Source (Standard option)	$V_{CC}=4.5V$, $V_{OUT}=0V$	0.5		mA
	$V_{CC}=2.4V$, $V_{OUT}=0V$	0.1		mA
Source (Low current option)	$V_{CC}=4.5V$, $V_{OUT}=0V$	30	330	μA
	$V_{CC}=2.4V$, $V_{OUT}=0V$	6	80	μA
Allowable Sink/Source current per pin (Note 6)			5	mA
Allowable Loading on CKOH			100	pF
Current needed to over-ride HALT				
(Note 3)				
To continue	$V_{CC}=4.5V$, $V_{IN}=2V_{CC}$		.7	mA
To halt	$V_{CC}=4.5V$, $V_{IN}=7V_{CC}$		1.6	mA
TRI-STATE leakage current		−2.5	+2.5	μA

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

COP404C

AC Electrical Characteristics $0°C \leq T_A \leq 70°C$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle	$V_{CC} \geq 4.5V$	4	DC	µs
Time (t_c)	$4.5V > V_{CC} \geq 2.4V$	16	DC	µs
Operating CKI	$V_{CC} \geq 4.5V$	DC	1.0	MHz
Frequency	$4.5V > V_{CC} \geq 2.4V$	DC	250	kHz
Duty Cycle (Note 4)	$f_1 = 4$ MHz	40	60	%
Rise Time (Note 4)	$f_1 = 4$ MHz external clock		60	ns
Fall Time (Note 4)	$f_1 = 4$ MHz external clock		40	ns
Instruction Cycle	$R = 30k$, $V_{CC} = 5V$			
Time using D0 as a	$C = 82$ pF	8	16	µs
RC Oscillator Dual-				
Clock Input (Note 4)				
INPUTS: (See *Fig. 3*)				
t_{SETUP}	G Inputs	$T_c/4 + .7$		µs
	SI Input $\quad V_{CC} \geq 4.5V$	0.3		µs
	IP Input	1.0		µs
	All Others	1.7		µs
t_{HOLD}	$V_{CC} \geq 4.5V$	0.25		µs
	$4.5V > V_{CC} \geq 2.4V$	1.0		µs
OUTPUT PROPAGATION DELAY	$V_{OUT} = 1.5V$, $C_L = 100$ pF, $R_L = 5K$			
IP7–IP0, A10–A8, SKIP				
t_{PD1}, t_{PD0}	$V_{CC} \geq 4.5V$		1.94	µs
	$4.5V > V_{CC} \geq 2.4V$		7.75	µs
AD/$\overline{DATA}$				
t_{PD1}, t_{PD0}	$V_{CC} \geq 4.5V$		375	ns
	$4.5V > V_{CC} \geq 2.4V$		1.5	µs
ALL OTHER OUTPUTS				
t_{PD1}, t_{PD0}	$V_{CC} > 4.5V$		1.0	µs
	$4.5V > V_{CC} \geq 2.4V$		4.0	µs
MICROBUS TIMING Read Operation (*Fig. 4*)	$C_L = 50$ pF, $V_{CC} = 5V \pm 5\%$			
Chip select stable before $\overline{RD}$ $-t_{CSR}$		65		ns
Chip select hold time for $\overline{RD}$ $-t_{RCS}$		20		ns
$\overline{RD}$ pulse width $-t_{RR}$		400		ns
Data delay from $\overline{RD}$ $-t_{RD}$			375	ns
$\overline{RD}$ to data floating $-t_{DF}$ (Note 4)			250	ns
Write Operation (*Fig. 5*)				
Chip select stable before $\overline{WR}$ $-t_{CSW}$		65		ns
Chip select hold time for $\overline{WR}$ $-t_{WCS}$		20		ns
$\overline{WR}$ pulse width $-t_{WW}$		400		ns
Data set-up time for $\overline{WR}$ $-t_{DW}$		320		ns
Data hold time for $\overline{WR}$ $-t_{WD}$		100		ns
INTR transition time from $\overline{WR}$ $-t_{WI}$			700	ns

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI and all other pins pulled up to V_{CC} with 20k resistors. See current drain equation on page 16.

Note 2: Test conditions: All inputs tied to V_{CC}; L lines in TRI-STATE mode and tied to Ground; all outputs tied to Ground.

Note 3: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 4: This parameter is only sampled and not 100% tested. Variation due to the device included.

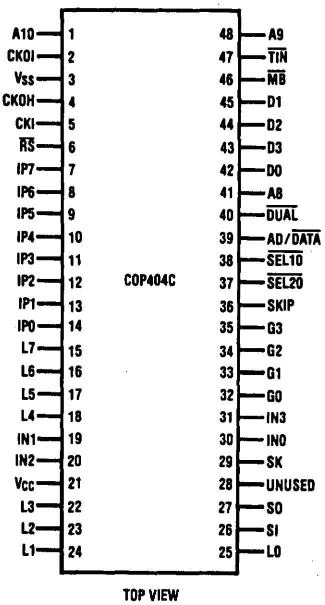
Note 5: Voltage change must be less than 0.5 volts in a 1 ms period.

Note 6: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} to prevent entering test mode.

Note 7: $\overline{MB}$, $\overline{TIN}$, $\overline{DUAL}$, $\overline{SEL10}$, $\overline{SEL20}$, input levels at V_{CC} or V_{SS}.

Connection Diagram

Dual-In-Line Package



TL/DD/5530-2

Order Number COP404CN
See NS Package Number N48A

Pin Descriptions

Pin	Description
V_{CC}	Most positive voltage
V_{SS}	Ground
CKI	Clock input
$\overline{RS}$	Reset input
CKOI	General purpose input
L0–L7	8 TRI-STATE I/O
G0–G3	4 general purpose I/O
D1–D3	3 general purpose outputs
D0	Either general purpose output or Dual-Clock RC input
IN0–IN3	4 general purpose inputs
SO	Serial data output
SI	Serial data input
SK	Serial data clock output
IP0–IP7	I/O for ROM address and data
A8, A9, A10	3 address outputs
SKIP	Skip status output
AD/$\overline{DATA}$	Clock output
$\overline{MB}$	MICROBUS select input
CKOH	Halt I/O pin
$\overline{DUAL}$	Dual-Clock select input
$\overline{TIN}$	Timer input select pin
$\overline{SEL10}$	COP410C emulation select input
$\overline{SEL20}$	COP424C emulation select input
UNUSED	Ground

FIGURE 2

The internal architecture is shown in *Figure 1*. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1", when a bit is reset, it is a logic "0".

PROGRAM MEMORY

Program Memory consists of a 2048-byte external memory (typically PROM). Words of this memory may be program instructions, constants or ROM addressing data.

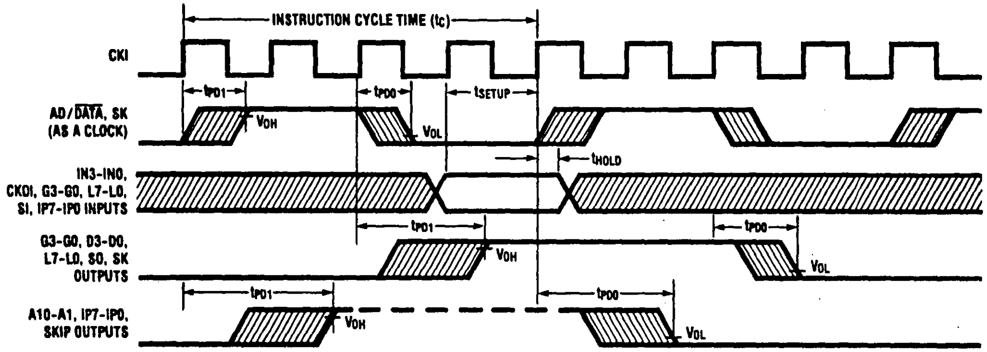
ROM addressing is accomplished by a 11-bit PC register which selects one of the 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value.

Three levels of subroutine nesting are implemented by a three level deep stack. Each subroutine call or interrupt pushes the next PC address into the stack. Each return pops the stack back into the PC register.

DATA MEMORY

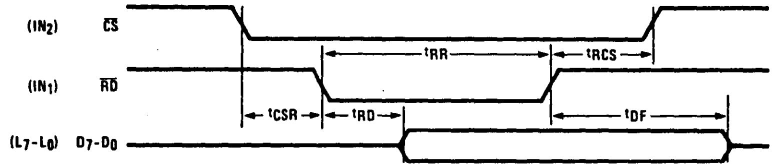
Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 × 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (B_r) select 1 of 8 data registers and lower 4 bits (B_d) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) are usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or T counter or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the immediate operand field of these instructions. The B_d register also serves as a source register for 4-bit data sent directly to the D outputs.

Timing Diagrams



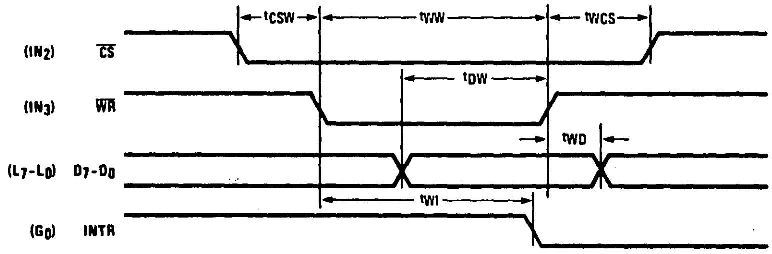
TL/DD/5530-3

FIGURE 3. Input/Output Timing



TL/DD/5530-4

FIGURE 4. MICROBUS Read Operation Timing



TL/DD/5530-5

FIGURE 5. MICROBUS Write Operation Timing

Functional Description

INTERNAL LOGIC

The processor contains its own 4-bit A register (accumulator) which is the source and destination register for most I/O, arithmetic, logic, and data memory access operations. It can also be used to load the B_r and B_d portions of the B register, to load and input 4 bits of the 8-bit Q latch or T counter, L I/O ports data, to input 4-bit G, or IN ports, and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing the results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register in conjunction with the XAS instruction and the EN register, also serves to control the SK output.

The 8-bit T counter is a binary up counter which can be loaded to and from M and A using CAMT and CTMA instructions. This counter may be operated in two modes: as a timer if $\overline{\text{TIN}}$ pin is tied to Ground or as an external event counter if $\overline{\text{TIN}}$ pin is tied to V_{CC}. When the T counter overflows, an overflow flag will be set (see SKT and IT instructions below). The T counter is cleared on reset. A functional block diagram of the timer/counter is illustrated in *Figure 10a*.

Four general-purpose inputs, IN3–IN0, are provided. IN1, IN2 and IN3 may be selected (by pulling $\overline{\text{MB}}$ pin low) as Read Strobe, Chip Select, and Write Strobe inputs, respectively, for use in MICROBUS application.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of B_d. In the dual clock mode, D0 latch controls the clock selection (see dual oscillator below).

The G register contents are outputs to a 4-bit general-purpose bidirectional I/O port. G0 may be selected as an output for MICROBUS applications.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are outputted to the L I/O ports when the L drivers are enabled under program control. With the MICROBUS option selected, Q can also be loaded with the 8-bit contents of the L I/O ports upon the occurrence of a write strobe from the host CPU.

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O port. Also, the contents of L may be read directly into A and M. As explained above, the MICROBUS option allows L I/O port data to be latched into the Q register.

The SIO register functions as a 4-bit serial-in/serial-out shift register for MICROWIRE™ I/O and COPS peripherals, or as a binary counter (depending on the contents of the EN register). Its contents can be exchanged with A.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

EN is an internal 4-bit register loaded by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register:

0. The least significant bit of the enable register, EN0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN0 set, SIO is an asynchronous binary counter, decrementing its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output equals the value of EN3. With EN0 reset, SIO is a serial shift register left shifting 1 bit each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. The SK outputs SKL ANDed with the instruction cycle clock.
1. With EN1 set, interrupt is enabled. Immediately following an interrupt, EN1 is reset to disable further interrupts.
2. With EN2 set, the L drivers are enabled to output the data in Q to the L I/O port. Resetting EN2 disables the L drivers, placing the L I/O port in a high-impedance input state.
3. EN3, in conjunction with EN0, affects the SO output. With EN0 set (binary counter option selected) SO will output the value loaded into EN3. With EN0 reset (serial shift register option selected), setting EN3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains set to "0".

INTERRUPT

The following features are associated with interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once recognized as explained below, pushes the next sequential program counter address (PC+1) onto the stack. Any previous contents at the bottom of the stack are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN1 is reset.

b. An interrupt will be recognized only on the following conditions:

   1. EN1 has been set.
   2. A low-going pulse ("1" to "0") at least two instruction cycles wide has occurred on the IN1 input.
   3. A currently executing instruction has been completed.

TABLE I. ENABLE REGISTER MODES — BITS EN0 AND EN3

EN0	EN3	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL=1, SK=clock If SKL=0, SK=0
0	1	Shift Register	Input to Shift Register	Serial out	If SKL=1, SK=clock If SKL=0, SK=0
1	0	Binary Counter	Input to Counter	0	SK = SKL
1	1	Binary Counter	Input to Counter	1	SK = SKL

Functional Description (Continued)

4. All successive transfer of control instructions and successive LBIs have been completed (e.g. if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed).

c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of an ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the end of the interrupt routine, a RET instruction is executed to pop the stack and return program control to the instruction following the original ASC. At this time, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines should not be nested within the interrupt service routine, since their popping of the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

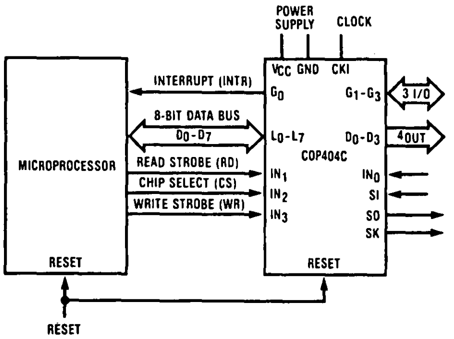
d. The instruction at hex address 0FF must be a NOP.

e. An LEI instruction may be put immediately before the RET instruction to re-enable interrupts.

MICROBUS INTERFACE

With $\overline{\text{MB}}$ pin tied to Ground, the COP404C can be used as a peripheral microprocessor device, inputting and outputting data from and to a host microprocessor (μP). IN1, IN2 and IN3 general purpose inputs become MICROBUS compatible read-strobe, chip-select, and write-strobe lines, respectively. IN1 becomes $\overline{\text{RD}}$ — a logic "0" on this input will cause Q latch data to be enabled to the L ports for input to the μP. IN2 becomes $\overline{\text{CS}}$ — a logic "0" on this line selects the COP404C and the μP peripheral device by enabling the operation of the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ lines and allows for the selection of one of several peripheral components. IN3 becomes $\overline{\text{WR}}$ — a logic "0" on this line will write bus data from the L ports to the Q latches for input to the COP404C. G0 becomes INTR a "ready" output, reset by a write pulse from the μP on the $\overline{\text{WR}}$ line, providing the "handshaking" capability necessary for asynchronous data transfer between the host CPU and the COP404C.

This option has been designed for compatibility with National's MICROBUS - a standard interconnect system for 8-bit parallel data transfer between MOS/LSI CPUs and interfacing devices. (See MICROBUS National Publication). The functioning and timing relationships between the signal lines affected by this option are as specified for the MICROBUS interface, and are given in the AC electrical characteristics and shown in the timing diagrams (*Figures 4* and *5*). Connection of the COP404C to the MICROBUS is shown in *Figure 6*.



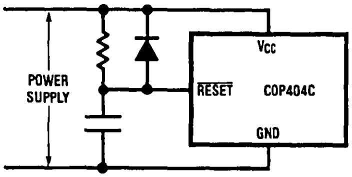
TL/DD/5530-7

FIGURE 6. MICROBUS Option Interconnect

INITIALIZATION

The external RC network shown in *Figure 7* must be connected to the $\overline{\text{RESET}}$ pin for the internal reset logic to initialize the device upon power-up. The $\overline{\text{RESET}}$ pin is configured as a Schmitt trigger input. If not used, it should be connected to V_{CC}. Initialization will occur whenever a logic "0" is applied to the $\overline{\text{RESET}}$ input, providing it stays low for at least three instruction cycle times.

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, IL, T and G registers are cleared. The SKL latch is set, thus enabling SK as a clock output. Data Memory (RAM) is not cleared upon initialization. The first instruction at address 0 must be a CLRA (clear A register).



RC ≥ 5X POWER SUPPLY RISE TIME AND RC ≥ 100X CKI PERIOD.

TL/DD/5530-8

FIGURE 7. Power-Up Circuit

TIMER

There are two modes selected by $\overline{\text{TIN}}$ pin:

a) Time-base counter ($\overline{\text{TIN}}$ pin low). In this mode, the instruction cycle frequency generated from CKI passes through a 2-bit divide-by-4 prescaler. The output of this prescaler increments the 8-bit T counter thus providing a 10-bit timer. The prescaler is cleared during execution of a CAMT instruction and on reset. For example, using a 1MHz crystal, the instruction cycle frequency of 250 kHz (divide by 4) increments the 10-bit timer every 4 μS. By presetting the counter and detecting overflow, accurate timeouts between 16 μS (4 counts) and 4.096 mS (1024 counts) are possible. Longer timeouts can be achieved by accumulating, under software control, multiple overflows.

b) External event counter ($\overline{\text{TIN}}$ pin high). In this mode, a low-going pulse ("1" to "0") at least 2 instruction cycles wide on the IN2 input will increment the 8-bit T counter.

Note: the IT instruction is not allowed in this mode.

HALT MODE

The COP404C is a FULLY STATIC circuit; therefore, the user may stop the system oscillator at any time to halt the chip. The chip may also be halted by two other ways (see *Figure 8*):

— Software HALT: by using the HALT instruction.

— Hardware HALT: by using the HALT I/O port CKOH. It is an I/O flip-flop which is an indicator of the HALT status. An external signal can over-ride this pin to start and stop the chip. By forcing CKOH high the

Functional Description (Continued)

chip will stop as soon as CKI is high and CKOH output will stay high to keep the chip stopped if the external driver returns to high impedance state.

Once in the HALT mode, the internal circuitry does not receive any clock signal and is therefore frozen in the exact state it was in when halted. All information is retained until continuing.

The chip may be awakened by one of two different methods:

— Continue function: by forcing CKOH low, the system clock will be re-enabled and the circuit will continue to operate from the point where it was stopped. CKOH will stay low.

— Restart: by forcing the $\overline{\text{RESET}}$ pin low (see Initialization)

The HALT mode is the minimum power dissipation state.

Note: if the user has selected dual-clock ($\overline{\text{DUAL}}$ pin tied to Ground) AND is forcing an external clock on D0 pin AND the COP404C is running from the D0 clock, the HALT mode - either hardware or software - will NOT be entered. Thus, the user should switch to the CKI clock to HALT. Alternatively, the user may stop the D0 clock to minimize power.

Oscillator Options

There are two basic clock oscillator configurations available as shown by *Figure 9.*

— CKI oscillator: CKI is configured as a LSTTL compatible input external clock signal. The external frequency is divided by 4 to give the instruction cycle time.

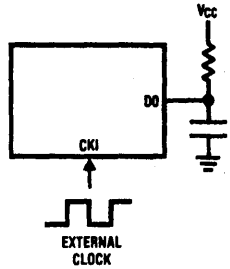
— Dual oscillator. By tying $\overline{\text{DUAL}}$ pin to Ground, pin D0 is now a single pin RC controlled Schmitt trigger oscillator input. The user may software select between the D0 oscillator (the instruction cycle time equals the D0 oscillation frequency divided by 4) by setting the D0 latch high or the CKI oscillator by resetting D0 latch low.

Note that even in dual clock mode, the counter, if used as a time-base counter, is always connected to the CKI oscillator.

For example, the user may connect up to a 1 MHz RC circuit to D0 for faster processing and a 32 kHz external clock to CKI for minimum current drain and time keeping.

Note: CTMA instruction is not allowed when the chip is running from D0 clock.

Figures 10a and *10b* show the timer and clock diagrams with and without Dual-Clock.

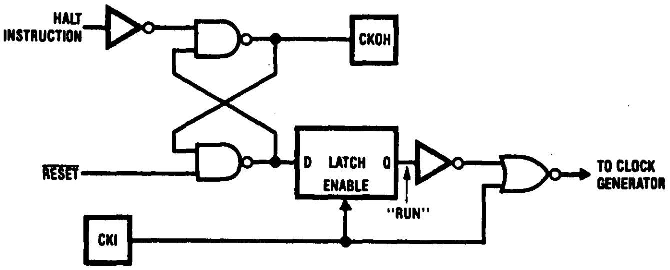


TL/DD/5530-9

R	C	Cycle Time	V_{CC}
15k	82 pF	4–9 μs	≥4.5V
30k	82 pF	8–16 μs	≥4.5V
60k	100 pF	16–32 μs	2.4–4.5V

Note: 15k≤R≤150k
50 pF≤C≤150 pF

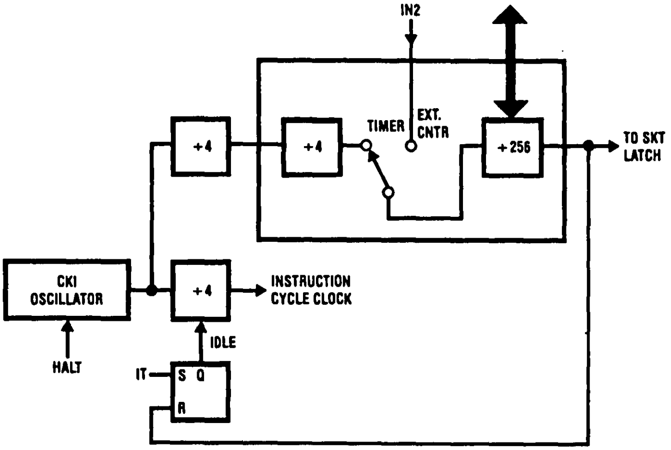
FIGURE 9. Dual-Oscillator Component Values



TL/DD/5530-10

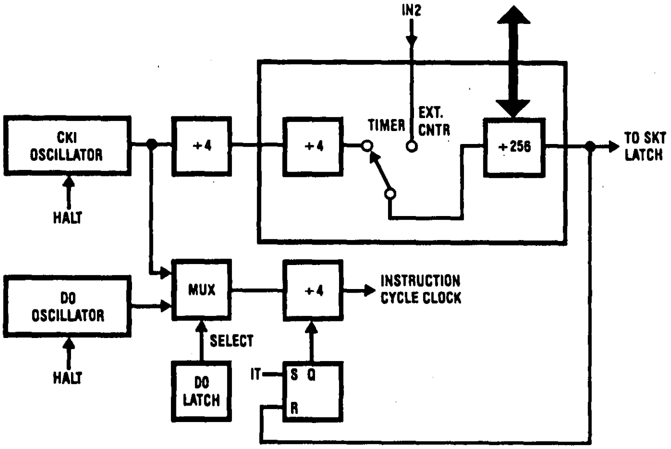
FIGURE 8. HALT Mode

Functional Description (Continued)



TL/DD/5530-11

FIGURE 10a. Clock and Timer Block Diagram without Dual-Clock



TL/DD/5530-12

Figure 10b. Clock and Timer Block Diagram with Dual-Clock

External Memory Interface

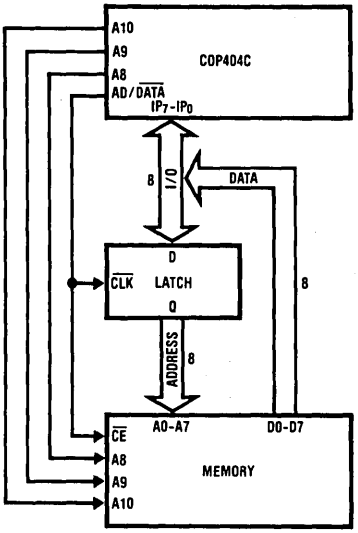
The COP404C is designed for use with an external Program Memory.

This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. LSTTL or CMOS-compatible TRI-STATE outputs
3. LSTTL or CMOS-compatible inputs
4. access time = 1. 0 µs max.

Typically, these requirements are met using bipolar PROMs or MOS/CMOS PROMs, EPROMs or E^{2}PROMs.

During operation, the address of the next instruction is sent out on A10, A9, A8 and IP7 through IP0 during the time that AD/$\overline{\text{DATA}}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/DATA line; A10, A9 and A8 are dedicated address outputs, and do not need to be latched. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or data input. A simplified block diagram of the external memory interface is shown in *Figure 11.*



TL/DD/5530-13

FIGURE 11. External Memory Interface to COP404C

COP404C Instruction Set

Table II is a symbol table providing internal architecture, instruction operand and operation symbols used in the instruction set table.

Table III provides the mnemonic, operand, machine code data flow, skip conditions and description of each instruction.

Table II. Instruction Set Table Symbols

Symbol	**Definition**
Internal Architecture Symbols	
A	4-bit Accumulator
B	7-bit RAM address register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry register
D	4-bit Data output port
EN	4-bit Enable register
G	4-bit General purpose I/O port
IL	two 1-bit (IN0 and IN3) latches
IN	4-bit input port
L	8-bit TRI-STATE I/O port
M	4-bit contents of RAM addressed by B
PC	11-bit ROM address program counter
Q	8-bit latch for L port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift register and counter
SK	Logic-controlled clock output
SKL	1-bit latch for SK output
T	8-bit timer
Instruction operand symbols	
d	4-bit operand field, 0–15 binary (RAM digit select)
r	3-bit operand field, 0–7 binary (RAM register select)
a	11-bit operand field, 0–2047
y	4-bit operand field, 0–15 (immediate data)
RAM(x)	RAM addressed by variable x
ROM(x)	ROM addressed by variable x
Operational Symbols	
+	Plus
–	Minus
->	Replaces
<->	is exchanged with
=	Is equal to
$\overline{A}$	one's complement of A
⊕	exclusive-or
:	range of values

Instruction Set (Continued)

TABLE III. COP404C Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	|0011|0000|	A+C+RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	|0011|0001|	A+RAM(B) → A	None	Add RAM to A
ADT		4A	|0011|0001|	$A+10_{10}$ → A	None	Add Ten to A
AISC	y	5−	|0101| y |	A+y → A	Carry	Add Immediate. Skip on Carry (y≠ 0)
CASC		10	|0001|0000|	$\overline{A}$+RAM(B)+C → A Carry → C	Carry	Compliment and Add with Carry, Skip on Carry
CLRA		00	|0000|0000|	0 → A	None	Clear A
COMP		40	|0100|0000|	$\overline{A}$ → A	None	Ones complement of A to A
NOP		44	|0100|0100|	None	None	No Operation
RC		32	|0011|0010|	"0" → C	None	Reset C
SC		22	|0010|0010|	"1" → C	None	Set C
XOR		02	|0000|0010|	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	|1111|1111|	ROM ($PC_{10:8}$ A,M) → $PC_{7:0}$	None	Jump Indirect (note 2)
JMP	a	6− —	|0110|0|$a_{10:8}$| | $a_{7:0}$ |	a → PC	None	Jump
JP	a	—	|1| $a_{6:0}$ | (pages 2,3 only) or	a → $PC_{6:0}$	None	Jump within Page (Note 3)
		—	|11| $a_{5:0}$ | (all other pages)	a → $PC_{5:0}$		
JSRP	a	—	|10| $a_{5:0}$ |	PC+1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 4)
JSR	a	6− —	|0110|1|$a_{10:8}$| | $a_{7:0}$ |	PC+1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	|0100|1000|	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	|0100|1001|	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
HALT		33 38	|0011|0011| |0011|1000|		None	HALT processor
IT		33 39	|0011|0011| |0011|1001|		None	IDLE till timer overflows then continues
MEMORY REFERENCE INSTRUCTIONS						
CAMT		33 3F	|0011|0011| |0011|1111|	A → $T_{7:4}$ RAM(B) → $T_{3:0}$	None	Copy A, RAM to T
CTMA		33 2F	|0011|0011| |0010|1111|	$T_{7:44}$ → RAM(B) $T_{3:0}$ → A	None	Copy T to RAM, A
CAMQ		33 3C	|0011|0011| |0011|1100|	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	|0011|0011| |0010|1100|	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	−5	|00| r |0101| (r=0:3)	RAM(B) → A Br⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 —	|0010|0011| |0| r | d |	RAM(r,d) → A	None	Load A with RAM pointed to direct by r,d
LQID		BF	|1011|1111|	ROM($PC_{10:8,A,M}$) → Q SB → SC	None	Load Q Indirect (Note 2)
RMB	0	4C	|0100|1100|	0 → $RAM(B)_0$	None	Reset RAM Bit
	1	45	|0100|0101|	0 → $RAM(B)_1$		
	2	42	|0100|0010|	0 → $RAM(B)_2$		
	3	43	|0100|0011|	0 → $RAM(B)_3$		

Instruction Set (Continued)

TABLE III. COP404C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
SMB	0	4D	|0100|1101|	1 → $RAM(B)_0$	None	Set RAM Bit
	1	47	|0100|0111|	1 → $RAM(B)_1$		
	2	46	|0100|0110|	1 → $RAM(B)_2$		
	3	4B	|0100|1011|	1 → $RAM(B)_3$		
STII	y	7–	|0111| y |	y → RAM(B)	None	Store Memory Immediate
				Bd + 1 → Bd		and Increment Bd
X	r	–6	|00| r |0110|	RAM(B) ←→ A	None	Exchange RAM with A,
			(r=0:3)	Br ⊕ r → Br		Exclusive-OR Br with r
XAD	r,d	23	|0010|0011|	RAM(r,d) ←→ A	None	Exchange A with RAM
		—	|1| r | d |			pointed to directly by r,d
XDS	r	–7	|00| r |0111|	RAM(B) ←→ A	Bd	Exchange RAM with A
			(r=0:3)	Bd–1 → Bd	decrements	and Decrement Bd.
				Br ⊕ r → Br	past 0	Exclusive-OR Br with r
XIS	r	–4	|00| r |0100|	RAM(B) ←→ A	Bd	Exchange RAM with A
			(r=0:3)	Bd+1 → Bd	increments	and Increment Bd,
				Br ⊕ r → Br	past 15	Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	|0101|0000|	A → Bd	None	Copy A to Bd
CBA		4E	|0100|1110|	Bd → A	None	Copy Bd to A
LBI	r,d	—	|00| r |(d-1)|	r,d → B	Skip until	Load B Immediate with r,d
			(r=0:3:		not a LBI	(Note 5)
			d=0,9:15)			
			or			
		33	|0011|0011|			
		—	|1| r | d |			
			(any r, any d)			
LEI	y	33	|0011|0011|	y → EN	None	Load EN Immediate (Note 6)
		6–	|0110| y |			
XABR		12	|0001|0010|	A ←→ Br	None	Exchange A with Br (Note 7)
TEST INSTRUCTIONS						
SKC		20	|0010|0000|		C="1"	Skip if C is True
SKE		21	|0010|0001|		A=RAM(B)	Skip if A Equals RAM
SKGZ		33	|0011|0011|		$G_{3:0}=0$	Skip if G is Zero
		21	|0010|0001|			(all 4 bits)
SKGBZ		33	|0011|0011|	1st byte		Skip if G Bit is Zero
	0	01	|0000|0001|		$G_0=0$	
	1	11	|0001|0001|	2nd byte	$G_1=0$	
	2	03	|0000|0011|		$G_2=0$	
	3	13	|0001|0011|		$G_3=0$	
SKMBZ	0	01	|0000|0001|		$RAM(B)_0=0$	Skip if RAM Bit is Zero
	1	11	|0001|0001|		$RAM(B)_1=0$	
	2	03	|0000|0011|		$RAM(B)_2=0$	
	3	13	|0001|0011|		$RAM(B)_3=0$	
SKT		41	|0100|0001|		A time-base counter carry has occured since last test	Skip on Timer (Note 2)

Instruction Set (Continued)

TABLE III. COP404C Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	|0011|0011| |0010|1010|	G → A	None	Input G Ports to A
ININ		33 28	|0011|0011| |0010|1000|	IN → A	None	Input IN Inputs to A
INIL		33 29	|0011|0011| |0010|1001|	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 2)
INL		33 2E	|0011|0011| |0010|1110|	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM,A
OBD		33 3E	|0011|0011| |0011|1110|	Bd → D	None	Output Bd to D Outputs
OGI	y	33 5–	|0011|0011| |0101| y |	y → G	None	Output to G Ports Immediate
OMG		33 3A	|0011|0011| |0011|1010|	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	|0100|1111|	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered O to N where O signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data minus 1, e.g., to load the lower four bits of B(Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds with the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Note 7: If $\overline{\text{SEL20}}$ = 1, A ←→ Br (0 → A3)
If $\overline{\text{SEL20}}$ = 0, A ←→ Br (0,0 → A3, A2).

Description of Selected Instructions

XAS INSTRUCTION

XAS (Exchange A with SIO) copies C to the SKL latch and exchanges the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. If SIO is selected as a shift register, an XAS instruction can be performed once every 4 instruction cycles to effect a continuous data stream.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC10: PC8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of the PC as follows: A → PC (7:4), RAM(B) → PC(3:0), leaving PC(10), PC(9) and PC(8) unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost.

Note: LQID uses 2 instruction cycles if executed, one if skipped.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the contents of ROM addressed by the 11-bit word, PC10: 8, A, M. PC10, PC9 and PC8 are not affected by JID.

Note: JID uses 2 instruction cycles if executed, one if skipped.

Description of Selected Instructions (Continued)

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of the T counter overflow latch (see internal logic, above), executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction allow the processor to generate its own time-base for real-time processing, rather than relying on an external input signal

Note: If the most significant bit of the T counter is a 1 when a CAMT instruction loads the counter, the overflow flag will be set. The following sample of codes should be used when loading the counter:

```
CAMT      ; load T counter
SKT       ; skip if overflow flag is set and reset it
NOP
```

IT INSTRUCTION

The IT (idle till timer) instruction halts the processor and puts it in an idle state until the time-base counter overflows. This idle state reduces current drain since all logic (except the oscillator and time base counter) is stopped. IT instruction is not allowed if the T counter is used as an external event counter ($\overline{\text{TIN}}$ pin tied to V_{CC}).

INIL INSTRUCTION

INIL (Input IL Latches to A) inputs 2 latches, IL3 and IL0, CKOI and 0 into A. The IL3 and IL0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN3 and IN0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction cycles. Execution of an INIL inputs IL3 and IL0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN3 and IN0 lines. The state of CKOI is input into A2. A 0 is input into A1. IL latches are cleared on reset.

Instruction Set Notes

a. The first word of a program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, they are still fetched from the program memory. Thus program paths take the same number of cycles whether instructions are skipped or executed except for JID, and LQID.

c. The ROM is organized into pages of 64 words each. The Program Counter is a 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID, or LQID is the last word of a page, it operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a JID or LQID located in the last word of every fourth page (i.e. hex address 0FF, 1FF, 2FF, 3FF, 4FF, etc.) will access data in the next group of four pages.

Power Dissipation

The lowest power drain is when the clock is stopped. As the frequency increases so does current. Current is also lower at lower operating voltages. Therefore, for minimum power dissipation, the user should run at the lowest speed and voltage that his application will allow. The user should take care that all pins swing to full supply levels to insure that outputs are not loaded down and that inputs are not at some intermediate level which may draw current. Any input with a slow rise or fall time will draw additional current. For example, an RC oscillator on D0 will draw more current than a square wave clock input since it is a slow rising signal.

If using an external square wave oscillator, the following equation can be used to calculate the COP404C operating current drain:

$$I_{co} = Iq + V \times 40 \times F_i + V \times 1400 \times F_i / 4$$

where:

I_{co} = chip operating current drain in microamps

I_q = quiescent leakage current (from curve)

F_i = CKI frequency in MegaHertz

V = chip V_{CC} in volts

For example at 5 volts V_{CC} and 400 kHz:

$$I_{co} = 20 + 5 \times 40 \times .4 + 5 \times 1400 \times .4 / 4$$

$$I_{co} = 20 + 80 + 700 = 800\ \mu A$$

at 2.4 volts V_{CC} and 30 kHz:

$$I_{co} = 6 + 2.4 \times 40 \times .03 + 2.4 \times 1400 \times .0^{3}/_{4}$$

$$I_{co} = 6 + 2.88 + 25.2 = 34.08\ \mu A$$

If an IT instruction is executed, the chip goes into the IDLE mode until the timer overflows. In IDLE mode, the current drain can be calculated from the following equation:

$$I_{ci} = I_q + V \times 40 \times F_i$$

For example, at 5 volts V_{CC} and 400 kHz

$$I_{ci} = 20 + 5 \times 40 \times .4 = 100\ \mu A$$

The total average current will then be the weighted average of the operating current and the idle current:

$$Ita = Ico \times \frac{To}{To + Ti} + Ici \times \frac{Ti}{To + Ti}$$

where:

I_{ta} = total average current

I_{co} = operating current

I_{ci} = idle current

T_o = operating time

T_i = idle time

I/O OPTIONS

COP404C outputs have the following configurations, illustrated in *Figure 12.*

a. Standard — A CMOS push-pull buffer with an N-channel device to ground in conjunction with a P-channel device to V_{CC}, compatible with CMOS and LSTTL. (Used on SO, SK, AD/$\overline{\text{DATA}}$, SKIP, A10:8 and D outputs.)

b. Low Current — This is the same configuration as a. above except that the sourcing current is much less. (Used on G outputs.)

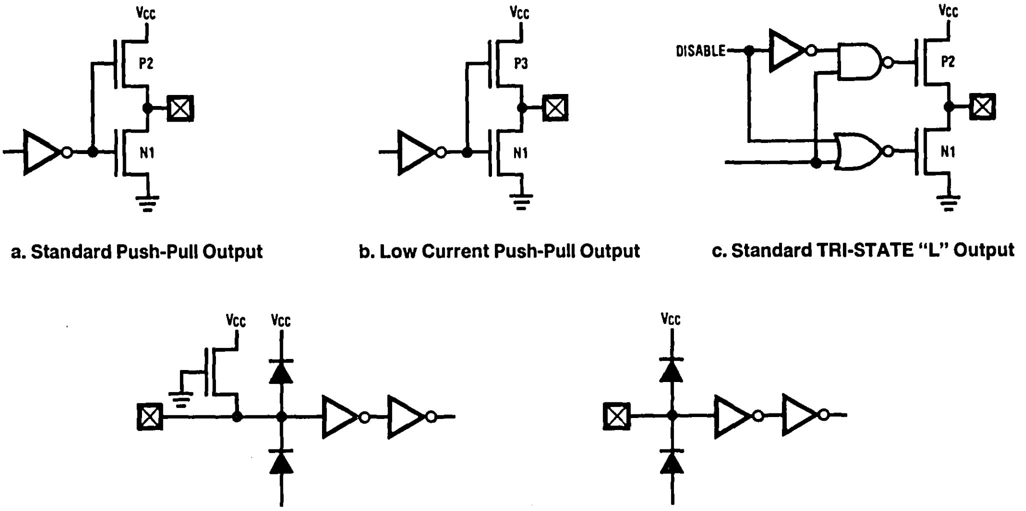
c. Standard TRI-STATE L Output — A CMOS output buffer similar to a. which may be disabled by program control. (Used on L outputs.)

All inputs have the following configuration:

d. Input with on chip load device to V_{CC}. (Used on CKOI.)

e. HI-Z input which must be driven by the users logic. (Used on CKI, $\overline{\text{RESET}}$, IN, SI, $\overline{\text{DUAL}}$, $\overline{\text{TIN}}$, $\overline{\text{MB}}$, $\overline{\text{SEL10}}$ and $\overline{\text{SEL20}}$ inputs.)

All output drivers use one or more of three common devices numbered 1 to 3. Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 13* for each of these devices to allow the designer to effectively use these I/O configurations.



a. Standard Push-Pull Output

b. Low Current Push-Pull Output

c. Standard TRI-STATE "L" Output

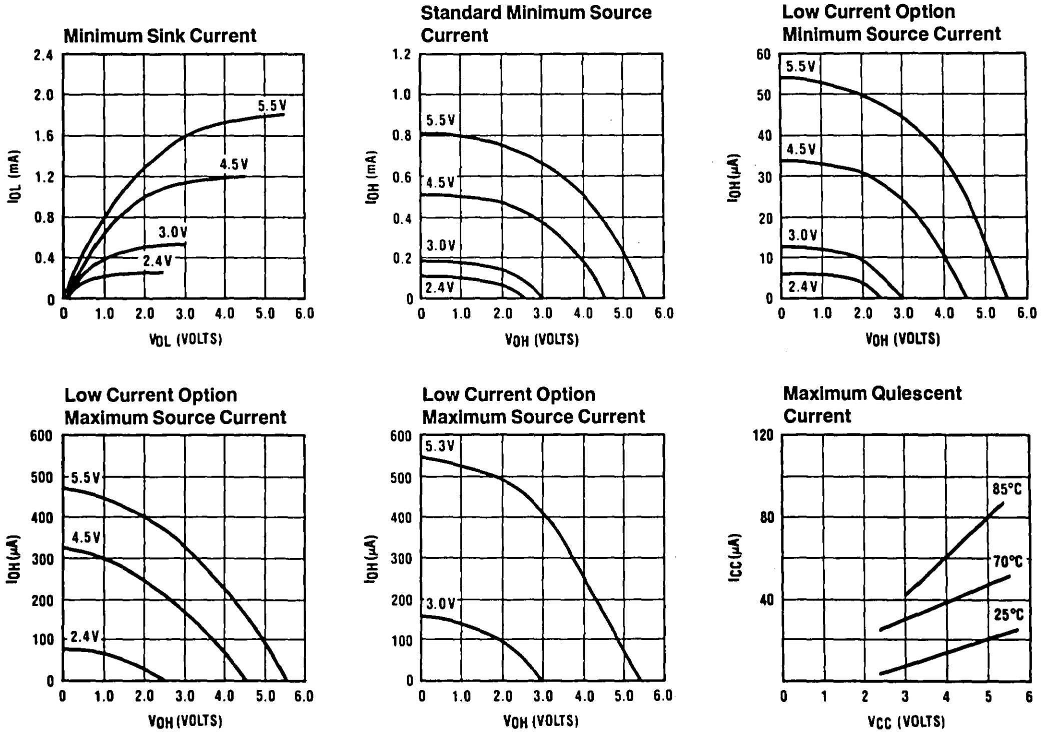
d. Input with Load

e. Hi-Z Input

TL/DD/5530-15

FIGURE 12. Input/Output Configurations

Typical Performance Characteristics



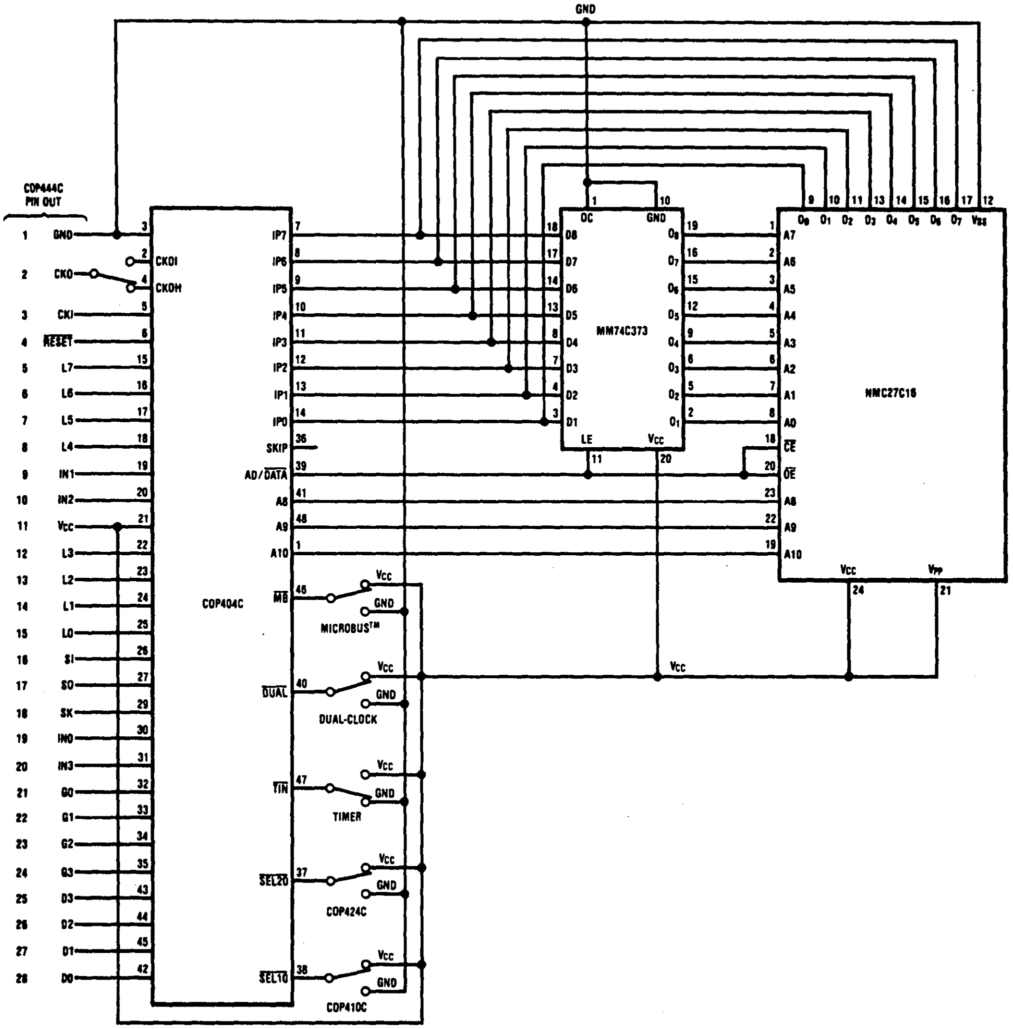
TL/DD/5530-16

FIGURE 13. Input/Output Characteristics

Emulation

The COP404C may be used to exactly emulate the COP444C/445C, COP424C/425C, and COP410C/411C. However, the Program Counter always addresses 2k of external ROM whatever chip is being emulated. *Figure 14* shows the interconnect to implement a hardware emulation. This connection uses a NMC27C16 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74C373) latches the addresses to drive the memory.



TL/DD/5530-14

FIGURE 14. COP404C Used To Emulate A COP444C

Emulation (Continued)

When AD/$\overline{\text{DATA}}$ turns off, the EPROM is enabled and the IP7–IP0 pins will input the memory data. A10, A9 and A8 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

— CKI is divided by 4. Other divide-by are emulated by external divider.

— CKO can be emulated as a general purpose input by using CKOI or as a Halt I/O port by using CKOH.

— $\overline{\text{MB}}$ pin can be pulled low if the MICROBUS feature of the COP444C and COP424C is needed. Othewise it should be high.

— $\overline{\text{DUAL}}$ pin can be pulled low if the Dual-Clock feature of the COP444C and COP424C is needed. Otherwise it should be high.

— $\overline{\text{TIN}}$ pin controls the input of the 8-bit timer of the COP444C and COP424C (internal timer if $\overline{\text{TIN}}$ is low, external event counter if $\overline{\text{TIN}}$ is high).

— The $\overline{\text{SEL10}}$ and $\overline{\text{SEL20}}$ inputs are used to emulate the COP444C/445C, COP424C/425C, or COP410C/411C.

- When emulating the COP444C/445C, the user must configure $\overline{\text{SEL20}}$ = 1 and $\overline{\text{SEL10}}$ = 1.
- When emulating the COP424C/425C, the user must configure $\overline{\text{SEL20}}$ = 0 and $\overline{\text{SEL10}}$ = 1. In this mode, the user RAM is physically halved. As in the COP424C/425C, the user has 64 digits (256 bits) of RAM available. Pin A10 should not be connected to the program memory (most significant address bit of the program memory should be grounded if using a 2k×8 memory).
- When emulating the COP410C/411C, the user must configure $\overline{\text{SEL20}}$ = 0 and $\overline{\text{SEL10}}$ = 0. In this mode, the user has 32 digits (128 bits) of RAM available organized in the same way as the COP410C/411C - 4 registers of 8 digits each. Pins A10 and A9 should not be connected to the program memory (the 2 most significant address bits of the program memory should be grounded).

  Furthermore, the subroutine stack is decreased from 3 levels to 2 levels.

The pins $\overline{\text{SEL10}}$ and $\overline{\text{SEL20}}$ change the internal logic of the device to accurately emulate the devices as indicated above. However, the user must remember that the COP424C/425C is a subset of the COP444C/COP445C with respect to memory size. The COP410C/411C is a subset both in memory size and in function. The user must take care not to use features and instructions which are not available on the COP410C/411C (see table IV. below) when using the COP404C to emulate the COP410C/411C.

TABLE IV. FEATURES AND INSTRUCTIONS NOT AVAILABLE ON COP410C/411C.

Timer	ADT		
Dual-clock	CASC		
Interrupt	CAMT		
Microbus	CTMA		
	IT		
	LDD	r, d	
	XAD	r, d	(except 3, 15)
	XABR		
	SKT		
	ININ		
	INIL		
	OGI	y	

Option Table

COP404C MASK OPTIONS

The following COP444C options have been implemented in the COP404C:

Option value	Comment
Option 1 = 0	Ground Pin — no option available
Option 2 = 1, 2	CKO is replaced by CKOI and CKOH
Option 3 = 5	CKI is external clock input divided by 4
Option 4 = 1	$\overline{\text{RESET}}$ is Hi-Z input
Option 5–8 = 0	L outputs are standard TRI-STATE
Option 9 = 1	IN1 is a Hi-Z input
Option 10 = 1	IN2 is a Hi-Z input
Option 11 = 0	V_{CC} pin — no option available
Option 12–15 = 0	L outputs are standard TRI-STATE
Option 16 = 0	SI is a Hi-Z input
Option 17 = 0	SO is a standard output
Option 18 = 0	SK is a standard output
Option 19 = 1	IN0 is a Hi-Z input
Option 20 = 1	IN3 is a Hi-Z input
Option 21–24 = 1	G outputs are low-current
Option 25–28 = 0	D outputs are standard
Option 29 = 1	No internal initialization logic
Option 30 = 0, 1	DUAL-CLOCK is pin selectable
Option 31 = 0, 1	TIMER is pin selectable
Option 32 = 0, 1	MICROBUS is pin selectable
Option 33 = N/A	48-pin package

National Semiconductor

COP404LSN-5 ROMless N-Channel Microcontrollers

General Description

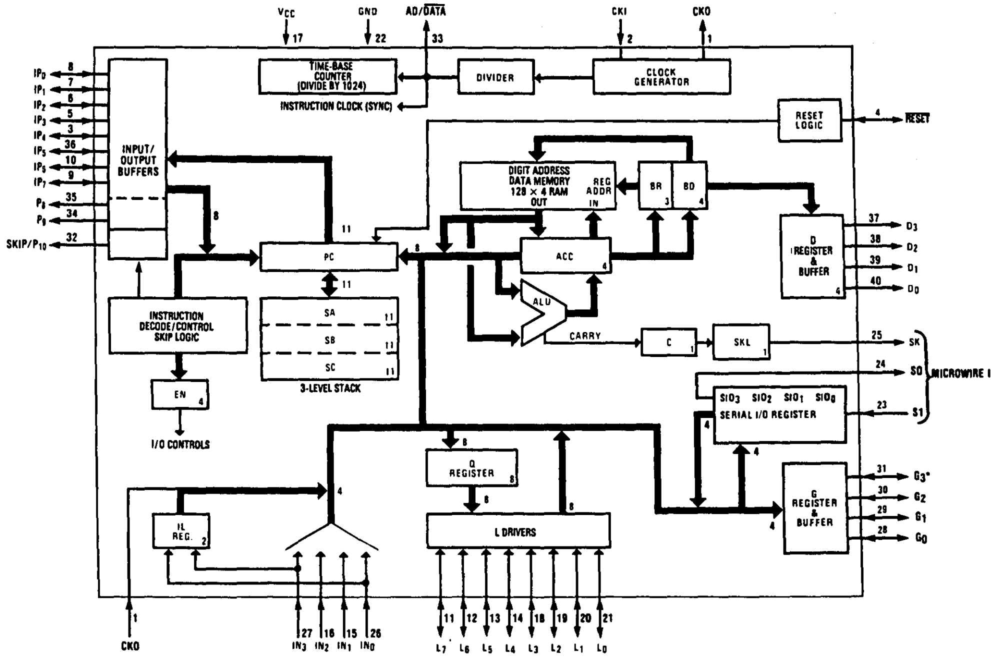
The COP404LSN-5 ROMless Microcontroller is a member of the COPS™ family, fabricated using N-channel, silicon gate MOS technology. The COP404LSN-5 contains CPU, RAM, I/O and is identical to a COP444L device except the ROM has been removed and pins have been added to output the ROM address and to input the ROM data. In a system the COP404LSNN-5 will perform exactly as the COP444L. This important benefit facilitates development and debug of a COP program prior to masking the final part. The COP404LSN-5 is also appropriate in low volume applications, or when the program might be changing. The COP404LSN-5 may be used to emulate the COP444L, COP445L, COP420L, and the COP421L.

Use COP404LSN-5 in volume applications. For extended temperature range (−40°C to +85°C), COP304L is available on a special order basis.

Features

- Exact circuit equivalent of COP444L
- Low cost
- Powerful instruction set
- 128 x 4 RAM, addresses 2048 x 8 ROM
- True vectored interrupt, plus restart
- Three-level subroutine stack
- 16 μs instruction time
- Single supply operation (4.5V–5.5V)
- Low current drain (16 mA max)
- Internal time-base counter for real-time processing
- Internal binary counter register with MICROWIRE™ compatible serial I/O
- General purpose outputs
- LSTTL/CMOS compatible in and out
- Direct drive of LED digit and segment lines
- Software/hardware compatible with other members of COP400 family

Block Diagram



TL/DD/8817-1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C
Total Source Current	120 mA
Total Sink Current	140 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

$4.5V \le V_{CC} \le 5.5V$; $0°C \le T_A \le 70°C$

Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 2)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current	All Inputs and Outputs Open		16	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})		2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
IP0–IP7, SI Input Levels				
Logic High	V_{CC} = 5.5V	2.4		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
All Other Inputs				
Logic High	High Trip Level Options	3.6		V
Logic Low	Selected	−0.3	1.2	V
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±10%			
Logic High (V_{OH})	I_{OH} = −25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
IP0–IP7, P8, P9, SKIP/P10	(Note 1)			
Logic High	I_{OH} = −80 μA	2.4		V
Logic Low	I_{OL} = 720 μA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L_0–L_7 Outputs	V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
G_0–G_3 and D_0–D_3 Outputs	V_{CC} = 4.5V, V_{OL} = 1.0V	7.5		mA
CKO	V_{CC} = 4.5V, V_{OL} = 0.4V	0.2		mA
Output Source Current				
D_0–D_3, G_0–G_3 Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 2.0V	−30	−250	μA
SO and SK Outputs (I_{OH})	V_{CC} = 4.5V, V_{OH} = 1.0V	−1.2		mA
L_0–L_7 Outputs	V_{CC} = 5.5V, V_{OH} = 2.0V	−1.4	−25	mA

DC Electrical Characteristics (Continued)

$0°C \leq T_A \leq +70°C$, $4.5V \leq V_{CC} \leq 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Input Load Source Current (I_{IL})	$V_{CC} = 5.0V$, $V_{IL} = 0V$	−10	−140	μA
Total Sink Current Allowed				
All Outputs Combined			140	mA
D, G Ports			120	mA
L_7–L_4			4	mA
L_3–L_0			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L_7–L_4			60	mA
L_3–L_0			60	mA
Each L Pin			30	mA
All Other Pins			1.5	mA

AC Electrical Characteristics

$0°C \leq T_A \leq 70°C$, $4.5V \leq V_{CC} \leq 5.5V$ unless otherwise specified

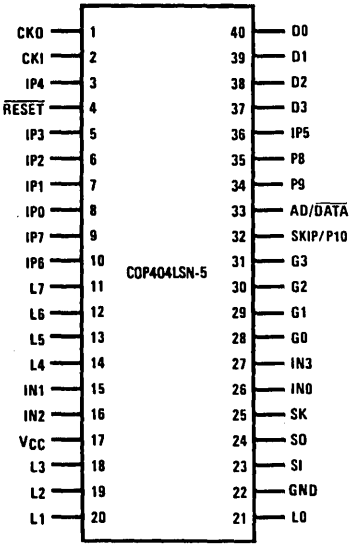
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		16	40	μs
CKI				
Input Frequency, f	(÷32 Mode)	0.8	2	MHz
Duty Cycle		30	60	%
Rise Time	$f_I = 2.0$ MHz		120	ns
Fall Time			80	ns
INPUTS:				
SI, IP7–IP0				
t_{SETUP}		2.0		μs
t_{HOLD}		1.0		μs
IN_3–IN_0, G_3–G_0, L_7–L_0				
t_{SETUP}		8.0		μs
t_{HOLD}		1.3		μs
OUTPUT PROPAGATION DELAY	Test Condition:			
	$C_L = 50$ pF, $V_{OUT} = 1.5V$			
SO, SK Outputs	$R_L = 20$ kΩ			
t_{pd1}, t_{pd0}			4.0	μs
D_3–D_0, G_3–G_0, L_7–L_0	$R_L = 20$ kΩ			
t_{pd1}, t_{pd0}			5.6	μs
IP7–IP0, P8, P9, SKIP	$R_L = 5$ kΩ			
t_{pd1}, t_{pd0}			7.2	μs
P10	$R_L = 5$ kΩ			
t_{pd1}, t_{pd0}			6.0	μs

Note 1: COP404LSN-5 has Push-Pull drivers on these outputs.

Note 2: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.

Connection Diagram

Dual-In-Line Package



TL/DD/8817-2

Top View

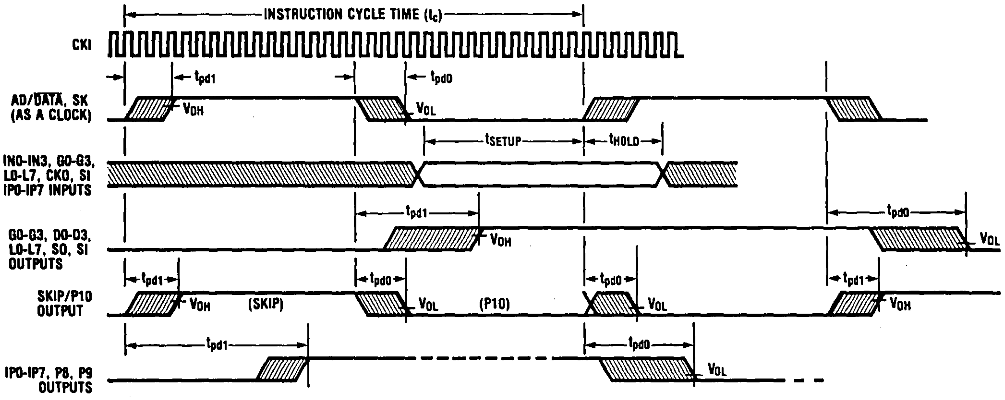
FIGURE 2

Order Number COP404LSN-5
See NS Package Number N40A

Pin Descriptions

Pin	Description
L_7–L_0	8 bidirecitonal I/O ports with TRI-STATE®
G_3–G_0	4 bidirectional I/O ports
D_3–D_0	4 general purpose outputs
IN_3–IN_0	4 general purpose outputs
SI	Serial input (or counter input
SO	Serial output (or general purpose output)
SK	Logic-controlled clock (or general purpose output)
AD/$\overline{\text{DATA}}$	Address out/data in flag
CKI	System oscillator input
CKO	System oscillator output (COP404LSN-5)
$\overline{\text{RESET}}$	System reset input
V_{CC}	Power supply
GND	Ground
IP7–IP0	8 bidirectional ROM address and data ports
P8, P9	2 ROM address outputs
SKIP/P10	Instruction skip output and most significant ROM address bit output

Timing Diagram



TL/DD/8817-3

FIGURE 3. Input/Output

Functional Description

A block diagram of the COP404LSN-5 is given in *Figure 1.* Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device. Positive logic is used. When a bit is set, it is a logic "1" (greater than 2V). When a bit is reset, it is a logic "0" (less than 0.8V).

PROGRAM MEMORY

Program Memory consists of a 2048 byte external memory. As can be seen by an examination of the COP404LSN-5 instruction set, these words may be program instructions, program data or ROM addressing data. Because of the special characteristics associated with the JP, JSRP, JID and LQID instructions, ROM must often be thought of as being organized into 32 pages of 64 words each.

ROM addressing is accomplished by an 11-bit PC register. Its binary value selects one of the 2048 8-bit words contained in ROM. A new address is loaded into the PC register during each instruction cycle. Unless the instruction is a transfer of control instruction, the PC register is loaded with the next sequential 11-bit binary count value. Three levels of subroutine nesting are implemented by the 11-bit subroutine saves registers, SA, SB, and SC, providing a last-in, first-out (LIFO) hardware subroutine stack.

ROM instruction words are fetched, decoded and executed by the Instruction Decode, Control and Skip Logic circuitry.

DATA MEMORY

Data memory consists of a 512-bit RAM, organized as 8 data registers of 16 4-bit digits. RAM addressing is implemented by a 7-bit B register whose upper 3 bits (Br) select 1 of 8 data registers and lower 4 bits (Bd) select 1 of 16 4-bit digits in the selected data register. While the 4-bit contents of the selected RAM digit (M) is usually loaded into or from, or exchanged with, the A register (accumulator), it may also be loaded into or from the Q latches or loaded from the L ports. RAM addressing may also be performed directly by the LDD and XAD instructions based upon the 7-bit contents of the operand field of these instructions. The Bd register also serves as a source register for 4-bit data sent directly to the D outputs.

INTERNAL LOGIC

The 4-bit A register (accumulator) is the source and destination register for most I/O, arithmetic, logic and data memory access operations. It can also be used to load the Br and Bd portions of the B register, to load and input 4 bits of the 8-bit Q latch data, to input 4 bits of the 8-bit L I/O port data and to perform data exchanges with the SIO register.

A 4-bit adder performs the arithmetic and logic functions, storing its results in A. It also outputs a carry bit to the 1-bit C register, most often employed to indicate arithmetic overflow. The C register, in conjunction with the XAS instruction and the EN register, also serves to control the SK output. C can be outputted directly to SK or can enable SK to be a sync clock each instruction cycle time. (See XAS instruction and EN register description, below).

Four general-purpose inputs, IN_3-IN_0, are provided.

The D register provides 4 general-purpose outputs and is used as the destination register for the 4-bit contents of Bd. The D outputs can be directly connected to the digits of a multiplexed LED display.

The G register contents are outputs to 4 general-purpose bidirectional I/O ports. G I/O ports can be directly connected to the digits of a multiplexed LED display.

The Q register is an internal, latched, 8-bit register, used to hold data loaded to or from M and A, as well as 8-bit data from ROM. Its contents are output to the L I/O ports when the L drivers are enabled under program control. (See LEI instruction.)

The 8 L drivers, when enabled, output the contents of latched Q data to the L I/O ports. Also, the contents of L may be read directly into A and M. L I/O ports can be directly connected to the segments of a multiplexed LED display (using the LED Direct Drive output configuration option) with Q data being outputted to the Sa-Sg and decimal point segments of the display.

The SIO register functions as a 4-bit serial-in/serial-out shift register or as a binary counter depending on the contents of the EN register. (See EN register description, below.) Its contents can be exchanged with A, allowing it to input or output a continuous serial data stream. SIO may also be used to provide additional parallel I/O by connecting SO to external serial-in/parallel-out shift registers.

The XAS instruction copies C into the SKL latch. In the counter mode, SK is the output of SKL; in the shift register mode, SK outputs SKL ANDed with the clock.

The EN register is an internal 4-bit register loaded under program control by the LEI instruction. The state of each bit of this register selects or deselects the particular feature associated with each bit of the EN register (EN_3-EN_0).

1. The least significant bit of the enable register, EN_0, selects the SIO register as either a 4-bit shift register or a 4-bit binary counter. With EN_0 set, SIO is an asynchronous binary counter, *decrementing* its value by one upon each low-going pulse ("1" to "0") occurring on the SI input. Each pulse must be at least two instruction cycles wide. SK outputs the value of SKL. The SO output is equal to the value of EN_3. With EN_0 reset, SIO is a serial shift register shifting left each instruction cycle time. The data present at SI goes into the least significant bit of SIO. SO can be enabled to output the most significant bit of SIO each cycle time. (See 4 below.) The SK output becomes a logic-controlled clock.
2. With EN_1 set the IN_1 input is enabled as an interrupt input. Immediately following an interrupt, EN_1 is reset to disable further interrupts.
3. With EN_2 set, the L drivers are enabled to output the data in Q to the L I/O ports. Resetting EN_2 disables the L drivers, placing the L I/O ports in a high-impedance input state.

Functional Description (Continued)

4. EN_3, in conjunction with EN_0, affects the SO output. With EN_0 set (binary counter option selected) SO will output the value loaded into EN_3. With EN_0 reset (serial shift register option selected), setting EN_3 enables SO as the output of the SIO shift register, outputting serial shifted data each instruction time. Resetting EN_3 with the serial shift register option selected disables SO as the shift register output; data continues to be shifted through SIO and can be exchanged with A via an XAS instruction but SO remains reset to "0." The table below provides a summary of the modes associated with EN_3 and EN_0.

INTERRUPT

The following features are associated with the IN_1 interrupt procedure and protocol and must be considered by the programmer when utilizing interrupts.

a. The interrupt, once acknowledged as explained below, pushes the next sequential program counter address (PC+1) onto the stack, pushing in turn the contents of the other subroutine-save registers to the next lower level (PC+1 → SA → SB → SC). Any previous contents of SC are lost. The program counter is set to hex address 0FF (the last word of page 3) and EN_1 is reset.

b. An interrupt will be acknowledged only after the following conditions are met:

1. EN_1 has been set.
2. A low-going pulse ("1" to "0") at least two instruction cycles wide occurs on the IN_1 input.
3. A currently executing instruction has been completed.
4. All successive transfer of control instructions and successive LBIs have been completed (e.g., if the main program is executing a JP instruction which transfers program control to another JP instruction, the interrupt will not be acknowledged until the second JP instruction has been executed.

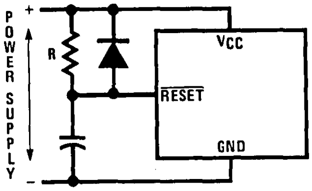
c. Upon acknowledgement of an interrupt, the skip logic status is saved and later restored upon popping of the stack. For example, if an interrupt occurs during the execution of ASC (Add with Carry, Skip on Carry) instruction which results in carry, the skip logic status is saved and program control is transferred to the interrupt servicing routine at hex address 0FF. At the *end* of the interrupt routine, a RET instruction is executed to "pop" the stack and return program control to the instruction following the original ASC. *At this time*, the skip logic is enabled and skips this instruction because of the previous ASC carry. Subroutines and LQID instructions should not be nested within the interrupt service routine, since their popping the stack will enable any previously saved main program skips, interfering with the orderly execution of the interrupt routine.

d. The first instruction of the interrupt routine at hex address 0FF must be a NOP.

e. A LEI instruction can be put immediately before the RET to re-enable interrupts.

INITIALIZATION

The Reset Logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. If the power supply rise time is greater than 1 ms, the user must provide an external RC network and diode to the $\overline{RESET}$ pin as shown below. The $\overline{RESET}$ pin is configured as a Schmitt trigger input. If the RC network is not used, the $\overline{RESET}$ pin should be left open. Initialization will occur whenever a logic "0" is applied to the $\overline{RESET}$ input, provided it stays low for at least three instruction cycle times.



TL/DD/8817-4

RC ≥ 5 × Power Supply Rise Time (R > 40k)

Upon initialization, the PC register is cleared to 0 (ROM address 0) and the A, B, C, D, EN, and G registers are cleared. The SK output is enabled as a SYNC output, providing a pulse each instruction cycle time. *Data Memory (RAM) is not cleared upon initialization.* The first instruction at address 0 must be a CLRA.

EXTERNAL MEMORY INTERFACE

The COP404LSN-5 is designed for use with an external Program Memory. This memory may be implemented using any devices having the following characteristics:

1. random addressing
2. TTL-compatible TRI-STATE outputs
3. TTL-compatible inputs
4. access time = 5 µs max.

Typically these requirements are met using bipolar or MOS PROMs.

During operation, the address of the next instruction is sent out on P10, P9, P8, and IP7 through IP0 during the time that AD/$\overline{DATA}$ is high (logic "1" = address mode). Address data on the IP lines is stored into an external latch on the high-to-low transition of the AD/$\overline{DATA}$ line; P9 and P8 are

Enable Register Modes — Bits EN_3 and EN_0

EN_3	EN_0	SIO	SI	SO	SK
0	0	Shift Register	Input to Shift Register	0	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
1	0	Shift Register	Input to Shift Register	Serial Out	If SKL = 1, SK = CLOCK If SKL = 0, SK = 0
0	1	Binary Counter	Input to Binary Counter	0	If SKL = 1, SK = 1 If SKL = 0, SK = 0
1	1	Binary Counter	Input to Binary Counter	1	If SKL = 1, SK = 1 If SKL = 0, SK = 0

Functional Description (Continued)

dedicated address outputs, and do not need to be latched. SKIP/P10 outputs address data when AD/$\overline{\text{DATA}}$ is low. When AD/$\overline{\text{DATA}}$ is low (logic "0" = data mode), the output of the memory is gated onto IP7 through IP0, forming the input bus. Note that the AD/$\overline{\text{DATA}}$ output has a period of one instruction time, a duty cycle of approximately 50%, and specifies whether the IP lines are used for address output or instruction input.

OSCILLATOR

The basic clock oscillator configurations is shown in *Figure 4*.

Crystal Controlled Oscillator—CKI and CKO are connected to an external crystal. The instruction cycle time equals the crystal frequency divided by 32.

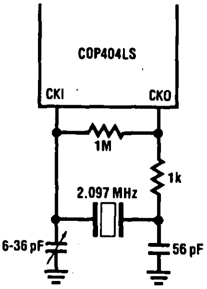


FIGURE 4. Oscillator

INPUT/OUTPUT CONFIGURATIONS

COP404LSN-5 outputs have the following configurations, illustrated in *Figure 5*:

a. **Standard**—an enhancement mode device to ground in conjunction with a depletion-mode device to V_{CC}, compatible with LSTTL and CMOS input requirements. (Used on D and G outputs.)

b. **Open-Drain**—an enhancement-mode device to ground only, allowing external pull-up as required by the user's application.

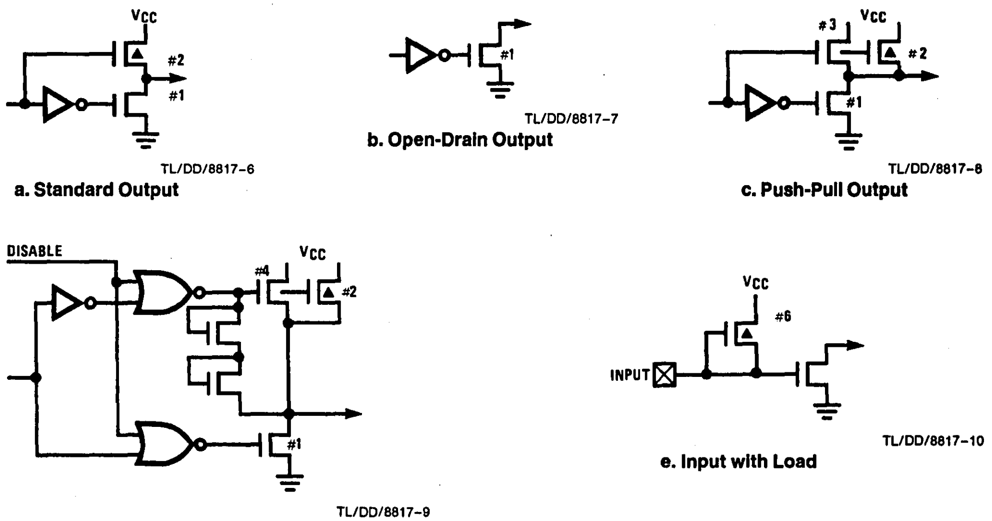
c. **Push-Pull**—an enhancement-mode device to ground in conjunction with a depletion-mode device paralleled by an enhancement-mode device to V_{CC}. This configuration has been provided to allow for fast rise and fall times when driving capacitive loads.

d. **LED Direct Drive**—an enhancement-mode device to ground and to V_{CC}, meeting the typical current sourcing requirements of the segments of an LED display. The sourcing device is clamped to limit current flow. These devices may be turned off under program control (see Functional Description, EN Register), placing the outputs in a high-impedance state to provide required LED segment blanking for a multiplexed display. (Used on L outputs.)

COP404LSN-5 inputs have an on-chip depletion load device to V_{CC}.

The above input and output configurations share common enhancement-mode and depletion-mode devices. Specifically, all configurations use one or more of six devices (numbered 1–6, respectively). Minimum and maximum current (I_{OUT} and V_{OUT}) curves are given in *Figure 6* for each of these devices to allow the designer to effectively use these I/O configurations in designing a system.

An important point to remember is that even when the L drivers are disabled, the depletion load device will source a small amount of current (see *Figure 6*, device 2); however, when the L-lines are used as inputs, the disabled depletion device can *not* be relied on to source sufficient current to pull an input to a logic "1".



a. Standard Output

b. Open-Drain Output

c. Push-Pull Output

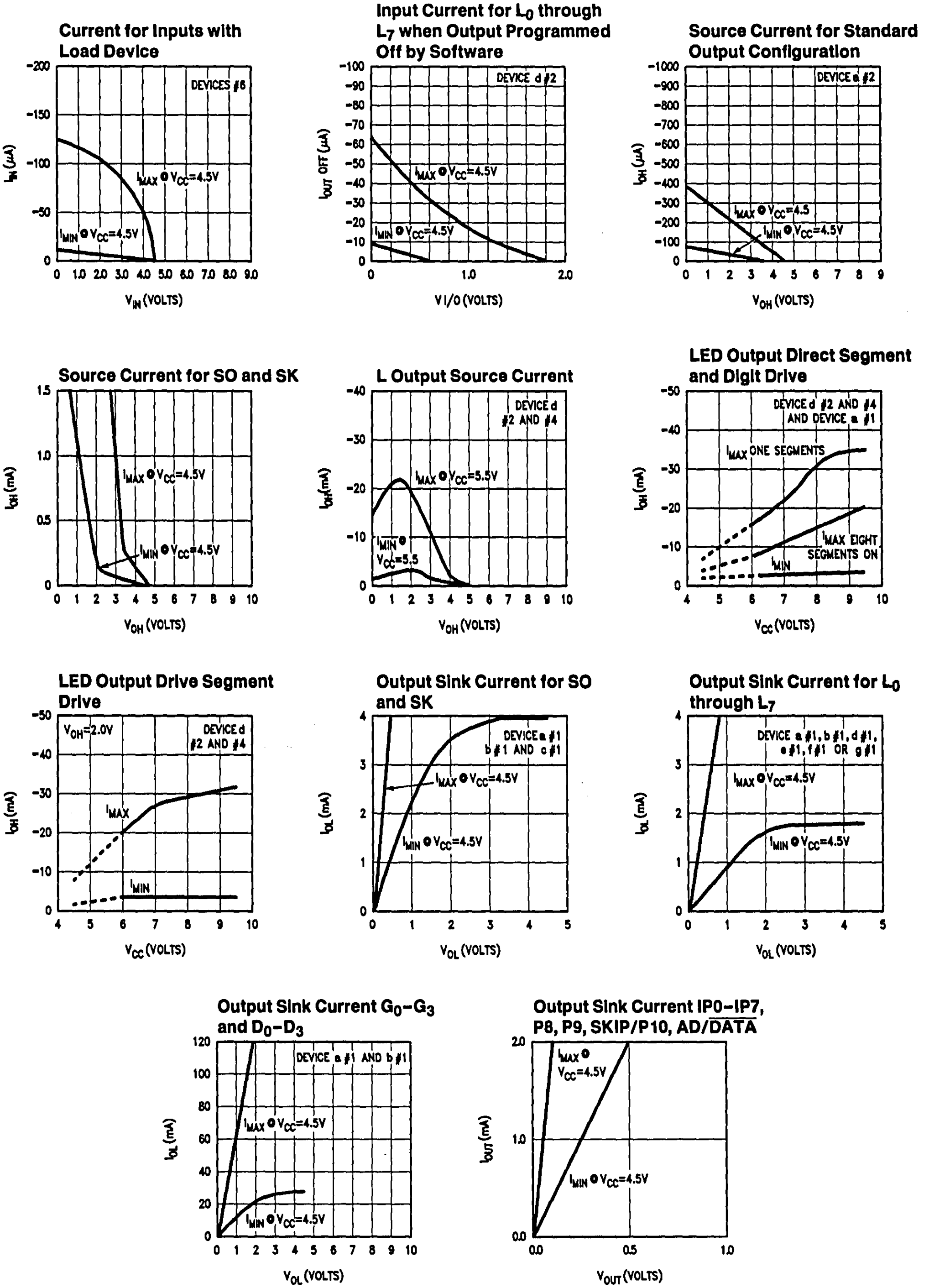
d. L Output (LED)

e. Input with Load

(▲ is Depletion Device)

FIGURE 5. Output Configurations

Typical Performance Characteristics



TL/DD/8817-11

FIGURE 6. COP404LSN-5 I/O Characteristics

COP404LSN-5 Instruction Set

Table I is a symbol table providing internal architecture, instruction operand and operational symbols used in the instruction set table.

Table II provides the mnemonic, operand, machine code, data flow, skip conditions, and description associated with each instruction in the COP404LSN-5 instruction set.

TABLE I. COP404LSN-5 Instruction Set Table Symbols

Symbol	Definition
INTERNAL ARCHITECTURE SYMBOLS	
A	4-bit Accumulator
B	10-bit RAM Address Register
Br	Upper 3 bits of B (register address)
Bd	Lower 4 bits of B (digit address)
C	1-bit Carry Register
D	4-bit Data Output Port
EN	4-bit Enable Register
G	4-bit Register to latch data for G I/O Port
IL	Two 1-bit latches associated with the IN_3 or IN_0 inputs
IN	4-bit Input Port
IP	8-bit bidirectional ROM address and Data Port
L	8-bit TRI-STATE I/O Port
M	4-bit contents of RAM Memory pointed to by B Register
P	3-bit ROM Address Register Port
PC	11-bit ROM Address Register (program counter)
Q	8-bit Register to latch data for L I/O Port
SA	11-bit Subroutine Save Register A
SB	11-bit Subroutine Save Register B
SC	11-bit Subroutine Save Register C
SIO	4-bit Shift Register and Counter
SK	Logic-Controlled Clock Output

Symbol	Definition
INSTRUCTION OPERAND SYMBOLS	
d	4-bit Operand Field, 0–15 binary (RAM Digit Select)
r	3-bit Operand Field, 0–7 binary (RAM Register Select)
a	11-bit Operand Field, 0–2047 binary (ROM Address)
y	4-bit Operand Field, 0–15 binary (Immediate Data)
RAM(s)	Contents of RAM location addressed by s
ROM(t)	Contents of ROM location addressed by t
OPERATIONAL SYMBOLS	
+	Plus
−	Minus
→	Replaces
↔	Is exchanged with
=	Is equal to
$\overline{A}$	The one's complement of A
⊕	Exclusive-OR
:	Range of values

TABLE II. COP404LSN-5 Instruction Set

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
ARITHMETIC INSTRUCTIONS						
ASC		30	0011 0000	A + C + RAM(B) → A Carry → C	Carry	Add with Carry, Skip on Carry
ADD		31	0011 0001	A + RAM(B) → A	None	Add RAM to A
ADT		4A	0100 1010	A + 10_{10} → A	None	Add Ten to A
AISC	y	5–	0101 y	A + y → A	Carry	Add Immediate, Skip on Carry (y ≠ 0)
CASC		10	0001 0000	$\overline{A}$ + RAM(B) + C → A Carry → C	Carry	Complement and Add with Carry, Skip on Carry
CLRA		00	0000 0000	0 → A	None	Clear A
COMP		40	0100 0000	$\overline{A}$ → A	None	One's complement of A to A
NOP		44	0100 0100	None	None	No Operation
RC		32	0011 0010	"0" → C	None	Reset C
SC		22	0010 0010	"1" → C	None	Set C
XOR		02	0000 0010	A ⊕ RAM(B) → A	None	Exclusive-OR RAM with A

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
TRANSFER OF CONTROL INSTRUCTIONS						
JID		FF	1111 1111	ROM ($PC_{10:8}$, A,M) → $PC_{7:0}$	None	Jump Indirect (Note 2)
JMP	a	6– – –	0110 0 $a_{10:8}$ $a_{7:0}$	a → PC	None	Jump
JP	a	– – – –	1 $a_{6:0}$ (pages 2,3 only) or 11 $a_{5:0}$ (all other pages)	a → $PC_{6:0}$ a → $PC_{5:0}$	None	Jump within Page (Note 4)
JSRP	a	– –	10 $a_{5:0}$	PC + 1 → SA → SB → SC 00010 → $PC_{10:6}$ a → $PC_{5:0}$	None	Jump to Subroutine Page (Note 5)
JSR	a	6– – –	0110 1 $a_{10:8}$ $a_{7:0}$	PC + 1 → SA → SB → SC a → PC	None	Jump to Subroutine
RET		48	0100 1000	SC → SB → SA → PC	None	Return from Subroutine
RETSK		49	0100 1001	SC → SB → SA → PC	Always Skip on Return	Return from Subroutine then Skip
MEMORY REFERENCE INSTRUCTIONS						
CAMQ		33 3C	0011 0011 0011 1100	A → $Q_{7:4}$ RAM(B) → $Q_{3:0}$	None	Copy A, RAM to Q
CQMA		33 2C	0011 0011 0010 1100	$Q_{7:4}$ → RAM(B) $Q_{3:0}$ → A	None	Copy Q to RAM, A
LD	r	–5	00 r 0101 (r = 0:3)	RAM(B) → A Br ⊕ r → Br	None	Load RAM into A, Exclusive-OR Br with r
LDD	r,d	23 – –	0010 0011 0 r d	RAM(r,d) → A	None	Load A with RAM pointed to directly by r,d
LQID		BF	1011 1111	ROM($PC_{10:8}$,A,M) → Q SB → SC	None	Load Q Indirect (Note 3)
RMB	0 1 2 3	4C 45 42 43	0100 1100 0100 0101 0100 0010 0100 0011	0 → $RAM(B)_0$ 0 → $RAM(B)_1$ 0 → $RAM(B)_2$ 0 → $RAM(B)_3$	None	Reset RAM Bit
SMB	0 1 2 3	4D 47 46 4B	0100 1101 0100 0111 0100 0110 0100 1011	1 → $RAM(B)_0$ 1 → $RAM(B)_1$ 1 → $RAM(B)_2$ 1 → $RAM(B)_3$	None	Set RAM Bit
STII	y	7–	0111 y	y → RAM(B) Bd + 1 → Bd	None	Store Memory Immediate and Increment Bd
X	r	–6	00 r 0110 (r = 0:3)	RAM(B) ←→ A Br ⊕ r → Br	None	Exchange RAM with A, Exclusive-OR Br with r
XAD	r,d	23 – –	0010 0011 1 r d	RAM(r,d) ←→ A	None	Exchange A with RAM pointed to directly by (r,d)
XDS	r	–7	00 r 0111 (r = 0:3)	RAM(B) ←→ A Bd − 1 → Bd Br ⊕ r → Br	Bd decrements past 0	Exchange RAM with A and Decrement Bd, Exclusive-OR Br with r

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
MEMORY REFERENCE INSTRUCTIONS (Continued)						
XIS	r	−4	00 r 0100 (r = 0:3)	RAM(B) ⟷ A Bd + 1 → Bd Br ⊕ r → Br	Bd increments past 15	Exchange RAM with A and Increment Bd, Exclusive-OR Br with r
REGISTER REFERENCE INSTRUCTIONS						
CAB		50	0101 0000	A → Bd	None	Copy A to Bd
CBA		4E	0100 1110	Bd → A	None	Copy Bd to A
LBI	r,d	-- 33 --	00 r (d−1) (r = 0:3; d = 0, 9:15) or 0011 0011 1 r d (any r, any d)	r,d → B	Skip until not a LBI	Load B Immediate with r,d (Note 6)
LEI	y	33 6−	0011 0011 0110 y	y → EN	None	Load EN Immediate (Note 7)
XABR		12	0001 0010	A ⟷ Br (0 → A_3)	None	Exchange A with Br
TEST INSTRUCTIONS						
SKC		20	0010 0000		C = "1"	Skip if C is True
SKE		21	0010 0001		A = RAM(B)	Skip if A Equals RAM
SKGZ		33 21	0011 0011 0010 0001		$G_{3:0} = 0$	Skip if G is Zero (all 4 bits)
SKGBZ		33	0011 0011	1st byte		Skip if G Bit is Zero
	0	01	0000 0001	2nd byte	$G_0 = 0$	
	1	11	0001 0001		$G_1 = 0$	
	2	03	0000 0011		$G_2 = 0$	
	3	13	0001 0011		$G_3 = 0$	
SKMBZ	0	01	0000 0001		$RAM(B)_0 = 0$	Skip if RAM Bit is Zero
	1	11	0001 0001		$RAM(B)_1 = 0$	
	2	03	0000 0011		$RAM(B)_2 = 0$	
	3	13	0001 0011		$RAM(B)_3 = 0$	
SKT		41	0100 0001		A time-base counter carry has occurred since last test	Skip on Timer (Note 2)
INPUT/OUTPUT INSTRUCTIONS						
ING		33 2A	0011 0011 0010 1010	G → A	None	Input G Ports to A
ININ		33 28	0011 0011 0010 1000	IN → A	None	Input IN Inputs to A
INIL		33 29	0011 0011 0010 1001	IL_3, CKO, "0", IL_0 → A	None	Input IL Latches to A (Note 2)
INL		33 2E	0011 0011 0010 1110	$L_{7:4}$ → RAM(B) $L_{3:0}$ → A	None	Input L Ports to RAM, A
OBD		33 3E	0011 0011 0011 1110	Bd → D	None	Output Bd to D Outputs

TABLE II. COP404LSN-5 Instruction Set (Continued)

Mnemonic	Operand	Hex Code	Machine Language Code (Binary)	Data Flow	Skip Conditions	Description
INPUT/OUTPUT INSTRUCTIONS (Continued)						
OGI	y	33 5–	0011 | 0011 0101 | y	y → G	None	Output to G Ports Immediate
OMG		33 3A	0011 | 0011 0011 | 1010	RAM(B) → G	None	Output RAM to G Ports
XAS		4F	0100 | 1111	A ←→ SIO, C → SKL	None	Exchange A with SIO (Note 2)

Note 1: All subscripts for alphabetical symbols indicate bit numbers unless explicitly defined (e.g., Br and Bd are explicitly defined). Bits are numbered 0 to N where 0 signifies the least significant bit (low-order, right-most bit). For example, A_3 indicates the most significant (left-most) bit of the 4-bit A register.

Note 2: For additional information on the operation of the XAS, JID, LQID, INIL, and SKT instructions, see below.

Note 3: The JP instruction allows a jump, while in subroutine pages 2 or 3, to any ROM location within the two-page boundary of pages 2 or 3. The JP instruction, otherwise, permits a jump to a ROM location within the current 64-word page. JP may not jump to the last word of a page.

Note 4: A JSRP transfers program control to subroutine page 2 (0010 is loaded into the upper 4 bits of P). A JSRP may not be used when in pages 2 or 3. JSRP may not jump to the last word in page 2.

Note 5: LBI is a single-byte instruction if d = 0, 9, 10, 11, 12, 13, 14, or 15. The machine code for the lower 4 bits equals the binary value of the "d" data *minus 1*, e.g., to load the lower four bits of B (Bd) with the value 9 (1001_2), the lower 4 bits of the LBI instruction equal 8 (1000_2). To load 0, the lower 4 bits of the LBI instruction should equal 15 (1111_2).

Note 6: Machine code for operand field y for LEI instruction should equal the binary value to be latched into EN, where a "1" or "0" in each bit of EN corresponds to the selection or deselection of a particular function associated with each bit. (See Functional Description, EN Register.)

Description of Selection Instructions

The following information is provided to assist the user in understanding the operation of several unique instructions and to provide notes useful to programmers in writing COP404LSN-5 programs.

XAS INSTRUCTIONS

XAS (Exchange A with SIO) exchanges the 4-bit contents of the accumulator with the 4-bit contents of the SIO register. The contents of SIO will contain serial-in/serial-out shift register or binary counter data, depending on the value of the EN register. An XAS instruction will also affect the SK output. (See Functional Description, EN Register.) If SIO is selected as a shift register, an XAS instruction must be performed once every 4 instruction cycles to effect a continuous data stream.

JID INSTRUCTION

JID (Jump Indirect) is an indirect addressing instruction, transferring program control to a new ROM location pointed to indirectly by A and M. It loads the lower 8 bits of the ROM address register PC with the *contents* of ROM addressed by the 11-bit word, $PC_{10:8}$, A, M. PC_{10}, PC_9 and PC_8 are not affected by this instruction.

Note: JID requires 2 instruction cycles to execute.

INIL INSTRUCTION

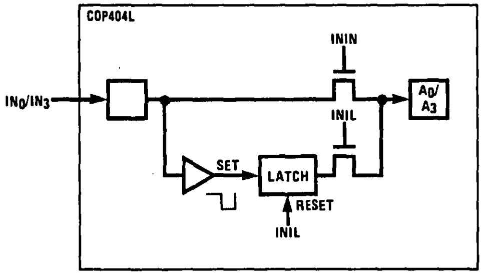
INIL (Input IL Latches to A) inputs 2 latches, IL_3 and IL_0 (see *Figure 7*) and CKO into A. The IL_3 and IL_0 latches are set if a low-going pulse ("1" to "0") has occurred on the IN_3 and IN_0 inputs since the last INIL instruction, provided the input pulse stays low for at least two instruction times. Execution of an INIL inputs IL_3 and IL_0 into A3 and A0 respectively, and resets these latches to allow them to respond to subsequent low-going pulses on the IN_3 and IN_0 lines. INIL will input "1" into A2 on the COP404LSN-5. A "0" is always placed in A1 upon the execution of an INIL. The general purpose inputs IN_3–IN_0 are input to A upon execution of an ININ instruction. (See Table II, ININ instruction.) INIL is useful in recognizing pulses of short duration or pulses which occur too often to be read conveniently by an ININ instruction.

Note: IL latches are not cleared on reset.

LQID INSTRUCTION

LQID (Load Q Indirect) loads the 8-bit Q register with the contents of ROM pointed to by the 11-bit word PC_{10}, PC_9, PC_8, A, M. LQID can be used for table lookup or code conversion such as BCD to seven-segment. The LQID instruction "pushes" the stack (PC + 1 → SA → SB → SC) and replaces the least significant 8 bits of PC as follows: A → $PC_{7:4}$, RAM(B) → $PC_{3:0}$, leaving PC_{10}, PC_9 and PC_8 unchanged. The ROM data pointed to by the new address is fetched and loaded into the Q latches. Next, the stack is "popped" (SC → SB → SA → PC), restoring the saved value of PC to continue sequential program execution. Since LQID pushes SB → SC, the previous contents of SC are lost. Also, when LQID pops the stack, the previously pushed contents of SB are left in SC. The net result is that the contents of SB are placed in SC (SB → SC).

Note: LQID takes two instruction cycle times to execute.



TL/DD/8817-12

FIGURE 7. INIL Hardware Implementation

Description of Selected Instructions (Continued)

SKT INSTRUCTION

The SKT (Skip On Timer) instruction tests the state of an internal 10-bit time-base counter. This counter divides the instruction cycle clock frequency by 1024 and provides a latched indication of counter overflow. The SKT instruction tests this latch, executing the next program instruction if the latch is not set. If the latch has been set since the previous test, the next program instruction is skipped and the latch is reset. The features associated with this instruction, therefore, allow the COP404LSN-5 to generate its own time-base for real-time processing rather than relying on an external input signal.

For example, using a 2.097 MHz oscillator as the time-base to the clock generator, the instruction cycle clock frequency will be 65 kHz (crystal frequency ÷ 32) and the binary counter output pulse frequency will be 64 Hz. For time-of-day or similar real-time processing, the SKT instruction can call a routine which increments a "seconds" counter every 64 ticks.

INSTRUCTION SET NOTES

a. The first word of a COP404LSN-5 program (ROM address 0) must be a CLRA (Clear A) instruction.

b. Although skipped instructions are not executed, one instruction cycle time is devoted to skipping each byte of the skipped instruction. Thus all program paths except JID and LQID take the same number of cycle times whether instructions are skipped or executed. JID and LQID instructions take 2 cycles if executed and 1 cycle if skipped.

c. The ROM is organized into 32 pages of 64 words each. The Program Counter is an 11-bit binary counter, and will count through page boundaries. If a JP, JSRP, JID or LQID instruction is located in the last word of a page, the instruction operates as if it were in the next page. For example: a JP located in the last word of a page will jump to a location in the next page. Also, a LQID or JID located in the last word of page 3, 7, 11, 15, 19, 23 or 27 will access data in the next group of four pages.

Typical Applications

PROM-BASED SYSTEM

The COP404LSN-5 may be used to exactly emulate the COP444L. *Figure 8* shows the interconnect to implement a COP444L hardware emulation. This connection uses a MM2716 EPROM as external memory. Other memory can be used such as bipolar PROM or RAM.

Pins IP7–IP0 are bidirectional inputs and outputs. When the AD/$\overline{\text{DATA}}$ clocking output turns on, the EPROM drivers are disabled and IP7–IP0 output addresses. The 8-bit latch (MM74LS373) latches the addresses to drive the memory.

When AD/$\overline{\text{DATA}}$ turns off, the EPROM is enabled and the IP7–IP0 pins will input the memory data. P8, P9 and SKIP/P10 output the most significant address bits to the memory. (SKIP output may be used for program debug if needed.)

The other 28 pins of the COP404LSN-5 may be configured exactly the same as a COP444L. The COP404LSN-5 V_{CC} can vary from 4.5V to 5.5V. However, 5V is used for the memory.

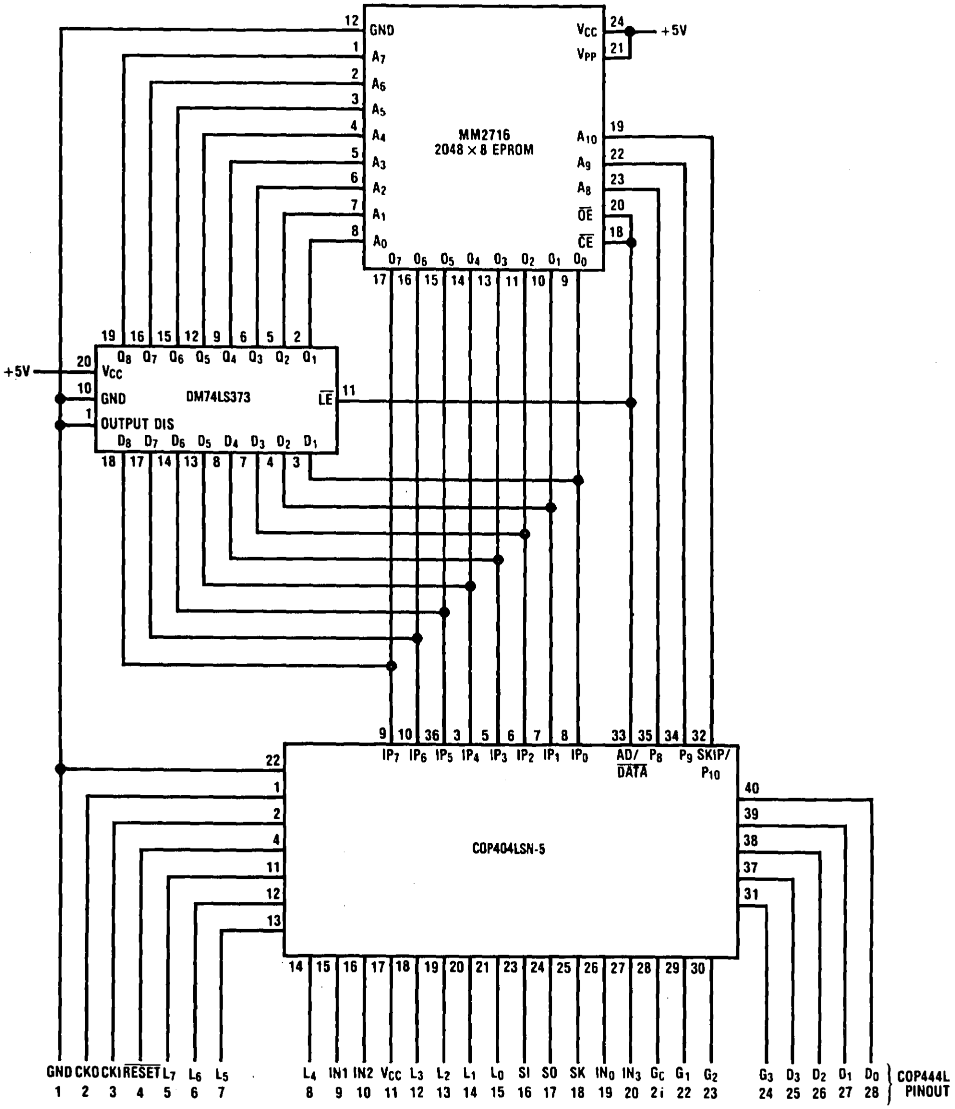
For In-Circuit emulation, see also COP444LP.

COP404LSN-5 Mask Options

The following COP444L options have been implemented on the COP404LSN-5.

Option Value	Comment	Option Value	Comment
Option 1 = 0	Ground, no option available	Option 18 = 2	SK has push-pull output
Option 2 = 0	CKO is clock generator output to crystal/resonator	Option 19 = 0	IN0 has load device to V_{CC}
		Option 20 = 0	IN3 has load device to V_{CC}
Option 3 = 0	CKI is oscillator input (divide by 32)	Option 21 = 0	G_0
Option 4 = 0	$\overline{\text{RESET}}$ pin has load device to V_{CC}	Option 22 = 0	G_1 have high current
Option 5 = 2	L_7	Option 23 = 0	G_2 standard output
Option 6 = 2	L_6 have LED direct-drive	Option 24 = 0	G_3
Option 7 = 2	L_5 output	Option 25 = 0	D_3
Option 8 = 2	L_4	Option 26 = 0	D_2 have high current
Option 9 = 0	IN1 has load device to V_{CC}	Option 27 = 0	D_1 standard output
Option 10 = 0	IN2 has load device to V_{CC}	Option 28 = 0	D_0
Option 11 = 1	V_{CC} 4.5V to 5.5V operation	Option 29 = 1	L
Option 12 = 2	L_3	Option 30 = 1	IN have higher voltage
Option 13 = 2	L_2 have LED direct-drive	Option 31 = 1	G input levels
Option 14 = 2	L_1 output	Option 32 = 0	SI has standard input level
Option 15 = 2	L_0	Option 33 = 0	$\overline{\text{RESET}}$ has Schmitt trigger input
Option 16 = 0	SI has load to V_{CC}	Option 34 = 0	CKO has standard input levels
Option 17 = 2	SO has push-pull output	Option 35 = N/A	40-pin package

Typical Applications (Continued)



TL/DD/8817-13

FIGURE 8. COP404LSN-5 System Diagram

National Semiconductor

PRELIMINARY

COP420P/COP444CP/COP444LP Piggyback EPROM Microcontrollers

General Description

The COP420P, COP444CP, and COP444LP are piggyback versions of the COPS™ microcontroller families. These devices are identical to their respective device except the program ROM has been removed. The device package incorporates the circuitry and socket on top of package to accommodate the piggyback EPROM—MM2716, NMC27C16 or other appropriate EPROMs. With the addition of an EPROM, the device performs exactly as its masked equivalent.

The device is a complete microcontroller system with CPU, RAM, I/O and EPROM socket in a 28-lead package. The completed package allows field test of the system in the final electrical and mechanical configuration. This important benefit facilitates development and debug of the COP400 program prior to masking of a production part.

These devices are also economical in low and medium volume applications or when the program may require changing.

Device Selection	Device Emulated	Piggyback Device
Low Power NMOS	COP420L, COP444L	COP444LP
High Speed NMOS	COP420	COP420P
Low Power CMOS	COP424C, COP444C	COP444CP

Features

COP444LP

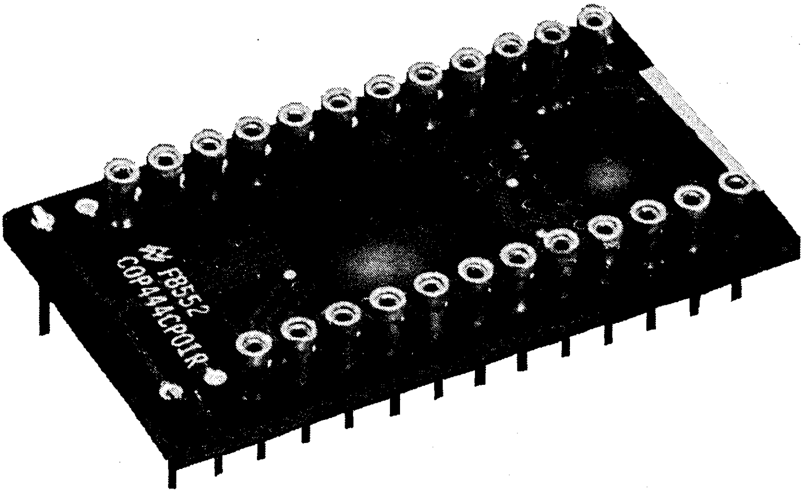
- 16 μs instruction time
- Same Specification as COP404LSN-5

COP420P

- 4 μs instruction time
- Same Specification as COP402N

COP444CP

- 4 μs instruction time
- Fully static (can turn off clock)
- Power-saving IDLE state and Halt mode
- Same Specification as COP404CN



TL/DD/8705-10

COP420P Absolute Maximum Ratings

Voltage at Any Pin	−0.3V to +7V
Operating Temperature Range	
COP420P	0°C to 70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Total Sink Current	50 mA
Total Source Current	70 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

COP420P DC Electrical Characteristics

$0°C \leq T_A \leq 70°C$, $4.5V \leq V_{CC} \leq 5.5V$ unless otherwise noted

Parameter	Conditions	Min	Max	Units
Operation Voltage		4.5	5.5	V
Power Supply Ripple	Peak to Peak (Note 3)		0.4	V
Supply Current	All Outputs Open		81	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High	V_{CC} = 5.5V	3.0		V
Logic High	V_{CC} = 4.5V	2.0		V
Logic Low		−0.3	0.4	V
Schmitt Trigger Input				
RESET				
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
All Other Inputs				
Logic High	V_{CC} = Max	3.0		V
Logic High	V_{CC} = 5V ±10%	2.0		V
Logic Low		−0.3	0.8	V
Input Load Source Current	V_{CC} = 5V, V_{IN} = 0V	−100	−800	μA
Input Capacitance			7	pF
Hi-Z Input Leakage		−1	+1	μA
Output Voltage Levels				
D, G, L, SK, SO Outputs				
TTL Operation	V_{CC} = 5V ±10%			
Logic High	I_{OH} = −100 μA	2.0		V
Logic Low	I_{OL} = 1.6 mA	−0.3	0.4	V
IP0–IP7, P8, P9, SKIP, CKO, AD/$\overline{\text{DATA}}$				
Logic High	I_{OH} = −75 μA	2.4		V
Logic Low	I_{OL} = 400 μA	−0.3	0.4	V
CMOS Operation (Note 2)				
Logic High	I_{OH} = −10 μA	V_{CC} − 1		V
Logic Low	I_{OL} = 10 μA	−0.3	0.2	V
Output Current Levels				
LED Direct Drive (Note 3)	V_{CC} = 5.0V			
Logic High	V_{OH} = 2.0V	1.0	14	mA
Allowable Sink Current				
Per Pin (L, D, G)			10	mA
Per Pin (All Others)			2	mA
Per Port (L)			16	mA
Per Port (D, G)			10	mA
Allowable Source Current				
Per Pin (L)			−15	mA
Per Pin (All Others)			−1.5	mA

COP420P AC Electrical Characteristics

0°C ≤ T_A ≤ 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		4	10	µs
Operating CKI Frequency	÷16 Mode	1.6	4.0	MHz
CKI Duty Cycle (Note 1)		40	60	%
Rise Time	Frequency = 4 MHz		60	ns
Fall Time	Frequency = 4 MHz		40	ns
Inputs				
SI				
t_{SETUP}		0.3		µs
t_{HOLD}		250		ns
All Other Inputs				
t_{SETUP}		1.7		µs
t_{HOLD}		300		ns
Output Propagation Delay	R_L = 5k, C_L = 50 pF, V_{OUT} = 1.5V			
SO and SK				
t_{pd1}			1.0	µs
t_{pd0}			1.0	µs
CKO				
t_{pd1}			0.25	µs
t_{pd0}			0.25	µs
AD/DATA, SKIP				
t_{pd1}			0.6	µs
t_{pd0}			0.6	µs
All Other Outputs				
t_{pd1}			1.4	µs
t_{pd0}			1.4	µs

Note 1: Duty cycle = $t_{W1}/(t_{W1} + t_{W0})$.

Note 2: Voltage change must be less than 0.5V in a 1 ms period.

Note 3: Exercise great care not to exceed maximum device power dissipation limits when direct driving LEDs (or sourcing similar loads) at high temperature.

COP444CP Absolute Maximum Ratings

Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Allowable Source Current	25 mA
Total Allowable Sink Current	25 mA
Operating Temperature Range	0°C to 70°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

COP444CP DC Electrical Characteristics

$0°C < T_A < 70°C$, $4.5V \leq V_{CC} \leq 5.5V$ unless otherwise specified

Parameter	Conditions	Min	Max	Units
Operating Voltage		4.5	5.5	V
Power Supply Ripple (Note 3)	Peak to Peak		$0.1 V_{CC}$	V
Supply Current (Note 1)	$V_{CC} = 5V$, $t_C = 4\ \mu s$		15	mA
Input Voltage Levels				
RESET, D0				
Logic High		$0.9 V_{CC}$		V
Logic Low			$0.1 V_{CC}$	V
All Other Inputs				
Logic High		$0.7 V_{CC}$		V
Logic Low			$0.2 V_{CC}$	V
Input Pull-Up Current	$V_{CC} = 4.5V$, $V_{IN} = 0$	30	330	μA
Hi-Z Input Leakage		−1	+1	μA
Input Capacitance			7	pF
Output Voltage Levels	Standard Outputs			
LSTTL Operation	$V_{CC} = 5.0V \pm 5\%$			
Logic High	$I_{OH} = -100\ \mu A$	2.7		V
Logic Low	$I_{OL} = 400\ \mu A$		0.4	V
CMOS Operation				
Logic High	$I_{OH} = -10\ \mu A$	$V_{CC} - 0.2$		V
Logic Low	$I_{OL} = 10\ \mu A$		0.2	V
Output Current Levels				
Sink (Note 6)	$V_{CC} = 4.5V$, $V_{OUT} = V_{CC}$	1.2		mA
Source (Standard Option)	$V_{CC} = 4.5V$, $V_{OUT} = 0V$	0.5		mA
Source (Low Current Option)	$V_{CC} = 4.5V$, $V_{OUT} = 0V$	30	330	μA
Allowable Sink/Source Current Per Pin (Note 4)			5	mA
Allowable Loading on CKOH			100	pF
Current Needed to Over-Ride HALT (Note 3)				
To Continue	$V_{CC} = 4.5V$, $V_{IN} = 2 V_{CC}$		0.7	mA
To Halt	$V_{CC} = 4.5V$, $V_{IN} = 7 V_{CC}$		1.6	mA
TRI-STATE Leakage Current		−2.5	+2.5	μA

COP444CP AC Electrical Characteristics

0°C < T_A < 70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise specified

Parameter	Conditions	Min	Max	Units
Instruction Cycle Time (t_C)	$V_{CC} \geq 4.5V$	4	DC	μs
Operating CKI Frequency	$V_{CC} \geq 4.5V$	DC	1.0	MHz
Inputs				
t_{SETUP}	G Inputs }	$t_C/4 + 0.7$		μs
	SI Input } $V_{CC} \geq 4.5V$	0.3		μs
	IP Input }	1.0		μs
	All Others }	1.7		μs
t_{CLOCK}	$V_{CC} \geq 4.5V$	0.25		μs
Output Propagation Delay	$V_{OUT} = 1.5V, C_L = 100\text{ pF}, R_L = 5k$			
IP7–IP0, A10–A8, SKIP				
$t_{(pd1)}, T_{(pd0)}$	$V_{CC} \geq 4.5V$		1.94	μs
AD/$\overline{\text{DATA}}$				
$t_{(pd1)}, t_{(pd0)}$	$V_{CC} \geq 4.5V$		375	μs
All Other Outputs				
$t_{(pd1)}, t_{(pd0)}$	$V_{CC} > 4.5V$		1.0	μs

Note 1: Supply current is measured after running for 2000 cycle times with a square-wave clock on CKI and all other pins pulled up to V_{CC} with 20k resistors.

Note 2: When forcing HALT, current is only needed for a short time (approx. 200 ns) to flip the HALT flip-flop.

Note 3: Voltage change must be less than 0.5V in a 1 ms period.

Note 4: SO output sink current must be limited to keep V_{OL} less than 0.2 V_{CC} (i.e., 0.1 mA at 2.4V V_{CC} and 0.5 mA at 4.5V V_{CC}).

COP444LP Absolute Maximum Ratings

Voltage at Any Pin Relative to GND	−0.5V to +10V
Ambient Operating Temperature	0°C to +70°C
Ambient Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	0.75W at 25°C 0.4W at 70°C

Total Source Current	120 mA
Total Sink Current	140 mA

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

COP444LP DC Electrical Characteristics

0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

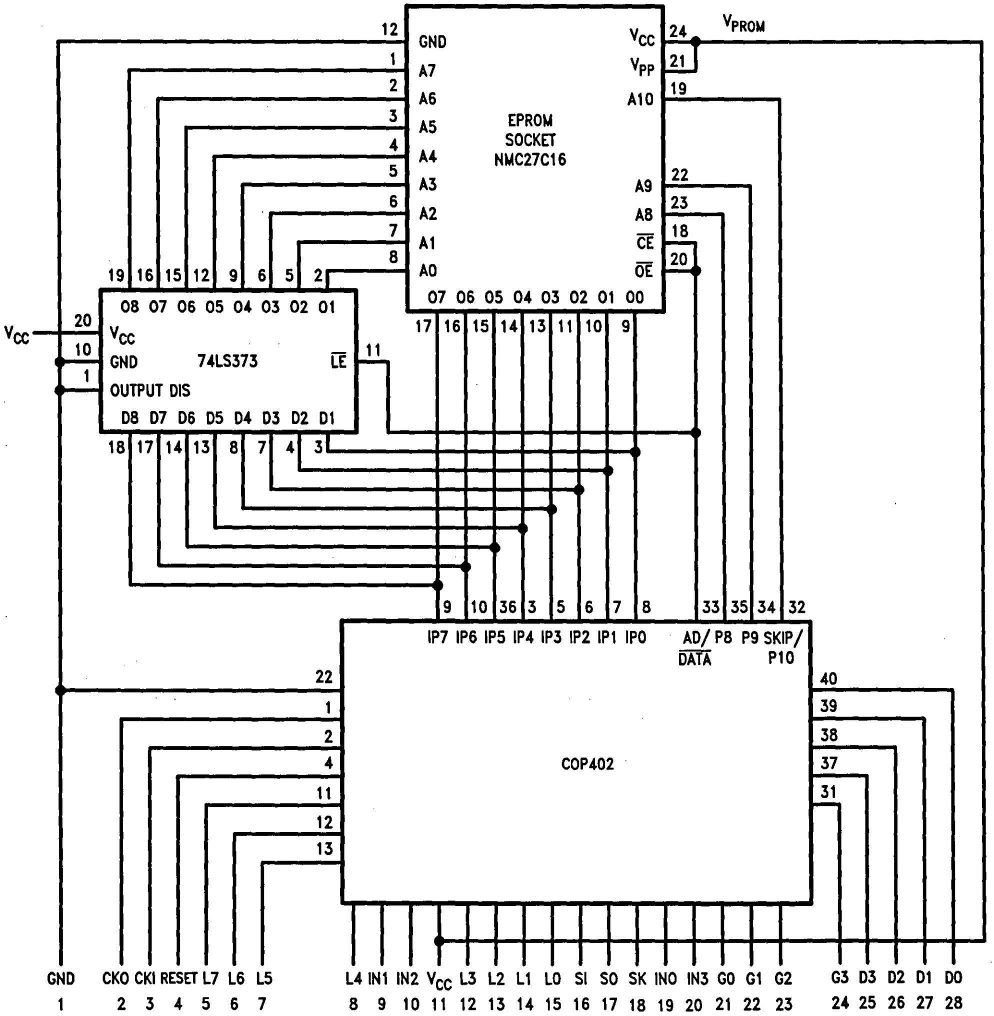
Parameter	Conditions	Min	Max	Units
Operating Voltage (V_{CC})	(Note 1)	4.5	5.5	V
Power Supply Ripple	Peak to Peak		0.5	V
Operating Supply Current			66	mA
Input Voltage Levels				
CKI Input Levels				
Crystal Input				
Logic High (V_{IH})	V_{CC} = 5.5V	3.0		V
Logic High (V_{IH})	V_{CC} = 4.5V	2.0		V
Logic Low (V_{IL})		−0.3	0.4	V
$\overline{\text{RESET}}$ Input Levels	Schmitt Trigger Input			
Logic High		0.7 V_{CC}		V
Logic Low		−0.3	0.6	V
IP0–IP7, SI Input Levels				
Logic High	*V_{CC} = 5.5V	2.4		V
Logic High	V_{CC} = 5V ±5%	2.0		V
Logic Low		−0.3	0.8	V
All Other Inputs				
Logic High	High Trip Level Options	3.6		V
Logic Low		−0.3	1.2	V
Input Capacitance			7	pF
Output Voltage Levels				
LSTTL Operation	V_{CC} = 5V ±5%			
Logic High (V_{OH})	I_{OH} = 25 μA	2.7		V
Logic Low (V_{OL})	I_{OL} = 0.36 mA		0.4	V
Output Current Levels				
Output Sink Current				
SO and SK Outputs (I_{OL})	*V_{CC} = 4.5V, V_{OL} = 0.4V	0.9		mA
L0–L7 Outputs	*V_{CC} = 4.5V, V_{OL} = 0.4V	0.4		mA
G0–G3 and D0–D3 Outputs	*V_{CC} = 4.5V, V_{OL} = 1.0V	7.5		mA
CKO	*V_{CC} = 4.5V, V_{OL} = 0.4V	0.2		mA
Output Source Current				
D0–D3, G0–G3 Outputs (I_{OH})	*V_{CC} = 4.5V, V_{OH} = 2.0V	−30	−250	μA
SO and SK Outputs (I_{OH})	*V_{CC} = 4.5V, V_{OH} = 1.0V	1.2		mA
L0–L7 Outputs	*V_{CC} = 4.5V, V_{OH} = 2.0V	−1.4	−20	mA
Input Load Source Current (I_{IL})	V_{CC} = 5.0V, V_{IL} = 0V	−10	−140	μA
Total Sink Current Allowed				
All Outputs Combined			140	mA
D, G Ports			120	mA
L7–L4			4	mA
L3–L0			4	mA
All Other Pins			1.8	mA
Total Source Current Allowed				
All I/O Combined			120	mA
L7–L4			60	mA
L3–L0			60	mA
Each L Pin			30	mA
All Other Pins			1.4	mA

COP444LP AC Electrical Characteristics

0°C ≤ T_A ≤ +70°C, 4.5V ≤ V_{CC} ≤ 5.5V unless otherwise noted

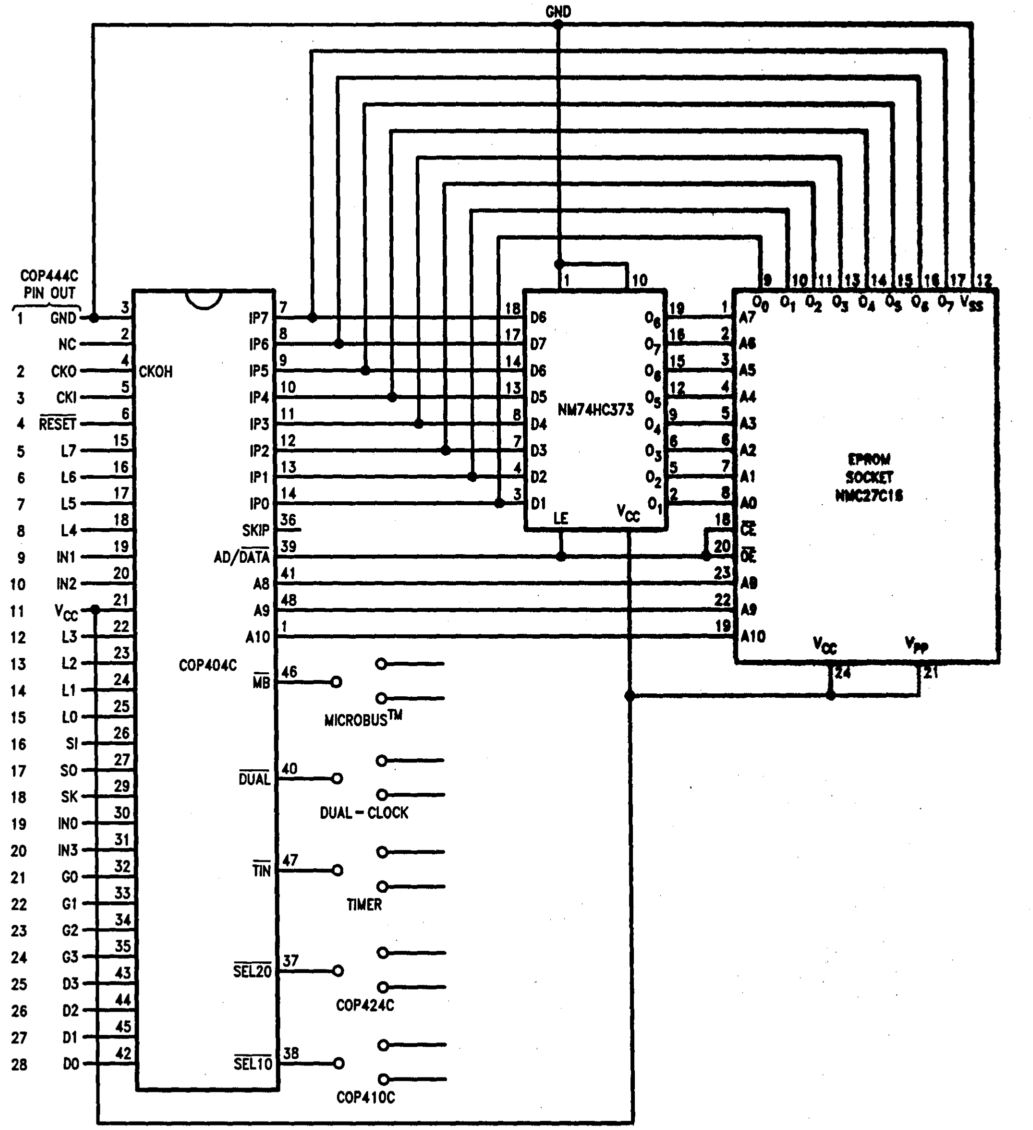
Parameter	Conditions	Min	Max	Units
Instruction Cycle Time		16	40	µs
CKI				
Input Frequency f_I	÷ 32 mode	0.8	2.0	MHz
Duty Cycle		30	60	%
Rise Time	f_I = 2.0 MHz		120	ns
Fall Time			80	ns
Inputs				
SI, IP7–IP0				
t_{SETUP}			2.0	µs
t_{HOLD}			1.0	µs
IN3–IN0, G3–G0, L7–L0				
t_{SETUP}			8.0	µs
t_{HOLD}			1.3	µs
Output Propagation Delay	C_L = 50 pF, V_{OUT} = 1.5V			
SO, SK Outputs	R_L = 20 kΩ		4.0	µs
t_{pd1}, t_{pd0}				
D3–D0, G3–G0, L7–L0	R_L = 20 kΩ		5.6	µs
t_{pd1}, t_{pd0}				
A0–A7			7.5	µs
t_{pd1}, t_{pd0}			11.5	µs
A_8, A_9				
t_{pd1}, t_{pd0}			6.0	µs
A_{10}				
t_{pd1}, t_{pd0}				

Note 1: V_{CC} voltage change must be less than 0.5V in a 1 ms period to maintain proper operation.



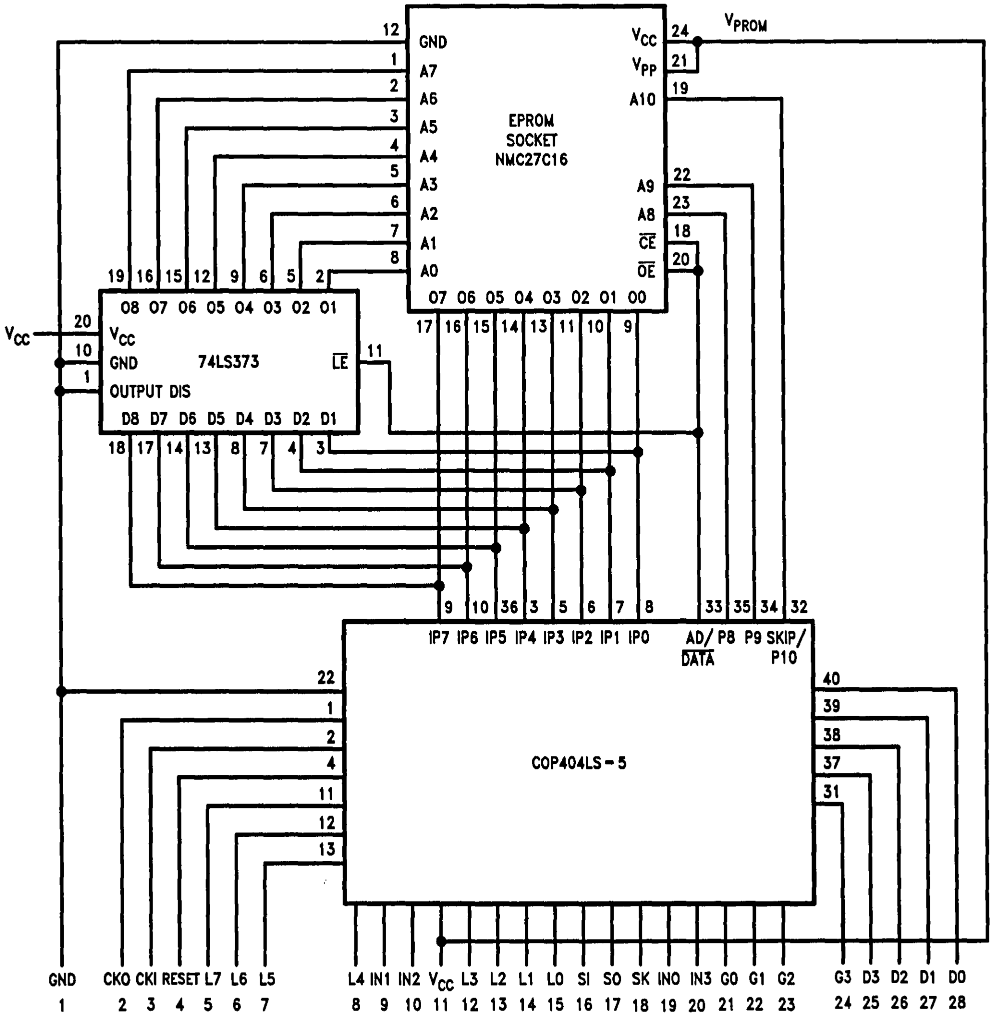
TL/DD/8705-1

FIGURE 1. COP420P Block Diagram



TL/DD/8705-2

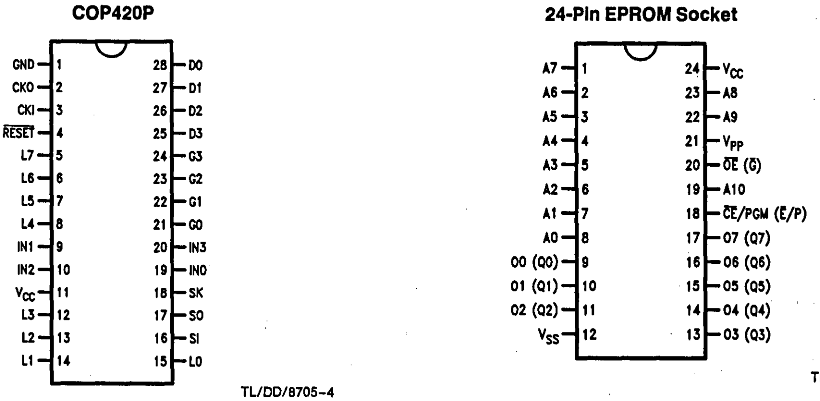
FIGURE 2. COP444C Block Diagram



TL/DD/8705-3

FIGURE 3. COP444LP Block Diagram

Connection Diagrams



COP420P

24-Pin EPROM Socket

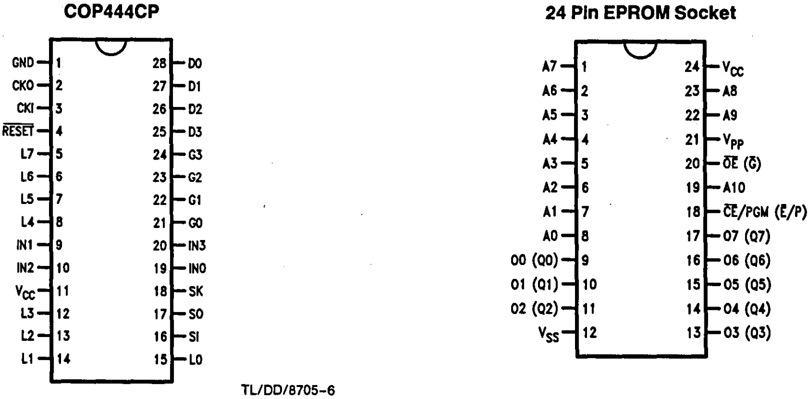
TL/DD/8705-4

TL/DD/8705-5

FIGURE 4. COP420P Connection Diagrams

Pin	Description
L_7–L_0	8 Bidirectional I/O Ports with TRI-STATE
G_3–G_0	4 Bidirectional I/O Ports
D_3–D_0	4 General Purpose Outputs
IN_3–IN_0	4 General Purpose Inputs
SI	Serial Input (or Counter Input)
SO	Serial Output (or General Purpose Output)
SK	Logic-Controlled Clock (or General Purpose Output)

Pin	Description
AD/$\overline{\text{DATA}}$	Address Out/Data In Flag
CKI	System Oscillator Input
CKO	Clock Generator Output to Crystal/Resonator
$\overline{\text{RESET}}$	System Reset Input
V_{CC}	Power Supply
GND	Ground
O_7–O_0	PROM Data Lines
A_9–A_0	PROM Address Outputs



COP444CP

24 Pin EPROM Socket

TL/DD/8705-6

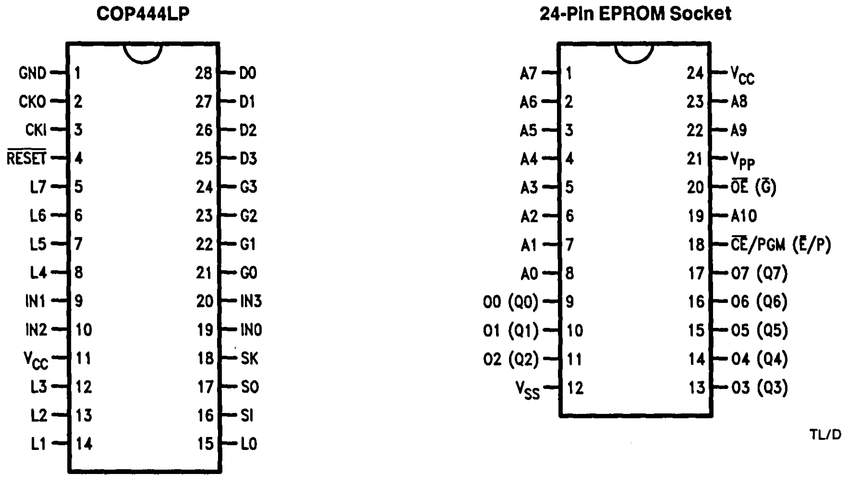
TL/DD/8705-7

FIGURE 5. COP444CP Connection Diagrams

Pin	Description
L_7–L_0	8 Bidirectional I/O Ports with TRI-STATE
G_3–G_0	4 Bidirectional Very High Current Standard Output
D_3–D_0	4 General Very High Current Standard Output
IN_3–IN_0	4 General Purpose Inputs
SI	Serial Input (or Counter Input)
SO	Serial Output (or General Purpose Output)
SK	Logic-Controlled Clock (or General Purpose Output)

Pin	Description
AD/$\overline{\text{DATA}}$	Address Out/Data In Flag
CKI	System Oscillator Input
CKO	Clock Generator Output to Crystal/Resonator
$\overline{\text{RESET}}$	System Reset Input
V_{CC}	Power Supply
GND	Ground
O_7–O_0	PROM Data Lines
A_{10}–A_0	PROM Address Outputs

Connection Diagrams (Continued)



TL/DD/8705-9

TL/DD/8705-8

FIGURE 6. COP444LP Connection Diagrams

Pin	Description	Pin	Description
L_7–L_0	8 LED Direct Drive	AD/$\overline{\text{DATA}}$	Address Out/Data In Flag
G_3–G_0	4 Bidirectional Low Current I/O Ports	CKI	System Oscillator Input
D_3–D_0	4 General Purpose Outputs	CKO	Clock Generator Output to Crystal/Resonator
IN_3–IN_0	4 General Purpose Inputs	$\overline{\text{RESET}}$	System Reset Input
SI	Serial Input (or Counter Input)	V_{CC}	Power Supply
SO	Serial Output (or General Purpose Output)	GND	Ground
SK	Logic-Controlled Clock (or General Purpose Output)	O_7–O_0	PROM Data Lines
		A_{10}–A_0	PROM Address Outputs

COP420 (COP444LP) Mask Options

The following COP420 (COP444L) options have been implemented in the COP420P (COP444LP):

Option Value	Comment
Option 1 = 0	GND pin—no option available
Option 2 = 0	CKO is clock generator output to crystal
Option 3 = 0	CKI is crystal input ÷ 16 (÷ 32 COP444LP)
Option 4 = 0	RESET pin has load device to V_{CC}
Option 5–8 = 2	L outputs have LED direct-drive
Option 9 = 0	IN1 has load device to V_{CC}
Option 10 = 0	IN2 has load device to V_{CC}
Option 11 = 0 (COP420P)	V_{CC} pin—no option available
(Option 11 = 1 COP444LP)	V_{CC} pin—4.5V–5.5V operation
Option 12–15 = 2	L outputs have LED direct-drive
Option 16 = 0	SI has load device to V_{CC}
Option 17 = 2	SO has push-pull output
Option 18 = 2	SK has push-pull output
Option 19 = 0	IN0 has load device to V_{CC}
Option 20 = 0	IN3 has load device to V_{CC}
Option 21–24 = 0	G outputs are standard (COP420P). G outputs have very high current standard output (COP444LP)
Option 25–28 = 0	D outputs are standard (COP420P). D outputs have very high current standard output. (COP444LP)
Option 29 = 0 (COP420P)	Normal operation
(Option 29 = 1 COP444LP)	L has higher voltage input levels
Option 30 = 0 (COP420P)	28-pin package
(Option 30 = 1 COP444LP)	IN has higher voltage input levels
Option 31 = 0 (COP420P)	IN has standard input levels
(Option 31 = 1 COP444LP)	G has higher voltage input levels
Option 32 = 0	G has standard input levels (COP420P). SI has standard input levels (COP444LP)
Option 33 = 0	L has standard input levels (COP420P). RESET has Schmitt trigger input (COP444LP)
Option 34 = 0	No option
Option 35 = 0	SI has standard input levels (COP420P). 28-pin package (COP444LP)

COP444CP Mask Options

The following COP444C options have been implemented in the COP444CP:

Option Value	Comment
Option 1 = 0	GND pin—no option available
Option 2 = 1	CKO is HALT I/O
Option 3 = 5	CKI is external clock input ÷ 4
Option 4 = 1	RESET is Hi-Z input
Option 5–8 = 0	L outputs are standard TRI-STATE
Option 9 = 1	IN1 is a Hi-Z input
Option 10 = 1	IN2 is a Hi-Z input
Option 11 = 0	V_{CC} pin (4.5V–5.5V)
Option 12–15 = 0	L outputs are standard TRI-STATE
Option 16 = 0	SI is a Hi-Z input
Option 17 = 0	SO is a standard output
Option 18 = 0	SK is a standard output
Option 19 = 1	IN0 is a Hi-Z input
Option 20 = 1	IN3 is a Hi-Z input
Option 21–24 = 1	G outputs are low current
Option 25–28 = 0	D outputs are standard
Option 29 = 1	No internal initialization logic
Option 30 = 0	Normal operation
Option 31 = 0	Time-base counter
Option 32 = 0	Normal
Option 33 = 0	28-pin package

Section 2
COP800 Family

Section 2 Contents

National Semiconductor

The 8-Bit COP800 Family: Optimized for Value

National's COP800 family provides cost-effective solutions for feature-rich, 8-bit microcontroller applications.

Key Features

- High-performance 8-bit microcontroller
- Full 8-bit architecture and implementation
- 1 μs instruction-cycle time
- High code efficiency with single-byte, multiple-function instructions
- UART
- A/D converter
- Watchdog/clock monitor
- On-chip ROM from 1 kbyte
- On-chip RAM to 192 bytes
- EEPROM
- M^{2}CMOS™ fabrication
- MICROWIRE/PLUS™ serial interface
- ROMless versions available
- Wide operating voltage range: +2.5V to +6V
- Military temp range available: −55°C to +125°C
- MIL-STD-883C versions available
- 20- to 44-pin packages

The COP800 combines a powerful single-byte, multiple-function instruction set with a memory-mapped core architecture similar to the HPC™.

And like the HPC, the COP800 family supports a wide variety of ROM, RAM, I/O and peripheral functions.

The COP800 has an instruction-cycle time of only 1 μs, and because over 70% of its instruction set is composed of single-cycle, single-byte instructions, the COP800 can deliver exceptional performance for an 8-bit engine.

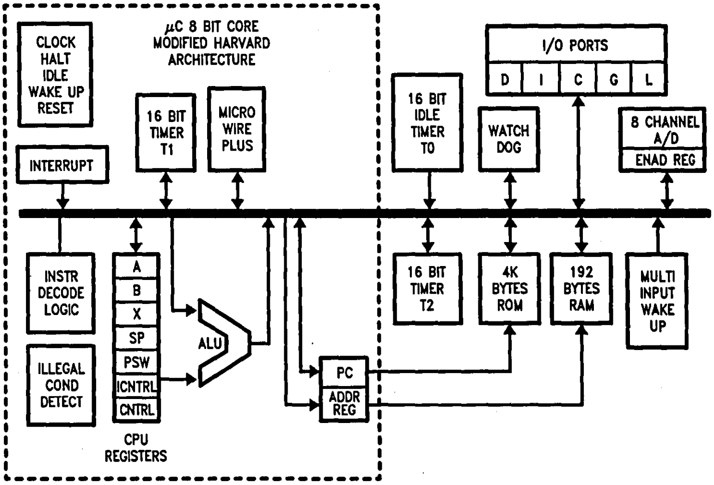
And since it's fabricated in National's advanced M^{2}CMOS process, the COP800 has low current drain, low heat dissipation, and a wide operating voltage range.

Key Applications

- Automotive systems
- Process control
- Robotics
- Telecommunications
- AC-motor control
- DC-motor control
- Keyboard controllers
- Modems
- RS232C controllers

The COP800 family offers high performance in a low-cost, easy-to-design-in package.

COP888CF Block Diagram



TL/XX/0073-3

COP800 Family of Microcontrollers

Commercial Temp Version 0°C to +70°C	Industrial Temp Version −40°C to +85°C	Military Temp Version −55°C to +125°C	Memory		Features						
			ROM (Bytes)	RAM (Bytes)	I/O		Interrupt	Stack	Timer Base Counters	Size (Pins)	Other
					I/O Pins	Serial I/O					
	COP820C	COP620C	1.0k	64	24	Yes	3 Sources	In RAM	1	28	
	COP821C	COP621C	1.0k	64	20	Yes	3 Sources	In RAM	1	24	
	COP822C	COP622C	1.0k	64	16	Yes	3 Sources	In RAM	1	20	
	COP8640		2.0k	64	24	Yes	3 Sources	In RAM	1	28	64 x 8
	COP8641		2.0k	64	20	Yes	3 Sources	In RAM	1	24	EEPROM
	COP8642		2.0k	64	16	Yes	3 Sources	In RAM	1	20	in RAM
	COP8620		1.0k	64	24	Yes	3 Sources	In RAM	1	28	64 x 8
	COP8621		1.0k	64	20	Yes	3 Sources	In RAM	1	24	EEPROM
	COP8622		1.0k	64	16	Yes	3 Sources	In RAM	1	20	in RAM
	COP8720C		1.0k EE	64	24	Yes	3 Sources	In RAM	1	28	64 x 8 EEPROM in RAM
	COP8721C		1.0k EE	64	20	Yes	3 Sources	In RAM	1	24	64 x 8 EEPROM in RAM
	COP8722C		1.0k EE	64	16	Yes	3 Sources	In RAM	1	20	64 x 8 EEPROM in RAM
	COP840C	COP640C	2.0k	128	24	Yes	3 Sources	In RAM	1	28	
	COP841C	COP641C	2.0k	128	20	Yes	3 Sources	In RAM	1	24	
	COP842C	COP642C	2.0k	128	16	Yes	3 Sources	In RAM	1	20	
	COP884CF	COP684CF	4.0k	128	21	Yes	10 Sources	In RAM	2	28	2 PWM & A/D
	COP884CG	COP684CG	4.0k	192	23	Yes	12 Sources	In RAM	3	28	3 PWM & UART
	COP884CL	COP684CL	4.0k	128	23	Yes	10 Sources	In RAM	2	28	2 PWM
	COP888CF	COP688CF	4.0k	128	33/37	Yes	10 Sources	In RAM	2	40/44	2 PWM & A/D
	COP888CG	COP688CG	4.0k	192	35/39	Yes	14 Sources	In RAM	3	40/44	3 PWM & UART
	COP888CL	COP688CL	4.0k	128	33/39	Yes	10 Sources	In RAM	2	40/44	2 PWM

Development Support

DEVELOPMENT SYSTEM

The Microcomputer On Line Emulator Development System is a low cost development system and emulator for all microcontroller products. These include COPS™ microcontrollers and the HPC family of products. The COP800 Development System consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the Development System is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other Development Systems in a multi-Development System environment.

The Development System can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

Development Support (Continued)

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	TBD	Programmer's Manual		TBD

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package contains:

DIAL-A-HELPER Users Manual
Public Domain Communications Software

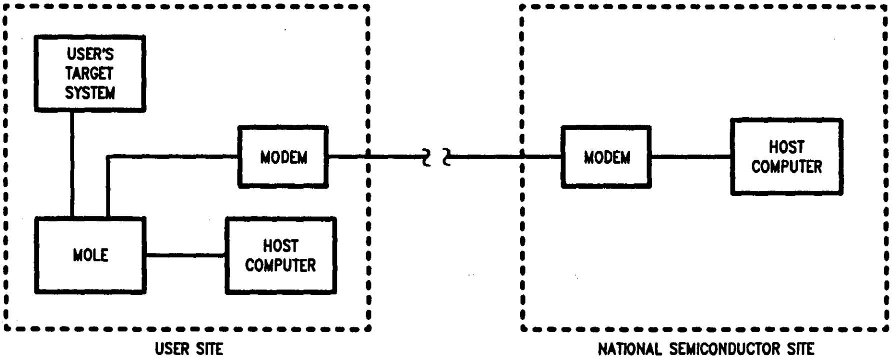
FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Development Support (Continued)

Voice:	(408) 721-5582
Modem:	(408) 739-1162
Baud:	300 or 1200 baud
Set-Up:	Length: 8-bit Parity: none Stop Bit: 1
Operation:	24 hrs., 7 days

DIAL-A-HELPER



TL/XX/0073-2

National Semiconductor

COP620C/COP621C/COP622C/COP640C/COP641C/ COP642C/COP820C/COP821C/COP822C/COP840C/ COP841C/COP842C Single-Chip microCMOS Microcontrollers

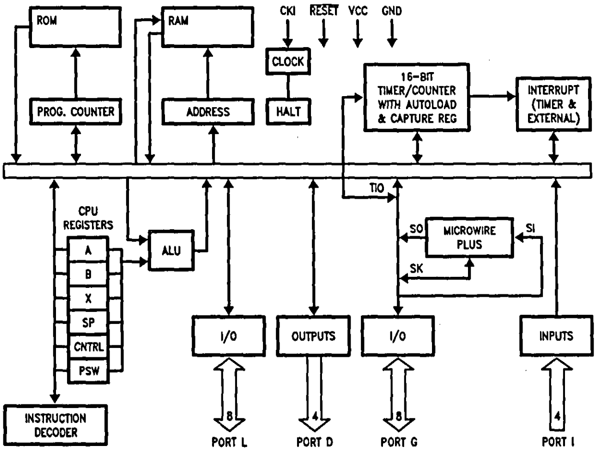
General Description

The COP820C and COP840C are members of the COPS™ microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP820C and COP840C to the specific application. The part operates over a voltage range of 2.5 to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate. The part may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Features

- Low Cost 8-bit microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain (2.2 mA at 3 μs instruction rate)
  Low current static HALT mode (Typically < 1 μA)
- Single supply operation: 2.5 to 6.0V
- 1024 bytes ROM/64 Bytes RAM—COP820C
- 2048 bytes ROM/128 Bytes RAM—COP840C
- 16-bit read/write timer operates in a variety of modes
  - Timer with 16-bit auto reload register
  - 16-bit external event counter
  - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - Reset master clear
  - External interrupt with selectable edge
  - Timer interrupt or capture interrupt
  - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE PLUS™ serial I/O
- 28 pin package (optionally 24 or 20 pin package)
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature ranges: −40°C to +85°C, −55°C to +125°C
- ROMless mode for accurate emulation and external program capability—expandable to 32k bytes in ROMless mode
- Form, fit and function EEPROM emulation device (COP8720C)
- Piggyback emulation devices (COP820CP/COP840CP)
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/9103-1

FIGURE 1

COP820C/COP821C/COP822C/COP840C/COP841C/COP842C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	50 mA
Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	$-65°C$ to $+140°C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		2.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Supply Current					
High Speed Mode, CKI = 20 MHz	V_{CC} = 6V, tc = 1 µs			9	mA
Normal Mode, CKI = 5 MHz	V_{CC} = 6V, tc = 2 µs			4	mA
Normal Mode, CKI = 2 MHz	V_{CC} = 2.5V, tc = 5 µs			0.7	mA
(Note 2)					
HALT Current	V_{CC} = 6V, CKI = 0 MHz		<1	10	µA
(Note 3)					
Input Levels					
$\overline{\text{RESET}}$, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6.0V	−2		+2	µA
Input Pullup Current	V_{CC} = 6.0V	40		250	µA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	2			mA
All Others					
Source (Weak Pull-Up)	V_{CC} = 4.5V, V_{OH} = 3.2V	10		110	µA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			
TRI-STATE Leakage		−2.0		+2.0	µA
Allowable Sink/Source					
Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Note 5)					
Without Latchup (Room Temp)	Room Temp			±100	mA
RAM Retention Voltage, Vr	500 ns Rise and				
	Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Human body mode, 100 pF through 1500Ω.

Note 5: Except pins G6, G7, $\overline{\text{RESET}}$
pins G6, $\overline{\text{RESET}}$: +60 mA, −100 mA
pin G7: +100 mA, −25 mA
Sampled but not 100% tested.

COP820C/COP821C/COP822C/COP840C/COP841C/COP842C

AC Electrical Characteristics $-40°C < T_A < +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$	1		DC	μs
(Div-by 20)	$2.5V \leq V_{CC} < 4.5V$	2.5		DC	μs
Normal Mode	$V_{CC} \geq 4.5V$	2		DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$	5		DC	μs
R/C Oscillator Mode	$V_{CC} \geq 4.5V$	3		DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 6)	fr = Max (÷20 Mode)	33		66	%
Rise Time (Note 6)	fr = 20 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 20 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	200			ns
	$2.5V \leq V_{CC} < 4.5V$	500			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	60			ns
	$2.5V \leq V_{CC} < 4.5V$	150			ns
Output Propagation Delay	$C_L = 100$ pF, $R_L = 2.2$ kΩ				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$			0.7	μs
	$2.5V \leq V_{CC} < 4.5V$			1.75	μs
All Others	$V_{CC} \geq 4.5V$			1	μs
	$2.5V \leq V_{CC} < 4.5V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output					
Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_C			
Interrupt Input Low Time		t_C			
Timer Input High Time		t_C			
Timer Input Low Time		t_C			
Reset Pulse Width		1.0			μs

Note 6: Parameter sampled but not 100% tested.

AC Electrical Characteristics in ROMless Mode $-40°C < T_A < 85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$		2	DC	μs
(Div-by 20)	$2.5V \leq V_{CC} < 4.5V$		5	DC	μs
Normal Mode	$V_{CC} \geq 4.5V$		4	DC	μs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$		10	DC	μs
R/C Oscillator Mode	$V_{CC} \geq 4.5V$		6	DC	μs
	$2.5V \leq V_{CC} < 4.5V$		15	DC	μs
CKI Clock Duty Clock	fr = Max (÷20 Mode)	40		60	%
Rise Time	fr = 10 MHz Ext Clock		24		ns
Fall Time	fr = 10 MHz Ext Clock		16		ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$		400		ns
	$2.5V \leq V_{CC} < 4.5V$		800		ns
t_{HOLD}	$V_{CC} \geq 4.5V$		120		ns
	$2.5V \leq V_{CC} < 4.5V$		300		ns
Output Propagation Delay	$C_L = 100$ pF, $R_L = 2.2$ kΩ				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$		1.4		μs
	$2.5V \leq V_{CC} < 4.5V$		3.5		μs
All Others	$V_{CC} \geq 4.5V$		2		μs
	$2.5V \leq V_{CC} < 4.5V$		5		μs
Minimum Pulse Width					
Interrupt Input		t_C			
Timer Input		t_C			
Reset Pulse Width		1.0			μs

COP620C/COP621C/COP622C/COP640C/COP641C/COP642C

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	40 mA
Total Current out of GND Pin (Sink)	48 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-55°C \le T_A \le +125°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			$0.1\ V_{CC}$	V
Supply Current					
High Speed Mode, CKI = 18 MHz	$V_{CC} = 5.5V$, tc = 1.1 μs			15	mA
Normal Mode, CKI = 4.5 MHz	$V_{CC} = 5.5V$, tc = 2.2 μs			5	mA
(Note 2)					
HALT Current	$V_{CC} = 5.5V$, CKI = 0 MHz		<10	30	μA
(Note 3)					
Input Levels					
$\overline{\text{RESET}}$, CKI					
Logic High		$0.9\ V_{CC}$			V
Logic Low				$0.1\ V_{CC}$	V
All Other Inputs					
Logic High		$0.7\ V_{CC}$			V
Logic Low				$0.2\ V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	−5		+5	μA
Input Pullup Current	$V_{CC} = 4.5V$	35		300	μA
G Port Input Hysteresis			$0.05\ V_{CC}$		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V$, $V_{OH} = 3.8V$	0.35			mA
Sink	$V_{CC} = 4.5V$, $V_{OL} = 1.0V$	9			mA
All Others					
Source (Weak Pull-Up)	$V_{CC} = 4.5V$, $V_{OH} = 3.2V$	9		120	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OH} = 3.8V$	0.35			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.4			mA
TRI-STATE Leakage		−5.0		+5.0	μA
Allowable Sink/Source					
Current Per Pin					
D Outputs (Sink)				12	mA
All Others				2.5	mA
Maximum Input Current (Room Temp)					
Without Latchup (Note 5)	Room Temp			±100	mA
RAM Retention Voltage, Vr	500 ns Rise and				
	Fall Time (Min)	2.5			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Human body mode, 100 pF through 1500Ω.

Note 5: Except pins G6, G7, $\overline{\text{RESET}}$
pins G6, $\overline{\text{RESET}}$: +60 mA, − 100 mA
pin G7: +100, −25 mA
Sampled but not 100% tested.

COP620C/COP621C/COP622C/COP640C/COP641C/COP642C

AC Electrical Characteristics $-55°C < T_A < +125°C$ unless otherwise specified

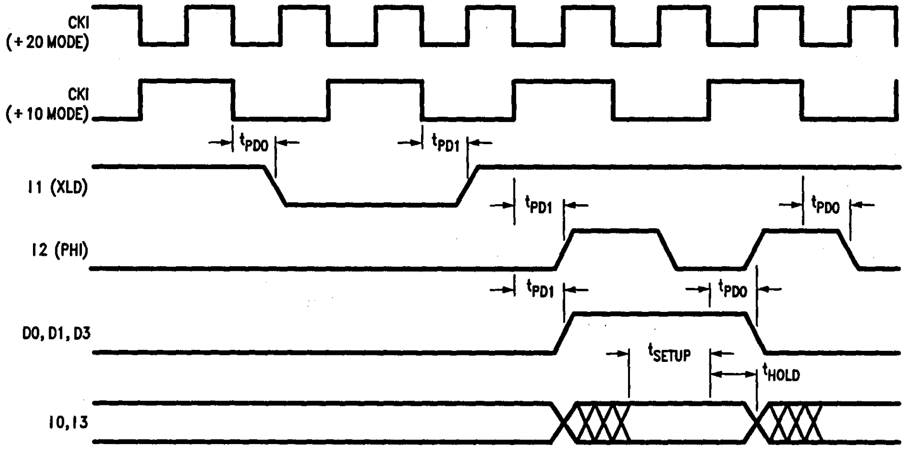
Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$	1.1		DC	µs
(Div-by 20)					
Normal Mode	$V_{CC} \geq 4.5V$	2.2		DC	µs
(Div-by 10)					
CKI Clock Duty Cycle	fr = Max (÷20 Mode)	33		66	%
(Note 6)					
Rise Time (Note 6)	fr = 18 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 18 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	220			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	66			ns
Output Propagation Delay	$R_L = 2.2k, C_L = 100$ pF				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$			0.8	µs
All Others	$V_{CC} \geq 4.5V$			1.1	µs
MICROWIRE Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Valid Time (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		t_C			
Interrupt Input Low Time		t_C			
Timer Input High Time		t_C			
Timer Input Low Time		t_C			
Reset Pulse Width		1			µs

Note 6: Parameter sampled but not 100% tested.

AC Electrical Characteristics in ROMless Mode $-55°C < T_A < +125°C$ unless otherwise specified

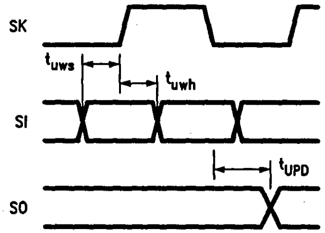
Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$		2.2	DC	µs
(Div-by 20)					
Normal Mode	$V_{CC} \geq 4.5V$		4.4	DC	µs
(Div-by 10)					
CKI Clock Duty Clock	fr = Max (÷20 Mode)	40		60	%
Rise Time	fr = 9 MHz Ext Clock		24		ns
Fall Time	fr = 9 MHz Ext Clock		16		ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$		440		ns
t_{HOLD}	$V_{CC} \geq 4.5V$		132		ns
Output Propagation Delay	$R_L = 2.2k, C_L = 100$ pF				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$		1.55		µs
All Others	$V_{CC} \geq 4.5V$		2.2		µs
Minimum Pulse Width					
Interrupt Input		t_C			
Timer Input		t_C			
Reset Pulse Width		1			µs

Timing Diagrams



TL/DD/9103-2

FIGURE 2a. AC Timing Diagrams in ROMless Mode



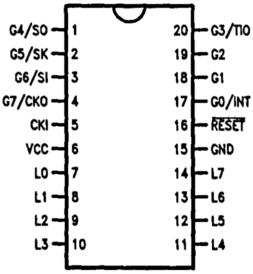
TL/DD/9103-19

FIGURE 2b. MICROWIRE/PLUS Timing

Connection Diagrams

DUAL-IN-LINE PACKAGE

20 DIP

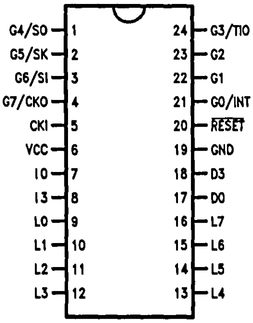


TL/DD/9103-3

Top View

Order Number COP822C-XXX/D, COP822C-XXX/N, COP842C-XXX/D or COP842C-XXX/N
See NS Package Number D20A or N20A

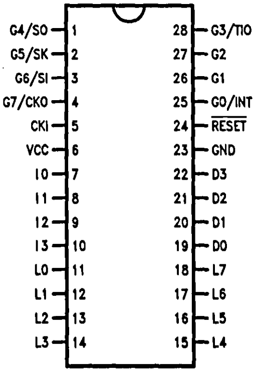
24 DIP



TL/DD/9103-4

Order Number COP821C-XXX/D, COP821C-XXX/N, COP841C-XXX/D or COP841C-XXX/N
See NS Package Number D24C or N24A

28 DIP

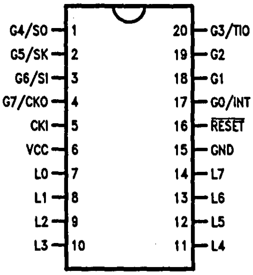


TL/DD/9103-5

Order Number COP820C-XXX/D, COP820C-XXX/N, COP840C-XXX/D or COP840C-XXX/N
See NS Package Number D28C or N28B

SURFACE MOUNT

20 SO Wide

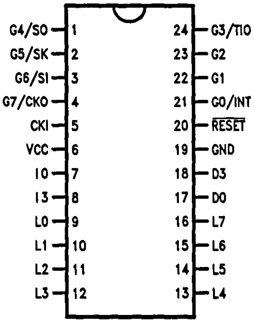


TL/DD/9103-3

Top View

Order Number COP822C-XXX/WM or COP842C-XXX/WM
See NS Package Number M20B

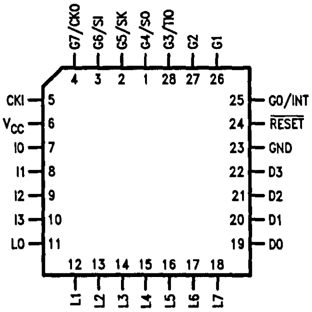
24 SO Wide



TL/DD/9103-4

Order Number COP821C-XXX/WM or COP841C-XXX/WM
See NS Package Number M24B

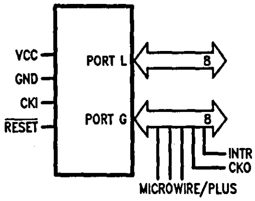
28 PLCC



TL/DD/9103-18

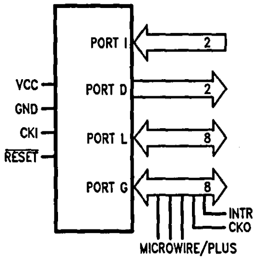
Order Number COP820C-XXX/V or COP840C-XXX/V
See NS Package Number V28A

COP822C
COP842C



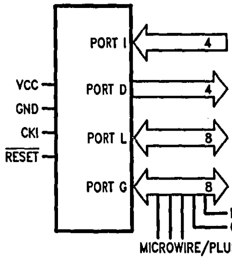
TL/DD/9103-6

COP821C
COP841C



TL/DD/9103-7

COP820C
COP840C



TL/DD/9103-8

FIGURE 3

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

RESET is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below.

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when RESET goes low.

The D2 pin is sampled at reset. If it is held low at reset the COP820C/COP840C enters the ROMless mode of operation.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 15-bit Program Counter register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the stack in RAM during subroutine calls and returns.

PROGRAM MEMORY

Program memory for the COP820C consists of 1024 bytes of ROM (2048 bytes of ROM for the COP840C). These bytes may hold program instructions or constant data. The program memory is addressed by the 15-bit program counter (PC). ROM can be indirectly read by the LAID instruction for table lookup.

DATA MEMORY

The data memory address space includes on chip RAM, I/O and registers. Data memory is addressed directly by the instruction or indirectly by the B, X and SP registers.

The COP820C has 64 bytes of RAM and the COP840C has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" that can be loaded immediately, decremented or tested. Three specific registers: B, X and SP are mapped into this space, the other bytes are available for general usage.

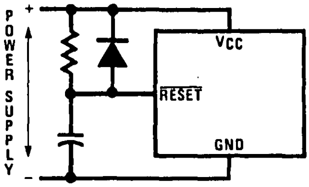
The instruction set permits any bit in memory to be set, reset or tested. All I/O and registers (except the A & PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested.

RESET

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the ports L and G are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L & G are cleared.

The external RC network shown in Figure 4 should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes.

Functional Description (Continued)



TL/DD/9103-9

RC ≥ 5X Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

OSCILLATOR CIRCUITS

Figure 5 shows the three clock oscillator configurations available for the COP820C and COP840C.

A. CRYSTAL OSCILLATOR

The COP820C/COP840C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

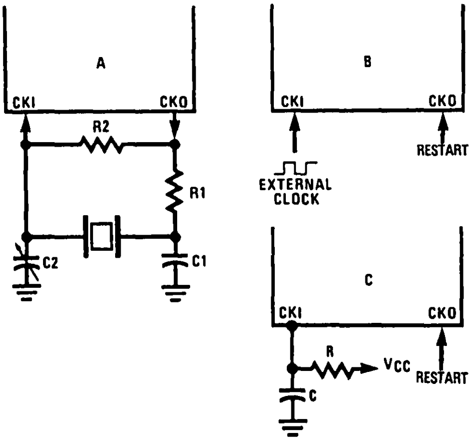
B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9103-10

FIGURE 5. Crystal and R-C Connection Diagrams

OSCILLATOR MASK OPTIONS

The COP820C and COP840C can be driven by clock inputs between DC and 20 MHz. For low input clock frequencies (≤ 5 MHz) the instruction cycle frequency can be selected to be the input clock frequency divided by 10. This mode is known as the Normal Mode.

For oscillator frequencies that are greater than 5 MHz the chip must run with a divide by 20. This is known as the High Speed mode.

TABLE I. Crystal Oscillator Configuration, $T_A = 25°C$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30-36	20	$V_{CC} = 5V$
0	1	30	30-36	10	$V_{CC} = 5V$
0	1	30	30-36	4 (÷ 20)	$V_{CC} = 2.5V$
0	1	200	100-150	0.455	$V_{CC} = 2.5V$

TABLE II. RC Oscillator Configuration, $T_A = 25°C$

R (kΩ)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8 to 2.2	3.6 to 4.5	$V_{CC} = 5V$
5.6	100	1.5 to 1.1	6.7 to 9	$V_{CC} = 5V$
6.8	100	1.1 to 0.8	9 to 12.5	$V_{CC} = 2.5V$

Functional Description (Continued)

The COP820C and COP840C microcontrollers have five mask options for configuring the clock input. The CKI and CKO pins are automatically configured upon selecting a particular option.

- High Speed Crystal (CKI/20) CKO for crystal configuration
- Normal Mode Crystal (CKI/10) CKO for crystal configuration
- High Speed External (CKI/20) CKO available as G7 input
- Normal Mode External (CKI/10) CKO available as G7 input
- R/C (CKI/10) CKO available as G7 input

G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

1) Oscillator operating mode—I1

2) Internal switching current—I2

3) Internal leakage current—I3

4) Output source current—I4

5) DC current caused by external input not at V_{CC} or GND—I5

Thus the total current drain, It is given as

$$It = I1 + I2 + I3 + I4 + I5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external squarewave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$I2 = C \times V \times f$

Where

C = equivalent capacitance of the chip.

V = operating voltage

f = CKI frequency

Some sample current drain values at V_{CC} = 6V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
20	1	9
3.58	3	2.2
2	5	1.2
0.3	33	0.2
0 (HALT)	—	<0.0001

HALT MODE

The COP820C and COP840C support a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to Vr (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the $\overline{\text{RESET}}$ or by the CKO pin. A low on the $\overline{\text{RESET}}$ line reinitializes the microcontroller and starts executing from the address 0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction. This also resets the G7 data bit.

INTERRUPTS

The COP820C and COP840C have a sophisticated interrupt structure to allow easy interface to the real word. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control)

A maskable interrupt on timer carry or timer capture

A non-maskable software/error interrupt on opcode zero

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

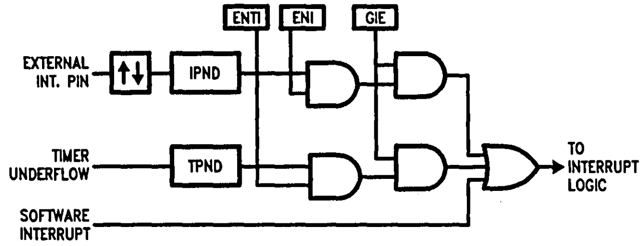
The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

Functional Description (Continued)



TL/DD/9103-11

FIGURE 6. Interrupt Block Diagram

DETECTION OF ILLEGAL CONDITIONS

The COP820C and COP840C incorporate a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP820C and COP840C grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the COP820C and COP840C to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, EEPROMS, etc.) and with other microcontrollers which support the MICROWIRE/PLUS interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 7* shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS interface with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS interface with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	$2t_C$
0	1	$4t_C$
1	x	$8t_C$

where,

t_C is the instruction cycle clock.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP820C and COP840C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 8* shows how two COP820C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP820C. The MICROWIRE/PLUS Master always initiates all data exchanges. (See *Figure 8*). The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summarizes the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by appropriately setting up the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See *Figure 8*.)

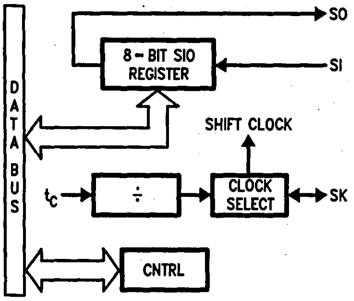
Functional Description (Continued)

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE Master
1	0	SO	Ext. SK	SI	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE Slave

TIMER/COUNTER

The COP820C and COP840C have a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table V details various timer operating modes and their requisite control settings.



TL/DD/9103-12

FIGURE 7. MICROWIRE/PLUS Block Diagram

MODE 1. TIMER WITH AUTO-LOAD REGISTER

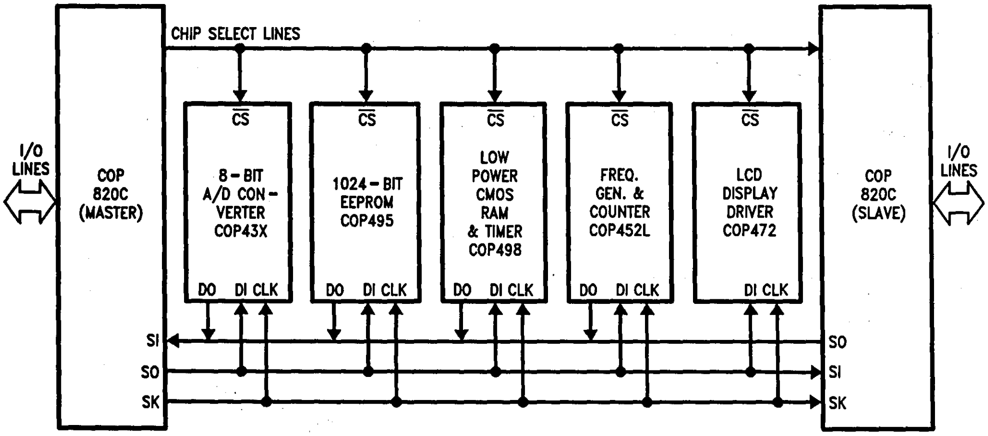
In this mode of operation, the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allow the generation of square-wave outputs or pulse width modulated outputs under software control. (See *Figure 9*)

MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See *Figure 9*)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See *Figure 10.*)



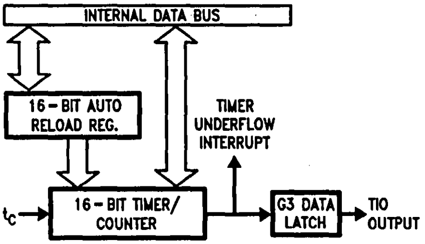
TL/DD/9103-13

FIGURE 8. MICROWIRE/PLUS Application

Functional Description (Continued)

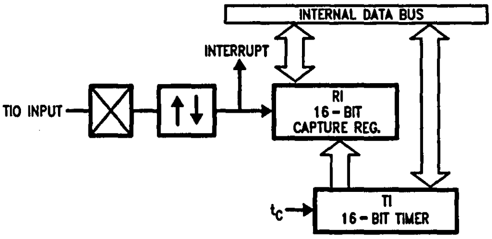
TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t_C
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t_C
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t_C
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t_C



TL/DD/9103-15

FIGURE 9. Timer/Counter Auto Reload Mode Block Diagram

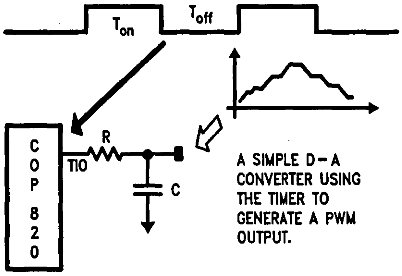


TL/DD/9103-14

FIGURE 10. Timer Capture Mode Block Diagram

TIMER PWM APPLICATION

Figure 11 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.


T_{on}
T_{off}
COP820
TIO
R
C
A SIMPLE D-A CONVERTER USING THE TIMER TO GENERATE A PWM OUTPUT.

TL/DD/9103-16

FIGURE 11. Timer Application

Control Registers

CNTRL REGISTER (ADDRESS X'00EE)

The Timer and MICROWIRE/PLUS control register contains the following bits:

S1 & S0	Select the MICROWIRE/PLUS clock divide-by
IEDG	External interrupt edge polarity select (0 = rising edge, 1 = falling edge)
MSEL	Enable MICROWIRE/PLUS functions SO and SK
TRUN	Start/Stop the Timer/Counter (1 = run, 0 = stop)
TC3	Timer input edge polarity select (0 = rising edge, 1 = falling edge)
TC2	Selects the capture mode
TC1	Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0
BIT 7							BIT 0

PSW REGISTER (ADDRESS X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable
ENI	External interrupt enable
BUSY	MICROWIRE/PLUS busy shifting
IPND	External interrupt pending
ENTI	Timer interrupt enable
TPND	Timer interrupt pending
C	Carry Flag
HC	Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE
Bit 7							Bit 0

Operating Modes

These controllers have two operating modes: Single Chip mode and the ROMless mode. The operating mode is determined by the state of the D2 pin at power on reset.

SINGLE CHIP MODE

In the Single Chip mode, the controller functions as a self contained microcontroller. It can address internal RAM and ROM. All ports configured as memory mapped I/O ports.

ROMLESS MODE

The COP820C and COP840C enter the ROMless mode of operation if the D2 pin is held at logical "0" at reset. In this case the internal ROM is disabled and the controller can now address up to 32 kbytes of external program memory. In the ROMless mode of operation, the COP820C uses the 64 bytes of onboard RAM and the COP840C uses the 128 bytes of onboard RAM. The ports D and I are used to access the external program memory. By providing a serial interface to external program memory, a large address space can be managed without the penalty of losing a large number of I/O pins in the process. *Figure 12* shows in schematic form the logic required for the ROMless mode operation and all support logic required to recreate the I/O.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
COP820C	
00 to 2F	On Chip RAM Bytes
30 to 7F	Unused RAM Address Space (Reads as all Ones)
COP840C	
00 to 6F	On Chip RAM Bytes
70 to 7F	Unused RAM Address Space (Reads as all Ones)
COP820C and COP840C	
80 to BF	Expansion Space for on Chip EERAM
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8–DB	Reserved for Port C
DC	Port D Data Register
DD–DF	Reserved for Port D
E0 to EF	On Chip Functions and Registers
E0–E7	Reserved for Future Parts
E8	Reserved
E9	MICROWIRE/PLUS Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for COP820C and COP840C. The operand is the memory addressed by the B register or X register.

DIRECT

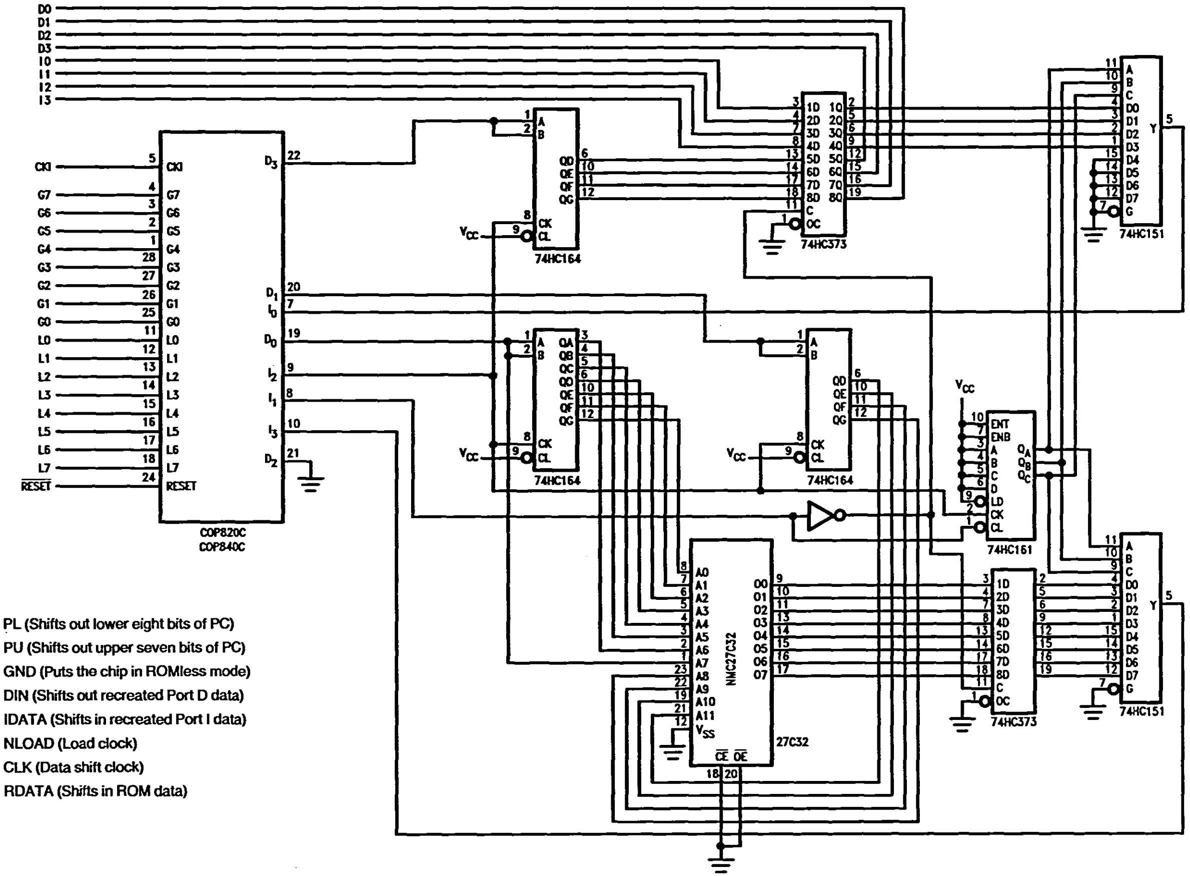
The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.



D_0 PL (Shifts out lower eight bits of PC)

D_1 PU (Shifts out upper seven bits of PC)

D_2 GND (Puts the chip in ROMless mode)

D_3 DIN (Shifts out recreated Port D data)

I_0 IDATA (Shifts in recreated Port I data)

I_1 NLOAD (Load clock)

I_2 CLK (Data shift clock)

I_3 RDATA (Shifts in ROM data)

TL/DD/9103-17

FIGURE 12. COP820C and COP840C ROMless Mode Schematic

Addressing Modes (Continued)

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from −31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
Meml	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
←→	Exchanged with

Instruction Set

ADD	add	A ← A + Meml

ADC	add with carry	A ← A + Meml + C, C ← Carry
		HC ← Half Carry

SUBC	subtract with carry	A ← A + $\overline{\text{Meml}}$ +C, C ← Carry
		HC ← Half Carry

AND	Logical AND	A ← A and Meml
OR	Logical OR	A ← A or Meml
XOR	Logical Exclusive-OR	A ← A xor Meml
IFEQ	IF equal	Compare A and Meml, Do next if A = Meml
IFGT	IF greater than	Compare A and Meml, Do next if A > Meml
IFBNE	IF B not equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Decrement Reg. ,skip if zero	Reg ← Reg − 1, skip if Reg goes to 0
SBIT	Set bit	1 to bit, Mem (bit= 0 to 7 immediate)
RBIT	Reset bit	0 to bit, Mem
IFBIT	If bit	If bit, Mem is true, do next instr.
X	Exchange A with memory	A ←→ Mem
LD A	Load A with memory	A ← Meml
LD mem	Load Direct memory Immed.	Mem ← Imm
LD Reg	Load Register memory Immed.	Reg ← Imm
X	Exchange A with memory [B]	A ←→ [B] (B ← B±1)
X	Exchange A with memory [X]	A ←→ [X] (X ← X±1)
LD A	Load A with memory [B]	A ← [B] (B ← B±1)
LD A	Load A with memory [X]	A ← [X] (X ← X±1)
LD M	Load Memory Immediate	[B] ← Imm (B ← B±1)
CLRA	Clear A	A ← 0
INCA	Increment A	A ← A + 1
DECA	Decrement A	A ← A − 1
LAID	Load A indirect from ROM	A ← ROM(PU,A)
DCORA	DECIMAL CORRECT A	A ← BCD correction (follows ADC, SUBC)
RRCA	ROTATE A RIGHT THRU C	C → A7 → ... → A0 → C
SWAPA	Swap nibbles of A	A7...A4 ←→ A3...A0
SC	Set C	C ← 1, HC ← 1
RC	Reset C	C ← 0, HC ← 0
IFC	If C	If C is true, do next instruction
IFNC	If not C	If C is not true, do next instruction
JMPL	Jump absolute long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Jump absolute	PC11..0 ← i (i = 12 bits)
JP	Jump relative short	PC ← PC + r (r is −31 to +32, not 1)
JSRL	Jump subroutine long	[SP] ← PL,[SP-1] ← PU,SP-2,PC ← ii
JSR	Jump subroutine	[SP] ← PL,[SP-1] ← PU,SP-2,PC11.. 0 ← i
JID	Jump indirect	PL ← ROM(PU,A)
RET	Return from subroutine	SP+2,PL ← [SP],PU ← [SP-1]
RETSK	Return and Skip	SP+2,PL ← [SP],PU ← [SP-1],Skip next instruction
RETI	Return from Interrupt	SP+2,PL ← [SP],PU ← [SP-1],GIE ← 1
INTR	Generate an interrupt	[SP] ← PL,[SP−1] ← PU,SP-2,PC ← 0FF
NOP	No operation	PC ← PC + 1

OPCODE LIST

Bits 7–4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	Bits 3–0
JP -15	JP -31	LD 0F0,#i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0,[B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1,#i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A,[B]	IFBIT 1,[B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2,#i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A,[B]	IFBIT 2,[B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3,#i	DRSZ 0F3	X A, [X−]	X A, [B−]	IFGT A, #i	IFGT A,[B]	IFBIT 3,[B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4,#i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5,#i	DRSZ 0F5	*	JID	AND A, #i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6,#i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A,[B]	IFBIT 6,[B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7,#i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	IFBIT 7,[B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8,#i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9,#i	DRSZ 0F9	*	*	*	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA,#i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+],#i	INCA	SBIT 2,[B]	RBIT 2,[B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB,#i	DRSZ 0FB	LD A, [X−]	LD A, [B−]	LD [B−],#i	DECA	SBIT 3,[B]	RBIT 3,[B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC,#i	DRSZ 0FC	LD Md, #i	JMPL	X A,Md	*	SBIT 4,[B]	RBIT 4,[B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP +13	C
JP -2	JP -18	LD 0FD,#i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5,[B]	RBIT 5,[B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP +14	D
JP -1	JP -17	LD 0FE,#i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00−0EFF	JMP 0E00−0EFF	JP + 31	JP +15	E
JP -0	JP -16	LD 0FF,#1	DRSZ 0FF	*	*	*	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time (1 μs at 20 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

BYTES and CYCLES per INSTRUCTION

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 20 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B,Imm				1/1			(If B < 16)
LD B,Imm				2/3			(If B > 15)
LD Mem,Imm	2/2		3/3		2/2		
LD Reg,Imm				2/3			

* => Memory location addressed by B or X or directly.

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPS™ and the HPC™ family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

Single Chip Emulator Device

The COP820C is fully supported by a form, fit and function emulator device, the COP8720C.

Option List

The COP820C/COP840C mask programmable options are listed out below. The options are programmed at the same time as the ROM pattern to provide the user with hardware flexibility to use a variety of oscillator configuration.

OPTION 1: CKI INPUT

= 1 Normal Mode Crystal	(CKI/10) CKO for crystal configuration
= 2 Normal Mode External	(CKI/10) CKO available as G7 input
= 3 R/C	(CKI/10) CKO available as G7 input
= 4 High Speed Crystal	(CKI/20) CKO for crystal configuration
= 5 High Speed External	(CKI/20) CKO available as G7 input

OPTION 2: COP820C/COP840C BONDING

= 1 28 pin package

= 2 24 pin package

= 3 20 pin package

The following option information is to be sent to National along with the EPROM.

Option Data

Option 1 Value_is: CKI Input

Option 2 Value_is: COP Bonding

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/ COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper is an Electronic Bulletin Board information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down-loaded to disk for later use.

ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:

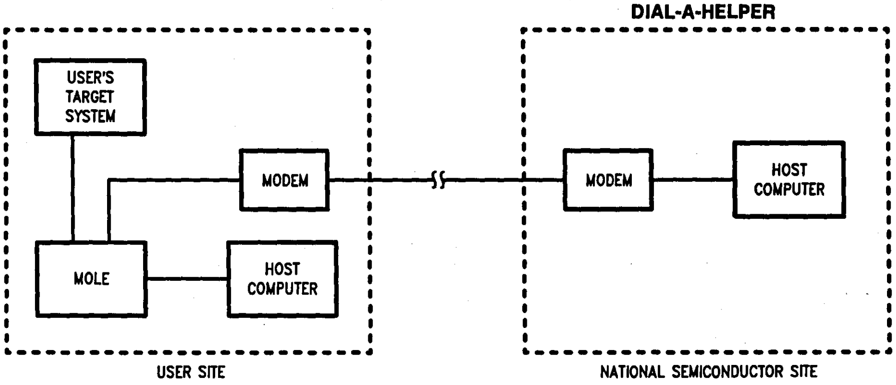
Dial-A-Helper Users Manual

Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 baud	
	Setup:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hrs. 7 Days	



TL/DD/9103-20

National Semiconductor

ADVANCE INFORMATION

COP820CB/COP821CB/COP822CB Single-Chip microCMOS Microcontrollers

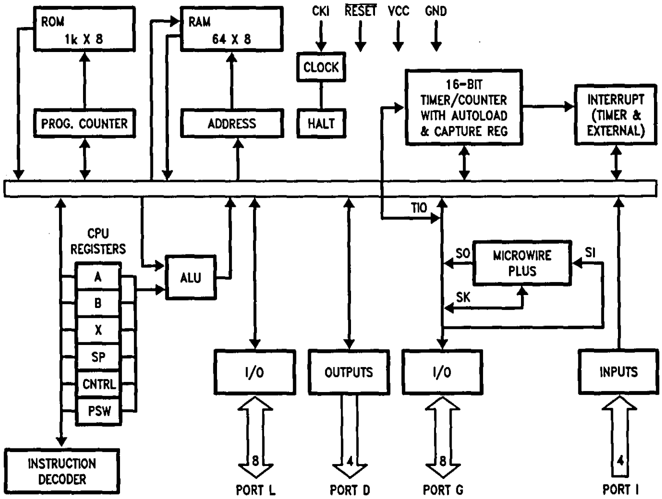
General Description

The COPS™ microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP820CB and COP840CB to the specific application. The part operates over a voltage range of 2.0V to 3.5V. The minimum operating voltage of 2.0V makes this part suitable for applications requiring low power consumption. The part may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain (500 mA at 10 μs instruction rate) Low current static HALT mode (Typically < 1 μA, max 2 μA)
- Single supply operation: 2.0V to 3.5V
- 1024 bytes ROM/64 Bytes RAM
- 16-bit read/write timer operates in a variety of modes
  - — Timer with 16-bit auto reload register
  - — 16-bit external event counter
  - — Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - — Reset master clear
  - — External interrupt with selectable edge
  - — Timer interrupt or capture interrupt
  - — Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE PLUS™ serial I/O
- 28 pin package (optionally 24 or 20 pin package)
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature range: 0°C to +70°C
- ROMless mode for accurate emulation and external program capability—expandable to 32k bytes in ROMless mode
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/10426-1

FIGURE 1

National Semiconductor

ADVANCE INFORMATION

COP8640C/COP8641C/COP8642C/ COP8620C/COP8621C/COP8622C Single-Chip microCMOS Microcontrollers

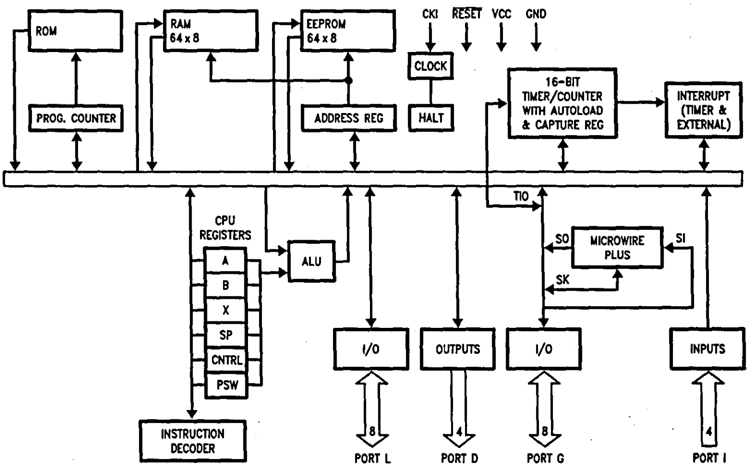
General Description

The COP8640C/COP8620C is a member of the COPS™ microcontroller family. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, EEPROM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP8640C/COP8620C to the specific application. The part operates over a voltage range of 4.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate. The part may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Features

- Low Cost 8-bit microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain (2.2 mA at 3 μs instruction rate) Low current static HALT mode (Typically < 1 μA)
- Single supply operation: 2.5 to 6.0V
- 2048 Bytes ROM/64 Bytes RAM/64 Bytes EEPROM on COP8640C
- 1024 bytes ROM/64 bytes RAM/64 bytes EEPROM on COP8620C
- 16-bit read/write timer operates in a variety of modes
  - — Timer with 16-bit auto reload register
  - — 16-bit external event counter
  - — Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - — Reset master clear
  - — External interrupt with selectable edge
  - — Timer interrupt or capture interrupt
  - — Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instructions
- MICROWIRE PLUS™ serial I/O
- 28 pin package (optionally 24 or 20 pin package)
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature range: −40°C to +85°C
- ROMless mode for accurate emulation and external program capability—expandable to 32k bytes in ROMless mode
- COP8620C Series compatible with COP8720C Series
- Fully supported by National's Development Systems

Block Diagram



TL/DD/10366-1

FIGURE 1

National Semiconductor

PRELIMINARY

COP8720C/COP8721C/COP8722C Single-Chip microCMOS Microcontrollers

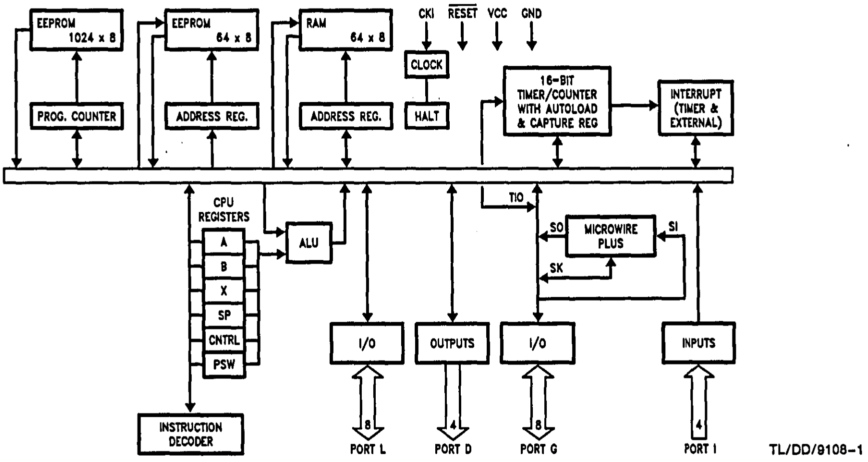
General Description

The COP8720C/COP8721C/COP8722C are members of the COPS™ microcontroller family featuring on-chip EEPROM modules. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. This low cost microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP8720C to the specific application. The part operates over a voltage range of 2.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 microsecond per instruction rate. The COP8720 is totally compatible with the ROM based COP820C microcontroller. It serves as a form, fit and function emulator device for the COP820 microcontroller family.

Features

- Low Cost 8-bit CORE microcontroller
- Fully static CMOS
- 1 µs instruction time (20 MHz clock)
- Low current drain (2.2 mA at 3 µs instruction rate)
  Low current static HALT mode (Typically < 10 µA)
- Single supply operation: 2.5V to 6.0V
- 1024 bytes EEPROM program memory
- 64 bytes of RAM
- 64 bytes EEPROM data memory
- 16-bit read/write timer operates in a variety of modes
  - Timer with 16-bit auto reload register
  - 16-bit external event counter
  - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - Reset master clear
  - External interrupt with selectable edge
  - Timer interrupt or capture interrupt
  - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instruction
- MICROWIRE/PLUS™ serial I/O
- 28 pin package (optionally 24 or 20 pin package)
- 24 input/output pins
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Form, fit and function EEPROM emulation device for COP820C/COP821C/COP822C
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/9108-1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	50 mA
Total Current out of GND Pin (Sink)	60 mA
Storage Temperature Range	$-65°C$ to $+140°C$

Note: Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		2.5		6.0	V
Power Supply Ripple (Note 1)	Peak to Peak			0.1 V_{CC}	V
Operating Voltage during EEPROM Write (Note 7)		4.5		6.0	V
Supply Current (see page 10)					
High Speed Mode, CKI = 20 MHz	V_{CC} = 6V, tc = 1 µs			13	mA
Normal Mode, CKI = 5 MHz	V_{CC} = 6V, tc = 2 µs			7	mA
Normal Mode, CKI = 2 MHz	V_{CC} = 2.5V, tc = 5 µs			2	mA
(Note 2)					
HALT Current	V_{CC} = 6V, CKI = 0 MHz		<10	30	µA
(Note 3)					
Input Levels					
$\overline{\text{RESET}}$, CKI					
Logic High		0.9 V_{CC}			V
Logic Low				0.1 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 6.0V	−2		+2	µA
Input Pullup Current	V_{CC} = 6.0V	40		250	µA
G Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1.0V	10			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	2.0			mA
All Others					
Source (Weak Pull-Up)	V_{CC} = 4.5V, V_{OH} = 3.2V	10		100	µA
	V_{CC} = 2.5V, V_{OH} = 1.8V	2.5		33	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.8V	0.4			mA
	V_{CC} = 2.5V, V_{OH} = 1.8V	0.2			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
	V_{CC} = 2.5V, V_{OL} = 0.4V	0.7			mA
TRI-STATE Leakage		−2.0		+2.0	µA
Allowable Sink/Source Current Per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current (Room Temp) without Latchup (Note 5)				±100	mA
RAM Retention Voltage, Vr	500 ns Rise and Fall Time (Min)	2.0			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $-40°C < T_A < +85°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (tc)					
High Speed Mode	$V_{CC} \geq 4.5V$	1		DC	µs
(Div-by 20)	$2.5V \leq V_{CC} < 4.5V$	2.5		DC	µs
Normal Mode	$V_{CC} \geq 4.5V$	2		DC	µs
(Div-by 10)	$2.5V \leq V_{CC} < 4.5V$	5		DC	µs
R/C Oscillator Mode	$V_{CC} \geq 4.5V$	3		DC	µs
(Div-by 10)					
	$2.5V \leq V_{CC} < 4.5V$	7.5		DC	µs
CKI Clock Duty Cycle (Note 6)	fr = Max (÷20 Mode)	33		66	%
Rise Time (Note 6)	fr = 20 MHz Ext Clock			12	ns
Fall Time (Note 6)	fr = 20 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	$V_{CC} \geq 4.5V$	200			ns
	$2.5V \leq V_{CC} < 4.5V$	500			ns
t_{HOLD}	$V_{CC} \geq 4.5V$	60			ns
	$2.5V \leq V_{CC} < 4.5V$	150			ns
Output Propagation Delay	$R_L = 2.2k, C_L = 100$ pF				
t_{PD1}, t_{PD0}					
SO, SK	$V_{CC} \geq 4.5V$			0.7	µs
	$2.5V \leq V_{CC} < 4.5V$			1.75	µs
All Others	$V_{CC} \geq 4.5V$			1	µs
	$2.5V \leq V_{CC} < 4.5V$			2.5	µs
MICROWIRE™ Setup Time t_{UWS}		20			ns
MICROWIRE Hold Time t_{UWH}		56			ns
MICROWIRE Output Propagation Delay t_{UPD}				220	ns
Input Pulse Width					
Interrupt Input High Time		t_C			
Interrupt Input Low Time		t_C			
Timer Input High Time		t_C			
Timer Input Low Time		t_C			
Reset Pulse Width		1.0			µs

Note 1: Rate of voltage change must be less than 0.5V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports are at TRI-STATE and tied to ground, all outputs low and tied to ground.

Note 4: Human body model, 100 pF through 1500Ω.

Note 5: Except pins G6, G7, $\overline{\text{RESET}}$
pins G6, $\overline{\text{RESET}}$: +60 mA
pin G7: −25 mA

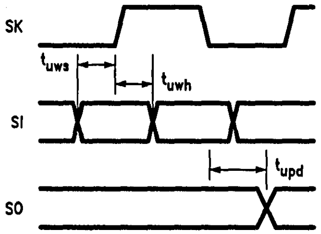
Note 6: Parameter sampled but not 100% tested.

Note 7: The temperature range for write operation is 0°C to 70°C.

EEPROM Characteristics

Parameter	Condition	Min	Typ	Max	Units
EEPROM Write Cycle Time	$4.5V \leq V_{CC} \leq 6.0V$	15	20	25	ms
EEPROM Number of Writes				10000	Cycles
V_{CC} Level for Write Lock Out	V_{LKO}	3.9		4.4	V
Programming Voltage to $\overline{\text{RESET}}$ Pin	V_{prg} $4.5V \leq V_{CC} \leq 6.0V$	11.5	12	12.5	V

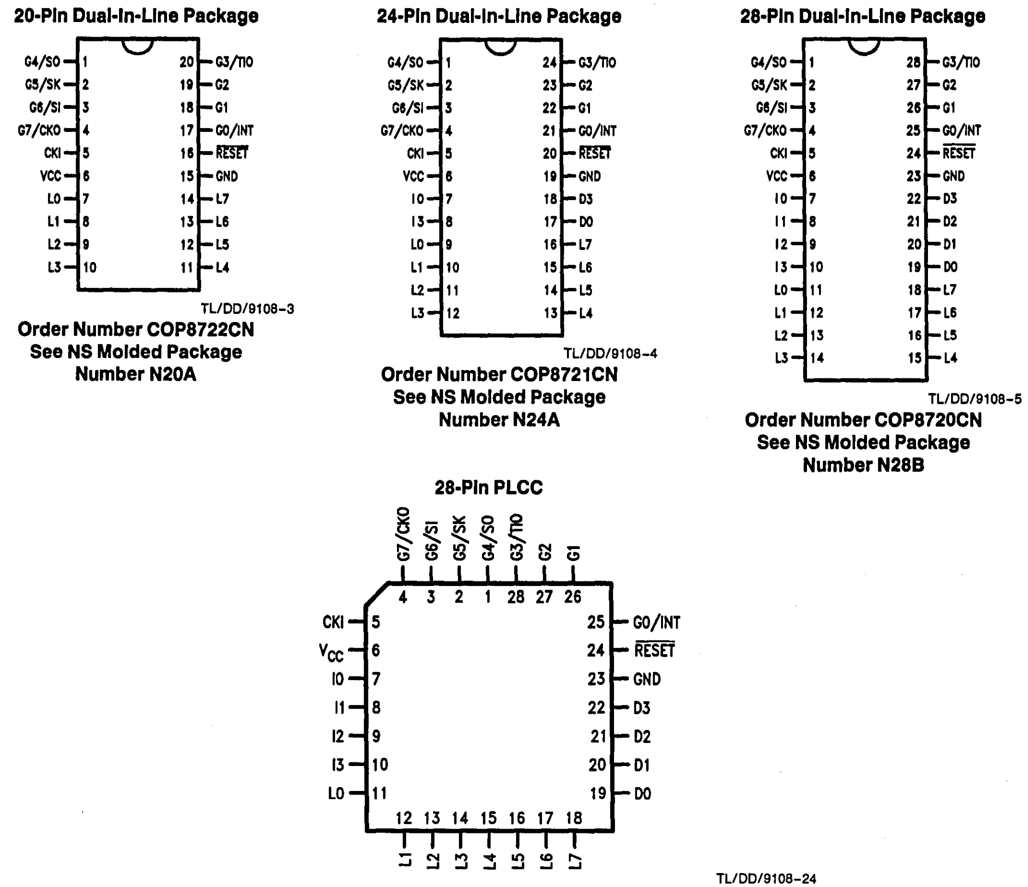
Timing Diagrams



TL/DD/9108-22

FIGURE 2. MICROWIRE/PLUS Timing Diagram

Connection Diagrams



20-Pin Dual-In-Line Package

TL/DD/9108-3

Order Number COP8722CN
See NS Molded Package
Number N20A

24-Pin Dual-In-Line Package

TL/DD/9108-4

Order Number COP8721CN
See NS Molded Package
Number N24A

28-Pin Dual-In-Line Package

TL/DD/9108-5

Order Number COP8720CN
See NS Molded Package
Number N28B

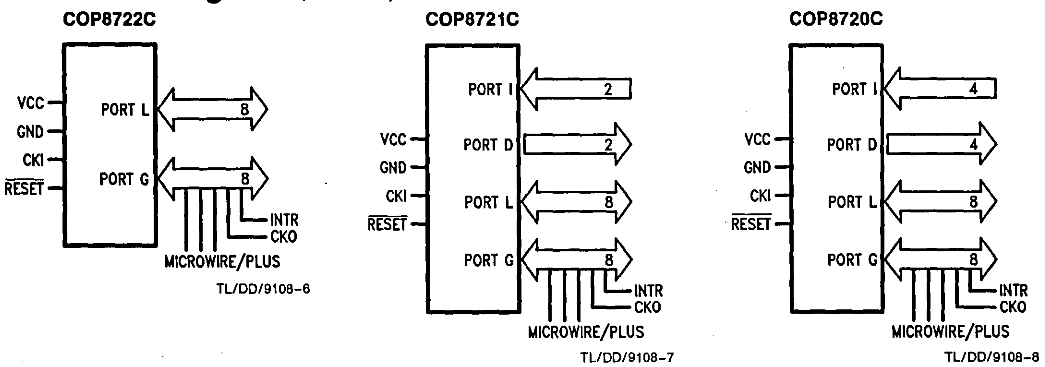
28-Pin PLCC

TL/DD/9108-24

Order Number COP8720CV
See NS PLCC Package
Number V28A

FIGURE 3

Connection Diagrams (Continued)



TL/DD/9108-6

TL/DD/9108-7

TL/DD/9108-8

FIGURE 3 (Continued)

Pin Descriptions

V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an external source, a R/C generated oscillator or a crystal (in conjunction with CKO). See Oscillator description.

$\overline{\text{RESET}}$ is the master reset input. See Reset description.

PORT I is a four bit Hi-Z input port.

PORT L is an 8-bit I/O port.

There are two registers associated with each L I/O port: a data register and a configuration register. Therefore, each L I/O bit can be individually configured under software control as shown below:

Port L Config.	Port L Data	Port L Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins.

PORT G is an 8-bit port with 6 I/O pins (G0–G5) and 2 input pins (G6, G7). All eight G-pins have Schmitt Triggers on the inputs. The G7 pin functions as an input pin under normal operation and as the continue pin to exit the HALT mode. There are two registers with each I/O port: a data register and a configuration register. Therefore, each I/O bit can be individually configured under software control as shown below.

Port G Config.	Port G Data	Port G Setup
0	0	Hi-Z Input (TRI-STATE)
0	1	Input With Weak Pull-Up
1	0	Push-Pull "0" Output
1	1	Push-Pull "1" Output

Three data memory address locations are allocated for these ports, one for data register, one for configuration register and one for the input pins. Since G6 and G7 are input only pins, any attempt by the user to set them up as outputs by writing a one to the configuration register will be disregarded. Reading the G6 and G7 configuration bits will return zeros. Note that the chip will be placed in the HALT mode by setting the G7 data bit.

Six bits of Port G have alternate features:

G0 INTR (an external interrupt)

G3 TIO (timer/counter input/output)

G4 SO (MICROWIRE serial data output)

G5 SK (MICROWIRE clock I/O)

G6 SI (MICROWIRE serial data input)

G7 CKO crystal oscillator output (selected by mask option) or HALT restart input (general purpose input)

Pins G1 and G2 currently do not have any alternate functions.

PORT D is a four bit output port that is set high when $\overline{\text{RESET}}$ goes low.

The D2 pin is sampled at reset. If it is held low at reset the COP8720C enters the ROMless mode of operation.

Functional Description

Figure 1 shows the block diagram of the internal architecture. Data paths are illustrated in simplified form to depict how the various logic elements communicate with each other in implementing the instruction set of the device.

ALU AND CPU REGISTERS

The ALU can do an 8-bit addition, subtraction, logical or shift operation in one cycle time.

There are five CPU registers:

A is the 15-bit Program Counter register

PU is the upper 7 bits of the program counter (PC)

PL is the lower 8 bits of the program counter (PC)

B is the 8-bit address register, can be auto incremented or decremented.

X is the 8-bit alternate address register, can be incremented or decremented.

SP is the 8-bit stack pointer, points to subroutine stack (in RAM).

B, X and SP registers are mapped into the on chip RAM. The B and X registers are used to address the on chip RAM. The SP register is used to address the program counter stack in RAM during subroutine calls and returns.

Functional Description (Continued)

MEMORY

The COP8720C contains 1 Kbyte of Program EEPROM, 64 bytes of on-chip RAM and Registers, I/O, 64 bytes of Data EEPROM and 256 bytes of firmware ROM.

PROGRAM MEMORY

Program memory for the COP8720C consists of two modules—the 1 Kbyte program EEPROM and the 256 byte ROM which contains the firmware routines for reading and programming the EEPROM.

Memory locations in the 1 Kbyte program EEPROM module are accessed by the address register, EEAR, and the data register, EROMDR. The EEAR is mapped into the address locations E2 and E3. The EROMDR register is located at the address E1.

Under normal conditions, the program EEPROM and the ROM are addressed by the PC and their contents go to the instruction bus. During the EEPROM program and verify cycle, the EEPROM is treated as data memory while the COP8720C is executing out of the firmware ROM. The EEPROM is addressed through the EEAR register. The EROMDR register holds the data read back from the EEPROM location during a verify cycle and holds the data to be written into the EEPROM location during a program cycle. The verify cycle takes 1 instruction cycle and the write cycle takes 20 ms.

Accesses to the program EEPROM is controlled by two flags, AEN and PEN, in the control register, EECR.

AEN	PEN	Access Type
0	0	Normal
0	1	Normal
1	0	EEPROM Read Cycle
1	1	EEPROM Write Cycle

To prevent accidental erasures and over-write situations the application program should not set the AEN and PEN flags in the EECR register. The COP8720C supports application accesses to the EEPROM module via two subroutines in the firmware ROM—an EEPROM read and an EEPROM write subroutine. To program an EEPROM memory location, the user loads the EECR and EROMDR registers and invokes the write subroutine at the address 40C0 Hex. To read an EEPROM location the user loads the EEAR register with the address of the EEPROM memory location and invokes the read subroutine at the address 40D4 Hex. The read subroutine returns the contents of the addressed EEPROM location in the EROMDR register.

DATA MEMORY

The data memory for the COP8720C consists of on-chip RAM, EEPROM, I/O and registers. Data memory is accessed directly by the instruction or indirectly by the B, X and SP registers.

RAM

The COP8720C has 64 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" that can be loaded efficiently, decremented and tested. Three specific registers: B, X and SP are mapped into this space, the other bytes are available for general use.

The instruction set of the COP8720C permits any bit in the data memory to be set, reset or tested. All I/O and the registers (except the A and PC) are memory mapped; therefore, I/O bits and register bits in addition to the normal data RAM can be directly and individually set, reset and tested.

DATA EEPROM

The COP8720C provides 64 bytes of EEPROM for nonvolatile data memory. The data EEPROM can be read and programmed in exactly the same way as the RAM. All instructions that perform read and write operations on the RAM work similarly upon the data EEPROM.

A data EEPROM programming cycle is initiated by an instruction such as X, LD, SBIT or RBIT. The EE memory support circuitry sets the BsyERAM flag in the EECR register immediately upon beginning a data EEPROM write cycle. It will be automatically reset by the hardware at the end of the data EEPROM write cycle. The application program should test the BsyERAM flag before attempting a write operation to the data EEPROM. A second EEPROM write operation while a write operation is in progress will be ignored. The Werr flag in the EECR register is set to indicate the error status.

SIGNATURE AND OPTION REGISTERS

The COP8720C provides a set of six additional registers implemented with EEPROM cells—the Signature and Option registers.

The Signature register is a four-byte register provided for storing ROM code rev. numbers or other application specific information. The Signature register is shadowed behind the data EEPROM cells at addresses 8C to 8F Hex. Two test modes are provided to allow the Signature register to be read or programmed.

The Option register consists of two bytes shadowed behind the addresses 89 and 8B Hex. The Option register allows the COP8720C to be programmed to accurately emulate the different mask options available on the COP820C.

—	—	—	—	ROMemu	x	0		89 Hex
—	—	—	—	HS	RC	XTAL	x	8B Hex

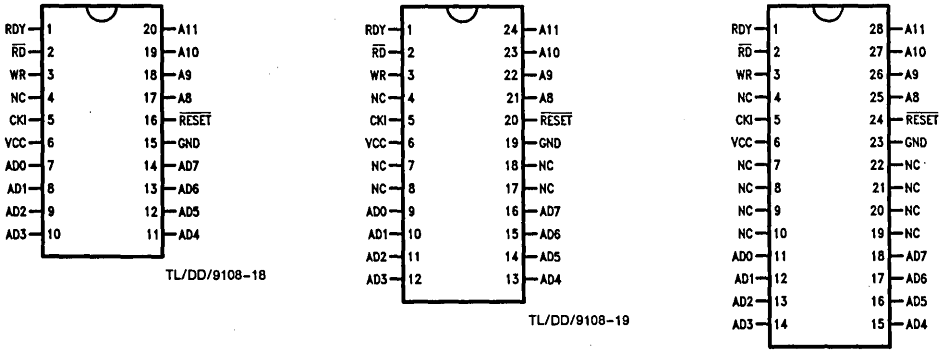
ROMemu: When set, the Data EEPROM and all the EE related registers become inaccessible. Thus, the EE registers look like nonexistent memory locations when addressed by the application program and the Program EEPROM behaves just like ordinary ROM. Thus, setting the ROMemu bit allows the COP8720C to emulate the ROM based COP820C with 100% accuracy.

HS, RC, XTAL: These three bits allow the COP8720C to emulate the clock options of the COP820C. Note that only five out of the possible eight combinations are legal—the combinations 0E, 0C and 06 are illegal combinations.

EECR and EE SUPPORT CIRCUITS

The EEPROM program and data modules share a common set of EE support circuits to generate all necessary high

Functional Description (Continued)



TL/DD/9108-18

TL/DD/9108-19

TL/DD/9108-20

FIGURE 4. Pinouts for the COP8720C in Programming Mode

voltage programming pulses. Each programming cycle consists of a 10 ms erase cycle followed by a 10 ms write cycle for each byte. An EEPROM cell in the erase state is read out as a 0 and the written state is read out as a 1. Since the two EE modules share the support circuitry, programming the two modules at the same time is not allowed.

The EECR register provides control, status and test mode functions for the EE modules.

The EECR register bit assignments are shown below.

EECR Register Bit Assignment

Wr	Test Mode Codes					AEN	PEN	
Rd	Test	Mode	Codes	BsyEROM	BsyERAM	AEN	V_{LKO}	Werr
Bit	7	6	5	4	3	2	1	0

Werr — Write Error. Writing to data EEPROM while a previous write cycle is still busy, that is BsyERAM is not 0, causes Werr to be set to 1 indicate error status. Werr is cleared by writing a 0 into it.

PEN — A program EEPROM programming cycle is started by setting PEN and AEN to 1 at the same time. PEN is "written thru". It is not latched.

V_{LKO} — EECR bit 1 is read as the lock out indicator. A low V_{CC} detector is enabled at the start of the EE programming cycle. If it finds V_{CC} less than V_{LKO}, the V_{LKO} status bit is set and the write cycle is aborted. The V_{LKO} status bit stays latched until the start of another EE programming cycle.

AEN — AEN controls the program EEPROM address/data interface. when AEN is 0, the EEPROM is the program memory. It is adressed by PC, and its output data goes onto the instruction bus. When AEN is set to 1, the EEPROM becomes data memory. It is addressed by the EEAR, and it is accessed from the EROMDR.

BsyERAM — Set to 1 when data EEPROM is being written, is automatically reset by the hardware upon completion of the write operation.

BsyEROM — Set to 1 when program EEPROM is being written, is automatically reset by the hardware upon completion of the write operation.

Bits 3 to 7 of the EECR are used for encoding various EEPROM module test modes, most of which are for factory manufacturing tests. Two of the test modes used for accessing the signature and option registers are described in a previous section. The EE test modes are activated by applying high voltage to the $\overline{\text{RESET}}$ pin. Some of the test modes, if activated improperly, can make the part inoperable. These test modes are reserved for use by the manufacturer only.

The EECR register is cleared by $\overline{\text{RESET}}$. EECR is mapped into address location E0.

When either BsyERAM or BsyEROM is set to 1, that is an EEPROM programming cycle is in progress, the AEN bit is locked up and cannot be changed by the processor.

EXTERNALLY PROGRAMMING THE PROGRAM EEPROM

As shown in the previous section, the COP8720C permits the program EEPROM memory module to be altered under program control via the EECR register. To facilitate ease of development the COP8720C also provides an external mode of loading executable code into the program EEPROM module.

This section describes the programming method for the COP8720C EEPROM.

Programming the COP8720C EEPROM or the special registers is initiated by applying V_{PRG} to the $\overline{\text{RESET}}$ pin. Control gets transferred to the firmware ROM when V_{PRG} is applied to the $\overline{\text{RESET}}$ pin. The program contained in the firmware ROM sets up the I/O of the COP8720C to simulate the I/O requirements of a 2-kbyte memory device. This is done by setting up the COP8720C I/O as eight bits of address/data lines, three address lines, read/write control and a ready signal.

Functional Description (Continued)

Figure 4 shows the three packages and the associated I/O. The pin descriptions are as follows:

V_{CC}	Positive 5V Power Supply
GND	Ground
$\overline{\text{RESET}}$	Active Low Reset Input
CKI	Clock Input
AD0–AD7	Multiplexed Address/Data Lines
A8–A11	Address Lines
$\overline{\text{RD}}$	Active Low Read Strobe
WR	Active High Write Strobe
RDY	Active High Ready Output

The firmware ROM program allows the user to reference the special registers as EEPROM memory locations in the address range 2048–2070 decimal. The following mapping is used:

Signature Register #1 at EEPROM address 800 Hex
Signature Register #2 at EEPROM address 801 Hex
Signature Register #3 at EEPROM address 802 Hex
Signature Register #4 at EEPROM address 803 Hex

Option Register #1 at EEPROM address 804 Hex
Option Register #2 at EEPROM address 805 Hex

Note that in order to reference these registers the user must come in with addresses in the range 800 Hex to 805 Hex.

PROGRAMMING STEPS

The programmig host has to go through the following steps for the write and verify cycles. (See *Figure 2*)

WRITE:

1. Power is applied with the $\overline{\text{RESET}}$ and WR pins low and the $\overline{\text{RD}}$ high.
2. $\overline{\text{RESET}}$ is then brought up to V_{prg} within 1 μs.
3. The lower byte of the address to be written into is applied to the pins AD0–AD7 and the upper 3 bits of the address applied to the pins A8–A11.
4. Observing the setup times, WR is brought high.
5. The data to be programmed is applied to the pins AD0–AD7.
6. The RDY signal from the COP8720C goes low. This indicates that the WR and data on AD0–AD7 have been accepted and these inputs can be removed.
7. The programming host must now either wait for the RDY signal to go high or wait at least 20 ms before initiating a new programming cycle.

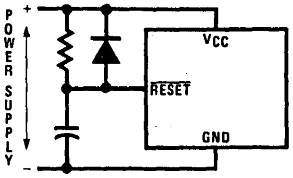
VERIFY:

1. Power is applied with $\overline{\text{RESET}}$ and WR pins held low and the $\overline{\text{RD}}$ high.
2. The $\overline{\text{RESET}}$ pin is brought up to V_{prg} within 1 μs.
3. The lower byte of the address to be read is applied to the pins AD0–AD7 and the upper three bits to the pins AD8–AD11.
4. Observing setup times the $\overline{\text{RD}}$ pin is brought low.
5. After a time T7, the RDY signal from the COP8720C goes low and data is ready for the host on the pins AD0–AD7. The data stays until the $\overline{\text{RD}}$ signal goes back high after which the RDY signal will go back high.
6. The host must wait for the RDY signal to go back high before the next read cycle is initiated.

RESET

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the ports L and G are placed in the TRI-STATE mode and the Port D is set high. The PC, PSW and CNTRL registers are cleared. The data and configuration registers for Ports L & G are cleared.

The external RC network shown in *Figure 5* should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.



TL/DD/9108-9

RC ≥ 5X Power Supply Rise Time

FIGURE 5. Recommended Reset Circuit

OSCILLATOR CIRCUITS

Figure 6 shows the three clock oscillator configurations available for the COP8720C.

A. CRYSTAL OSCILLATOR

The COP8720C can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table I shows the component values required for various standard crystal values.

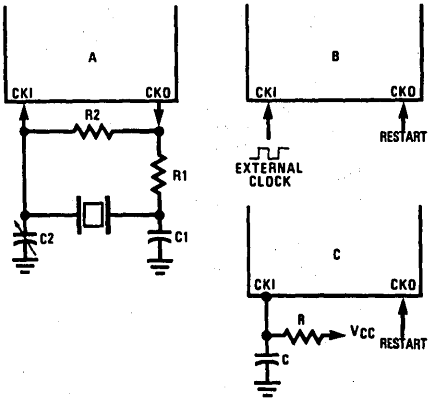
B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. R/C OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT restart control.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/9108-10

FIGURE 6. Crystal and R-C Connection Diagrams

OSCILLATOR OPTIONS

The COP8720C can be driven by clock inputs between DC and 20 MHz. For low input clock frequencies (≤ 5 MHz) the instruction cycle frequency can be selected to be the input clock frequency divided by 10. This mode is known as the Normal Mode.

Functional Description (Continued)

TABLE I. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	20	V_{CC} = 5V
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4 (÷ 20)	V_{CC} = 2.5V
0	1	200	100–150	0.455	V_{CC} = 2.5V

TABLE II. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8–2.2	3.6–4.5	V_{CC} = 5V
5.6	100	1.5–1.1	6.7–9	V_{CC} = 5V
6.8	100	1.1–0.8	9–12.5	V_{CC} = 2.5V

For oscillator frequencies that are greater than 5 MHz the chip must run with a divide by 20. This is known as the High Speed mode.

The COP820C microcontroller has five mask options for configuring the clock input. To emulate these mask options 3 bits must be set in the Option register.

HS	RC	XTAL	Mask Option
1	0	1	High Speed Crystal
0	0	1	Normal Mode Crystal
1	0	0	High Speed External
0	0	0	Normal Mode External
0	1	0	R/C Oscillator

The CKI and CKO pins are automatically configured upon selecting a particular option.

— High Speed Crystal (CKI/20) CKO for crystal configuration

— Normal Mode Crystal (CKI/10) CKO for crystal configuration

— High Speed External (CKI/20) CKO available as G7 input

— Normal Mode External (CKI/10) CKO available as G7 input

— R/C (CKI/10) CKO available as G7 input

Where, G7 can be used either as a general purpose input or as a control input to continue from the HALT mode.

CURRENT DRAIN

The total current drain of the chip depends on:

1) Oscillator operating mode—I1

2) Internal switching current—I2

3) Internal leakage current—I3

4) Output source current—I4

5) DC current caused by external input not at V_{CC} or GND—I5

Thus the total current drain, It is given as

$$It = I1 + I2 + I3 + I4 + I5$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw the least current when in the normal mode. The high speed mode will draw additional current. The R/C mode will draw the most. Operating with a crystal network will draw more current than an external squarewave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

Where

C = equivalent capacitance of the chip. (TBD)

V = operating voltage

f = CKI frequency

The typical capacitance for the COP820C is TBD pF.

Some sample current drain values at V_{CC} = 6V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
20	1	13
3.58	3	2.2
2	5	1.2
0.3	33	0.2
0 (HALT)	—	<0.01

HALT MODE

The COP8720C supports a power saving mode of operation: HALT. The controller is placed in the HALT mode by setting the G7 data bit, alternatively the user can stop the clock input. In the HALT mode all internal processor activities including the clock oscillator are stopped. The fully static architecture freezes the state of the controller and retains all information until continuing. In the HALT mode, power requirements are minimal as it draws only leakage currents and output current. The applied voltage (V_{CC}) may be decreased down to Vr (minimum RAM retention voltage) without altering the state of the machine.

There are two ways to exit the HALT mode: via the $\overline{\text{RESET}}$ or by the CKO pin. A low on the $\overline{\text{RESET}}$ line reinitializes the

Functional Description (Continued)

microcontroller and starts executing from the address 0000H. A low to high transition on the CKO pin causes the microcontroller to continue with no reinitialization from the address following the HALT instruction. This also resets the G7 data bit.

INTERRUPTS

The COP8720C has a sophisticated interrupt structure to allow easy interface to the real world. There are three possible interrupt sources, as shown below.

A maskable interrupt on external G0 input (positive or negative edge sensitive under software control).

A maskable interrupt on timer carry or timer capture.

A non-maskable software/error interrupt on opcode zero.

INTERRUPT CONTROL

The GIE (global interrupt enable) bit enables the interrupt function. This is used in conjunction with ENI and ENTI to select one or both of the interrupt sources. This bit is reset when interrupt is acknowledged.

ENI and ENTI bits select external and timer interrupt respectively. Thus the user can select either or both sources to interrupt the microcontroller when GIE is enabled.

IEDG selects the external interrupt edge (0 = rising edge, 1 = falling edge). The user can get an interrupt on both rising and falling edges by toggling the state of IEDG bit after each interrupt.

IPND and TPND bits signal which interrupt is pending. After interrupt is acknowledged, the user can check these two bits to determine which interrupt is pending. This permits the interrupts to be prioritized under software. The pending flags have to be cleared by the user. Setting the GIE bit high inside the interrupt subroutine allows nested interrupts.

The software interrupt does not reset the GIE bit. This means that the controller can be interrupted by other interrupt sources while servicing the software interrupt.

INTERRUPT PROCESSING

The interrupt, once acknowledged, pushes the program counter (PC) onto the stack and the stack pointer (SP) is decremented twice. The Global Interrupt Enable (GIE) bit is reset to disable further interrupts. The microcontroller then vectors to the address 00FFH and resumes execution from that address. This process takes 7 cycles to complete. At the end of the interrupt subroutine, any of the following three instructions return the processor back to the main program: RET, RETSK or RETI. Either one of the three instructions will pop the stack into the program counter (PC). The stack pointer is then incremented twice. The RETI instruction additionally sets the GIE bit to re-enable further interrupts.

Any of the three instructions can be used to return from a hardware interrupt subroutine. The RETSK instruction should be used when returning from a software interrupt subroutine to avoid entering an infinite loop.

DETECTION OF ILLEGAL CONDITIONS

The COP8720C incorporates a hardware mechanism that allows it to detect illegal conditions which may occur from coding errors, noise and 'brown out' voltage drop situations. Specifically it detects cases of executing out of undefined ROM area and unbalanced stack situations.

Reading an undefined ROM location returns 00 (hexadecimal) as its contents. The opcode for a software interrupt is also '00'. Thus a program accessing undefined ROM will cause a software interrupt.

Reading an undefined RAM location returns an FF (hexadecimal). The subroutine stack on the COP8720C grows down for each subroutine call. By initializing the stack pointer to the top of RAM, the first unbalanced return instruction will cause the stack pointer to address undefined RAM. As a result the program will attempt to execute from FFFF (hexadecimal), which is an undefined ROM location and will trigger a software interrupt.

MICROWIRE/PLUS™

MICROWIRE/PLUS is a serial synchronous bidirectional communications interface. The MICROWIRE/PLUS capability enables the COP8720C to interface with any of National Semiconductor's Microwire peripherals (i.e. A/D converters, display drivers, etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 8* shows the block diagram of the MICROWIRE/PLUS interface.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

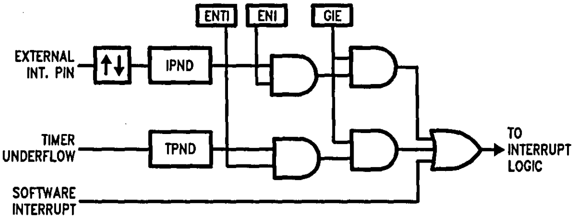
The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. The SK clock rate is selected by the two bits, S0 and S1, in the CNTRL register. Table III details the different clock rates that may be selected.

TABLE III

S1	S0	SK Cycle Time
0	0	$2t_C$
0	1	$4t_C$
1	x	$8t_C$

where,

t_C is the instruction cycle clock.



TL/DD/9108-11

FIGURE 7. Interrupt Block Diagram

Functional Description (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS arrangement to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. The COP8720C may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 9* shows how two COP8720C microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangement.

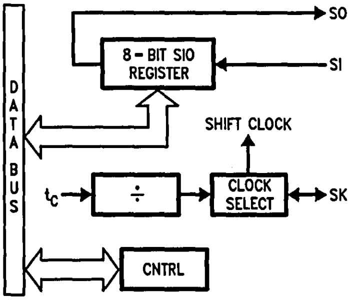
Master MICROWIRE/PLUS Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP8720C. The MICROWIRE/PLUS Master always initiates all data exchanges. (See *Figure 9*.) The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table IV summaries the bit settings required for Master mode of operation.

SLAVE MICROWIRE/PLUS OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by appropriately setting up the Port G configuration register. Table IV summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated. (See *Figure 9*.)



TL/DD/9108-12

FIGURE 8. MICROWIRE/PLUS Block Diagram

TABLE IV

G4 Config. Bit	G5 Config. Bit	G4 Fun.	G5 Fun.	G6 Fun.	Operation
1	1	SO	Int. SK	SI	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	SI	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	SI	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	SI	MICROWIRE/PLUS Slave

TIMER/COUNTER

The COP8720C has a powerful 16-bit timer with an associated 16-bit register enabling them to perform extensive timer functions. The timer T1 and its register R1 are each organized as two 8-bit read/write registers. Control bits in the register CNTRL allow the timer to be started and stopped under software control. The timer-register pair can be operated in one of three possible modes. Table V details various timer operating modes and their requisite control settings.

MODE 1. TIMER WITH AUTO-LOAD REGISTER

In this mode of operation the timer T1 counts down at the instruction cycle rate. Upon underflow the value in the register R1 gets automatically reloaded into the timer which continues to count down. The timer underflow can be programmed to interrupt the microcontroller. A bit in the control register CNTRL enables the TIO (G3) pin to toggle upon timer underflows. This allow the generation of square-wave outputs or pulse width modulated outputs under software control. (See *Figure 10*.)

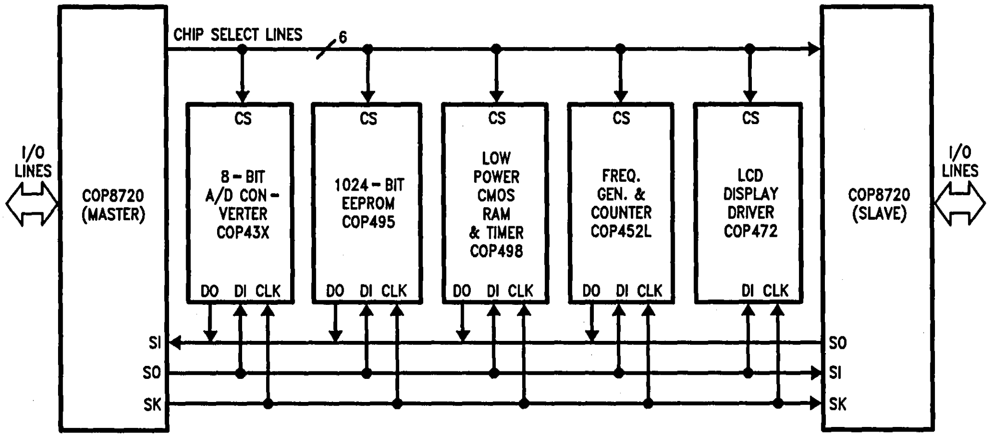
MODE 2. EXTERNAL COUNTER

In this mode, the timer T1 becomes a 16-bit external event counter. The counter counts down upon an edge on the TIO pin. Control bits in the register CNTRL program the counter to decrement either on a positive edge or on a negative edge. Upon underflow the contents of the register R1 are automatically copied into the counter. The underflow can also be programmed to generate an interrupt. (See *Figure 10*.)

MODE 3. TIMER WITH CAPTURE REGISTER

Timer T1 can be used to precisely measure external frequencies or events in this mode of operation. The timer T1 counts down at the instruction cycle rate. Upon the occurrence of a specified edge on the TIO pin the contents of the timer T1 are copied into the register R1. Bits in the control register CNTRL allow the trigger edge to be specified either as a positive edge or as a negative edge. In this mode the user can elect to be interrupted on the specified trigger edge. (See *Figure 11*.)

Functional Description (Continued)

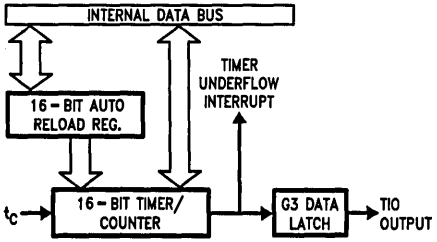

CHIP SELECT LINES
6
I/O LINES
COP8720 (MASTER)
CS
8-BIT A/D CON-VERTER COP43X
1024-BIT EEPROM COP495
LOW POWER CMOS RAM & TIMER COP498
FREQ. GEN. & COUNTER COP452L
LCD DISPLAY DRIVER COP472
DO DI CLK
SI
SO
SK
COP8720 (SLAVE)
I/O LINES

TL/DD/9108-13

FIGURE 9. MICROWIRE/PLUS Application

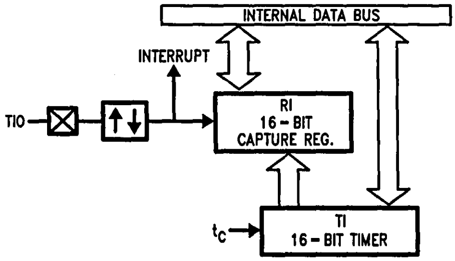
TABLE V. Timer Operating Modes

CNTRL Bits 7 6 5	Operation Mode	T Interrupt	Timer Counts On
0 0 0	External Counter W/Auto-Load Reg.	Timer Carry	TIO Pos. Edge
0 0 1	External Counter W/Auto-Load Reg.	Timer Carry	TIO Neg. Edge
0 1 0	Not Allowed	Not Allowed	Not Allowed
0 1 1	Not Allowed	Not Allowed	Not Allowed
1 0 0	Timer W/Auto-Load Reg.	Timer Carry	t_C
1 0 1	Timer W/Auto-Load Reg./Toggle TIO Out	Timer Carry	t_C
1 1 0	Timer W/Capture Register	TIO Pos. Edge	t_C
1 1 1	Timer W/Capture Register	TIO Neg. Edge	t_C



TL/DD/9108-15

FIGURE 10. Timer/Counter Auto Reload Mode Block Diagram



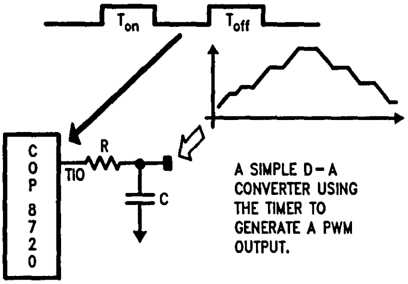
TL/DD/9108-14

FIGURE 11. Timer Capture Mode Block Diagram

Functional Description (Continued)

TIMER PWM APPLICATION

Figure 12 shows how a minimal component D/A converter can be built out of the Timer-Register pair in the Auto-Reload mode. The timer is placed in the "Timer with auto reload" mode and the TIO pin is selected as the timer output. At the outset the TIO pin is set high, the timer T1 holds the on time and the register R1 holds the signal off time. Setting TRUN bit starts the timer which counts down at the instruction cycle rate. The underflow toggles the TIO output and copies the off time into the timer, which continues to run. By alternately loading in the on time and the off time at each successive interrupt a PWM frequency can be easily generated.



TL/DD/9108-16

FIGURE 12. Timer Application

Control Registers

CNTRL REGISTER (ADDRESS X'OOEE)

The Timer and MICROWIRE/PLUS control register contains the following bits:

S1 & S0 Select the MICROWIRE/PLUS clock divide-by

IEDG External interrupt edge polarity select (0 = rising edge, 1 = falling edge)

MSEL Enable MICROWIRE/PLUS functions S0 and SK

TRUN Start/Stop the Timer/Counter (1 = run, 0 = stop)

TC3 Timer input edge polarity select (0 = rising edge, 1 = falling edge)

TC2 Selects the capture mode

TC1 Selects the timer mode

TC1	TC2	TC3	TRUN	MSEL	IEDG	S1	S0
BIT 7							BIT 0

PSW REGISTER (ADDRESS x'00EF)

The PSW register contains the following select bits:

GIE Global interrupt enable

ENI External interrupt enable

BUSY MICROWIRE/PLUS busy shifting

IPND External interrupt pending

ENTI Timer interrupt enable

TPND Timer interrupt pending

C Carry Flag

HC Half carry Flag

HC	C	TPND	ENTI	IPND	BUSY	ENI	GIE
Bit 7							Bit 0

Operating Modes

These controllers have two operating modes: Single Chip mode and the ROMless mode. The operating mode is determined by the state of the D2 pin at power on reset.

SINGLE CHIP MODE

In the Single Chip mode, the controller functions as a self contained microcontroller. It can address internal RAM and ROM. All ports configured as memory mapped I/O ports.

ROMLESS MODE

The COP8720C will enter the ROMless mode of operation if the D2 pin is held at logical "0" at reset. In this case the internal PROGRAM EEPROM is disabled and the controller can now address up to 32 kbytes of external program memory. It continues to use the on board RAM, and DATA EEPROM.

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address	Contents
00 to 2F	On Chip RAM Bytes
30 to 7F	Unused RAM Address Space (Reads as all Ones)
80 to BF	64 Bytes DATA EEPROM
C0 to CF	Expansion Space for I/O and Registers
D0 to DF	On Chip I/O and Registers
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8-DB	Reserved for Port C
DC	Port D Data Register
DD-DF	Reserved for Port D
E0 to EF	On Chip Functions and Registers
E0	EECR
E1	EROMDR
E2	EEAR Low Byte
E3	EEAR High Byte
E4-E8	Reserved
E9	MICROWIRE/PLUS Shift Register
EA	Timer Lower Byte
EB	Timer Upper Byte
EC	Timer Autoload Register Lower Byte
ED	Timer Autoload Register Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FF	On Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register

Memory Map (Continued)

Reading unused memory locations below 7FH will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

REGISTER INDIRECT

This is the "normal" mode of addressing for the COP8720C. The operand is the memory addressed by the B register or X register.

DIRECT

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

IMMEDIATE

The instruction contains an 8-bit immediate field as the operand.

REGISTER INDIRECT (AUTO INCREMENT AND DECREMENT)

This is a register indirect mode that automatically increments or decrements the B or X register after executing the instruction.

RELATIVE

This mode is used for the JP instruction, the instruction field is added to the program counter to get the new program location. JP has a range of from −31 to +32 to allow a one byte relative jump (JP + 1 is implemented by a NOP instruction). There are no 'pages' when using JP, all 15 bits of PC are used.

Instruction Set

REGISTER AND SYMBOL DEFINITIONS

Registers

A	8-bit Accumulator register
B	8-bit Address register
X	8-bit Address register
SP	8-bit Stack pointer register
PC	15-bit Program counter register
PU	upper 7 bits of PC
PL	lower 8 bits of PC
C	1-bit of PSW register for carry
HC	Half Carry
GIE	1-bit of PSW register for global interrupt enable

Symbols

[B]	Memory indirectly addressed by B register
[X]	Memory indirectly addressed by X register
Mem	Direct address memory or [B]
MemI	Direct address memory or [B] or Immediate data
Imm	8-bit Immediate data
Reg	Register memory: addresses F0 to FF (Includes B, X and SP)
Bit	Bit number (0 to 7)
←	Loaded with
←→	Exchanged with

Instruction Set (Continued)

Instruction Set

ADD	add	A ← A + Meml

ADC	add with carry	A ← A + Meml + C, C ← Carry
		HC ← Half Carry

SUBC	subtract with carry	A ← A + $\overline{\text{Meml}}$ +C, C ← Carry
		HC ← Half Carry

AND	Logical AND	A ← A and Meml
OR	Logical OR	A ← A or Meml
XOR	Logical Exclusive-OR	A ← A xor Meml
IFEQ	IF equal	Compare A and Meml, Do next if A = Meml
IFGT	IF greater than	Compare A and Meml, Do next if A > Meml
IFBNE	IF B not equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Decrement Reg. ,skip if zero	Reg ← Reg − 1, skip if Reg goes to 0
SBIT	Set bit	1 to bit, Mem (bit= 0 to 7 immediate)
RBIT	Reset bit	0 to bit, Mem
IFBIT	If bit	If bit, Mem is true, do next instr.
X	Exchange A with memory	A ←→ Mem
LD A	Load A with memory	A ← Meml
LD mem	Load Direct memory Immed.	Mem ← Imm
LD Reg	Load Register memory Immed.	Reg ← Imm
X	Exchange A with memory [B]	A ←→ [B] (B ← B±1)
X	Exchange A with memory [X]	A ←→ [X] (X ← X±1)
LD A	Load A with memory [B]	A ← [B] (B ← B±1)
LD A	Load A with memory [X]	A ← [X] (X ← X±1)
LD M	Load Memory Immediate	[B] ← Imm (B ← B±1)
CLRA	Clear A	A ← 0
INCA	Increment A	A ← A + 1
DECA	Decrement A	A ← A − 1
LAID	Load A indirect from ROM	A ← ROM(PU,A)
DCORA	DECIMAL CORRECT A	A ← BCD correction (follows ADC, SUBC)
RRCA	ROTATE A RIGHT THRU C	C → A7 → ... → A0 → C
SWAPA	Swap nibbles of A	A7...A4 ←→ A3...A0
SC	Set C	C ← 1, HC ← 1
RC	Reset C	C ← 0, HC ← 0
IFC	If C	If C is true, do next instruction
IFNC	If not C	If C is not true, do next instruction
JMPL	Jump absolute long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Jump absolute	PC11..0 ← i (i = 12 bits)
JP	Jump relative short	PC ← PC + r (r is −31 to +32, not 1)
JSRL	Jump subroutine long	[SP] ← PL,[SP-1] ← PU,SP-2,PC ← ii
JSR	Jump subroutine	[SP] ← PL,[SP-1] ← PU,SP-2,PC11..0 ← i
JID	Jump indirect	PL ← ROM(PU,A)
RET	Return from subroutine	SP+2,PL ← [SP],PU ← [SP-1]
RETSK	Return and Skip	SP+2,PL ← [SP],PU ← [SP-1],Skip next instruction
RETI	Return from Interrupt	SP+2,PL ← [SP],PU ← [SP-1],GIE ← 1
INTR	Generate an interrupt	[SP] ← PL,[SP−1] ← PU,SP-2,PC ← 0FF
NOP	No operation	PC ← PC + 1

OPCODE LIST

Bits 7–4

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
JP -15	JP -31	LD 0F0,#i	DRSZ 0F0	RRCA	RC	ADC A, #i	ADC A, [B]	IFBIT 0,[B]	*	LD B, 0F	IFBNE 0	JSR 0000-00FF	JMP 0000-00FF	JP + 17	INTR	0
JP -14	JP -30	LD 0F1,#i	DRSZ 0F1	*	SC	SUBC A, #i	SUBC A,[B]	IFBIT 1,[B]	*	LD B, 0E	IFBNE 1	JSR 0100-01FF	JMP 0100-01FF	JP + 18	JP + 2	1
JP -13	JP -29	LD 0F2,#i	DRSZ 0F2	X A, [X+]	X A, [B+]	IFEQ A, #i	IFEQ A,[B]	IFBIT 2,[B]	*	LD B, 0D	IFBNE 2	JSR 0200-02FF	JMP 0200-02FF	JP + 19	JP + 3	2
JP -12	JP -28	LD 0F3,#i	DRSZ 0F3	X A, [X−]	X A, [B−]	IFGT A, #i	IFGT A,[B]	IFBIT 3,[B]	*	LD B, 0C	IFBNE 3	JSR 0300-03FF	JMP 0300-03FF	JP + 20	JP + 4	3
JP -11	JP -27	LD 0F4,#i	DRSZ 0F4	*	LAID	ADD A, #i	ADD A,[B]	IFBIT 4,[B]	CLRA	LD B, 0B	IFBNE 4	JSR 0400-04FF	JMP 0400-04FF	JP + 21	JP + 5	4
JP -10	JP -26	LD 0F5,#i	DRSZ 0F5	*	JID	AND A, #i	AND A,[B]	IFBIT 5,[B]	SWAPA	LD B, 0A	IFBNE 5	JSR 0500-05FF	JMP 0500-05FF	JP + 22	JP + 6	5
JP -9	JP -25	LD 0F6,#i	DRSZ 0F6	X A, [X]	X A, [B]	XOR A, #i	XOR A,[B]	IFBIT 6,[B]	DCORA	LD B, 9	IFBNE 6	JSR 0600-06FF	JMP 0600-06FF	JP + 23	JP + 7	6
JP -8	JP -24	LD 0F7,#i	DRSZ 0F7	*	*	OR A, #i	OR A,[B]	IFBIT 7,[B]	*	LD B, 8	IFBNE 7	JSR 0700-07FF	JMP 0700-07FF	JP + 24	JP + 8	7
JP -7	JP -23	LD 0F8,#i	DRSZ 0F8	NOP	*	LD A, #i	IFC	SBIT 0,[B]	RBIT 0,[B]	LD B, 7	IFBNE 8	JSR 0800-08FF	JMP 0800-08FF	JP + 25	JP + 9	8
JP -6	JP -22	LD 0F9,#i	DRSZ 0F9	*	*	*	IFNC	SBIT 1,[B]	RBIT 1,[B]	LD B, 6	IFBNE 9	JSR 0900-09FF	JMP 0900-09FF	JP + 26	JP + 10	9
JP -5	JP -21	LD 0FA,#i	DRSZ 0FA	LD A, [X+]	LD A, [B+]	LD [B+],#i	INCA	SBIT 2,[B]	RBIT 2,[B]	LD B, 5	IFBNE 0A	JSR 0A00-0AFF	JMP 0A00-0AFF	JP + 27	JP + 11	A
JP -4	JP -20	LD 0FB,#i	DRSZ 0FB	LD A, [X−]	LD A, [B−]	LD [B−],#i	DECA	SBIT 3,[B]	RBIT 3,[B]	LD B, 4	IFBNE 0B	JSR 0B00-0BFF	JMP 0B00-0BFF	JP + 28	JP + 12	B
JP -3	JP -19	LD 0FC,#i	DRSZ 0FC	LD Md, #i	JMPL	X A,Md	*	SBIT 4,[B]	RBIT 4,[B]	LD B, 3	IFBNE 0C	JSR 0C00-0CFF	JMP 0C00-0CFF	JP + 29	JP +13	C
JP -2	JP -18	LD 0FD,#i	DRSZ 0FD	DIR	JSRL	LD A, Md	RETSK	SBIT 5,[B]	RBIT 5,[B]	LD B, 2	IFBNE 0D	JSR 0D00-0DFF	JMP 0D00-0DFF	JP + 30	JP +14	D
JP -1	JP -17	LD 0FE,#i	DRSZ 0FE	LD A, [X]	LD A, [B]	LD [B], #i	RET	SBIT 6, [B]	RBIT 6, [B]	LD B, 1	IFBNE 0E	JSR 0E00–0EFF	JMP 0E00–0EFF	JP + 31	JP +15	E
JP -0	JP -16	LD 0FF,#1	DRSZ 0FF	*	*	*	RETI	SBIT 7,[B]	RBIT 7,[B]	LD B, 0	IFBNE 0F	JSR 0F00-0FFF	JMP 0F00-0FFF	JP + 32	JP + 16	F

Bits 3–0

where, i is the immediate data Md is a directly addressed memory location * is an unused opcode (see following table)

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instruction taking two bytes).

Most single instructions take one cycle time (1 µs at 20 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 µs at 20 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr & Decr		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B,Imm				1/1			(If B < 16)
LD B,Imm				2/3			(If B > 15)
LD Mem,Imm	2/2		3/3		2/2		
LD Reg,Imm				2/3			

* => Memory location addressed by B or X or directly.

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCORA	1/1
RRCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Bytes and Cyles per Instruction (Continued)

The following table shows the instructions assigned to unused opcodes. This table is for information only. The operations performed are subject to change without notice. Do not use these opcodes.

Unused Opcode	Instruction	Unused Opcode	Instruction
60	NOP	A9	NOP
61	NOP	AF	LD A, [B]
62	NOP	B1	C → HC
63	NOP	B4	NOP
67	NOP	B5	NOP
8C	RET	B7	X A, [X]
99	NOP	B9	NOP
9F	LD [B], #i	BF	LD A, [X]
A7	X A, [B]		
A8	NOP		

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs, and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

To program the COP8720C, a special adapter board is provided. This adapter board contains a socket for the COP8720C and plugs directly into the MOLE prom programmer.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820/ COP840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes Compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

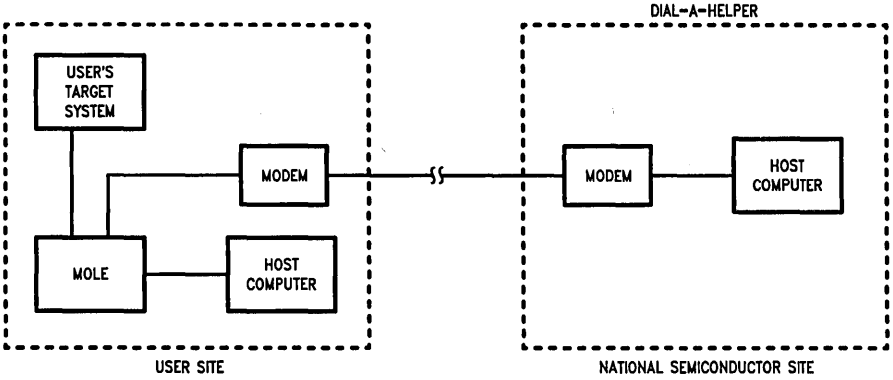
ORDER P/N: MOLE-DIAL-A-HLP

Information System Package Contains:
Dial-A-Helper User's Manual Pin
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582	
Modem:	(408) 739-1162	
	Baud:	300 or 1200 Baud
	Setup:	Length: 8-Bit
		Parity: None
		Stop Bit: 1
	Operation:	24 Hours, 7 Days



TL/DD/9108-23

National Semiconductor

ADVANCE INFORMATION

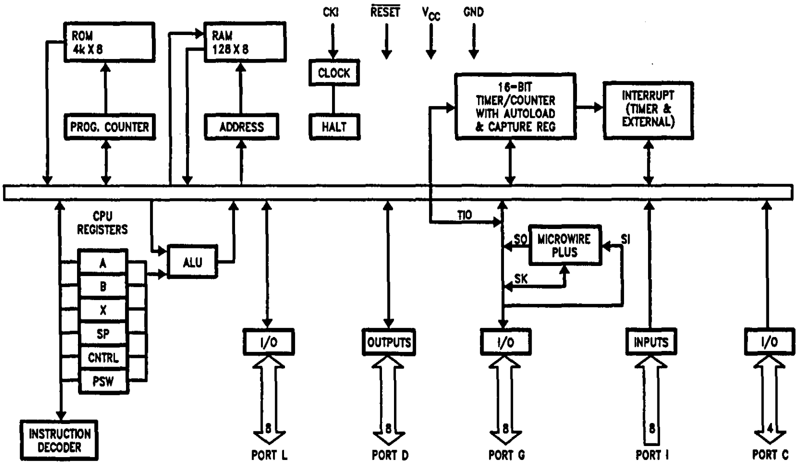
COP880 Single Chip Microcontroller

General Description

The COP880C is a member of the COPS™ 8-bit MicroController family. It is a fully static Microcontroller, fabricated using double-metal silicon gate microCMOS technology. This low cost Microcontroller is a complete microcomputer containing all system timing, interrupt logic, ROM, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the COP880C to the specific application. The COP880C operates over a voltage range of 2.5V to 6.0V. High throughput is achieved with an efficient, regular instruction set operating at a 1 µs per instruction rate. The COP880C may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Features

- Low cost 8-bit MicroController
- Fully static CMOS
- 1 µs instruction time (20 MHz clock)
- Low current drain (2.2 mA at 3 µs instruction rate)
- Extra-low current static HALT mode (Typically < 1 µA)
- Single supply operation: 2.5V to 6.0V
- 4096 x 8 on-chip ROM
  - Expandable to 32k bytes in ROMless mode
- 128 bytes on-chip RAM
- 16-bit read/write timer operates in a variety of modes
  - Timer with 16-bit auto reload register
  - 16-bit external event counter
  - Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - External interrupt with selectable edge
  - Timer interrupt or capture interrupt
  - Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions are single byte
- BCD arithmetic instructions
- MICROWIRE PLUS™ serial I/O
- Packages:
  - 44 PLCC with 36 I/O pins
  - 40 DIP with 36 I/O pins
  - 28 DIP and PLCC with 24 I/O pins
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Temperature ranges:
  - −40°C to +85°C
  - −55°C to +125°C
- ROMless mode for accurate emulation and external program capability
- Fully supported by National's Development Systems



TL/DD/10466-1

FIGURE 1. COP880C Block Diagram

National Semiconductor

PRELIMINARY

COP888CL Single-Chip microCMOS Microcontroller

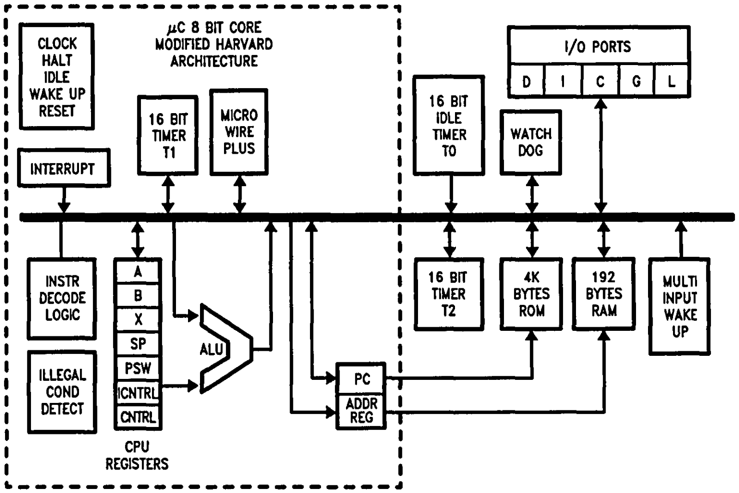
General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M2CMOS™ process technology. The COP888CL is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 4096 bytes on-board ROM
- 128 bytes on-board RAM
- Single supply operation: 2.5V–6V
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Timers TA, TB (Each with 2 Interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — Default VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC or 40 N or 28 N or 28 PCC
  - — 44 PCC with 39 I/O pins
  - — 40 N with 33 I/O pins
  - — 28 PCC or 28 N, each with 23 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: −40°C to +85°C,
  −55°C to +125°C
- ROMless mode for accurate emulation and external program memory capability
- Single chip COP888CLP piggy back emulation device
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram



TL/DD/9766–1

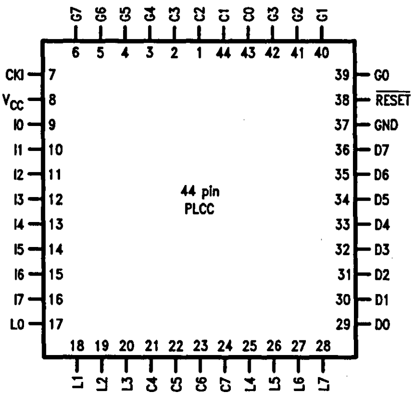
FIGURE 1. COP888CL Block Diagram

General Description (Continued)

It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CL operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The COP888CL may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Connection Diagrams

Plastic Chip Carrier

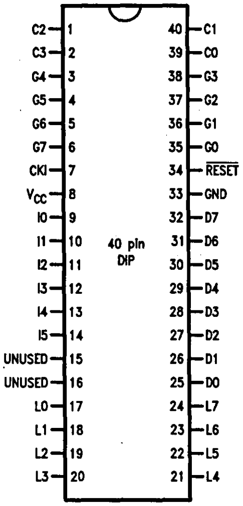


TL/DD/9766-2

Top View

Order Number COP888CL-XXX/V
See NS Plastic Chip Package Number V44A

Dual-In-Line Package

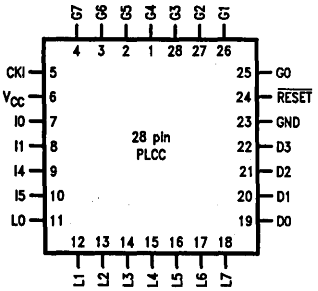


TL/DD/9766-4

Top View

Order Number COP888CL-XXX/N
See NS Molded Package Number N40A

Plastic Chip Carrier



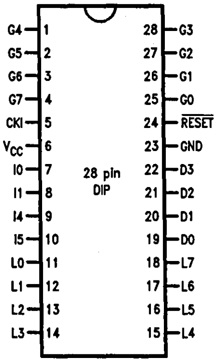
TL/DD/9766-3

Top View

Order Number COP884CL-XXX/V
See NS Plastic Chip Package Number V28A

***Note:** The pins labeled unused must be connected to GND.

Dual-In-Line Package



TL/DD/9766-5

Top View

Order Number COP884CL-XXX/N
See NS Molded Package Number N28A

FIGURE 2. COP888CL Connection Diagrams

Connection Diagrams (Continued)

COP888CL Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU		12	18	18
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT RESTART		4	6	6
D0	O	ROM DATA*		19	25	29
D1	O	PCL*		20	26	30
D2	O	EMUL*		21	27	31
D3	O	PCU*		22	28	32
I0	I			7	9	9
I1	I			8	10	10
I2	I				11	11
I3	I				12	12
I4	I			9	13	13
I5	I			10	14	14
I6	I					15
I7	I					16
D4	O	S CLOCK*			29	33
D5	O	HALTSEL*			30	34
D6	O	LOAD*			31	35
D7	O	D DATA*			32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
Unused**					16	
Unused**					15	
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

* = Only in the ROMless Mode

** = On the 40-pin package Pins 15 and 16 must be connected to GND.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	$-65°C$ to $+140°C$

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			$0.1\ V_{CC}$	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1\ \mu s$			15	mA
CKI = 4 MHz	$V_{CC} = 2.5V, t_c = 2.5\ \mu s$			2	mA
HALT Current (Note 3)	$V_{CC} = 6V$, CKI = 0 MHz		<1		μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1\ \mu s$			5	mA
CKI = 4 MHz	$V_{CC} = 2.5V, t_c = 2.5\ \mu s$			0.6	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		$0.8\ V_{CC}$			V
Logic Low				$0.2\ V_{CC}$	V
CKI (External and Crystal Osc. Modes)					
Logic High		$0.7\ V_{CC}$			V
Logic Low				$0.2\ V_{CC}$	V
All Other Inputs					
Logic High		$0.7\ V_{CC}$			V
Logic Low				$0.2\ V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 6V$	−2		+2	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis			$0.05\ V_{CC}$		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$	10			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	2.0			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$	10		100	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage		−2		+2	μA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source					
Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal or Resonator	$4V \leq V_{CC} \leq 6V$	1		DC	µs
	$2.5V \leq V_{CC} < 4V$	2.5		DC	µs
R/C Oscillator	$4V \leq V_{CC} \leq 6V$	3		DC	µs
	$2.5V \leq V_{CC} < 4V$	7.5		DC	µs
CKI Clock Duty Cycle (Note 5)	f_r = Max	40		60	%
Rise Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4V \leq V_{CC} \leq 6V$	200			ns
	$2.5V \leq V_{CC} < 4V$	500			ns
t_{HOLD}	$4V \leq V_{CC} \leq 6V$	60			ns
	$2.5V \leq V_{CC} < 4V$	150			ns
Output Propagation Delay	$R_L = 2.2k, C_L = 100$ pF				
t_{PD1}, t_{PD0}					
SO, SK	$4V \leq V_{CC} \leq 6V$			0.7	µs
	$2.5V \leq V_{CC} < 4V$			1.75	µs
All Others	$4V \leq V_{CC} \leq 6V$			1	µs
	$2.5V \leq V_{CC} < 4V$			2.5	µs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			µs

Note 5: Parameter sample but not 100% tested.

Note 6: Except Pin G7: −60 mA to +100 mA (sampled but not 100% tested).

AC Electrical Characteristics (Continued)

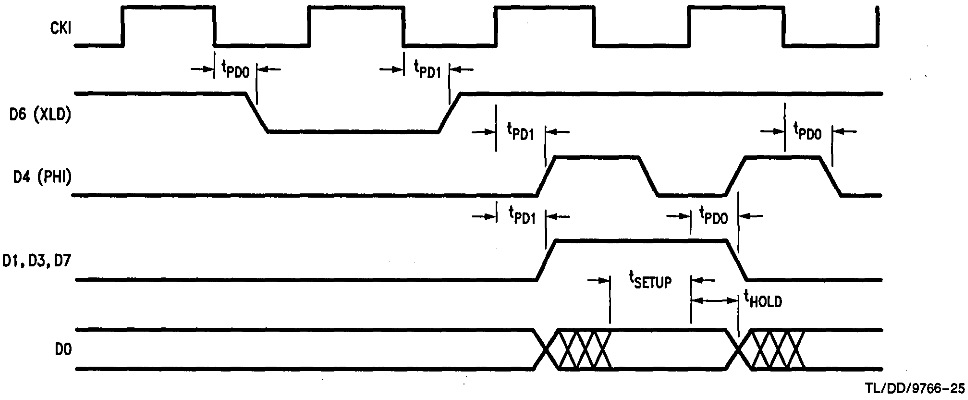


FIGURE 2a. AC Timing Diagrams in ROMless Mode

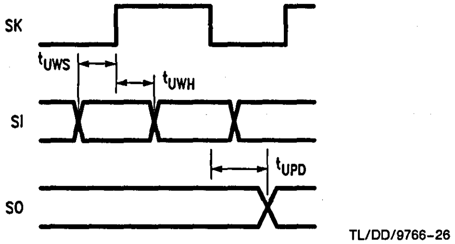


FIGURE 2b. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

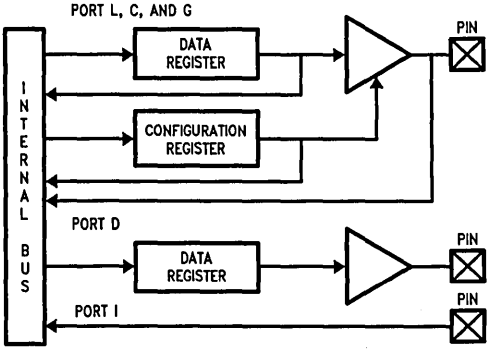
CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset Description section.

The COP888CL contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CL memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CL. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output

Pin Descriptions (Continued)



TL/DD/9766-6

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wakeup (MIWU) on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B.

Port L has the following alternate features:

L0	MIWU
L1	MIWU
L2	MIWU
L3	MIWU
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU
L7	MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin, but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin or general purpose input (R/C clock configuration), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0	INTR (External Interrupt Input)
G2	T1B (Timer T1 Capture Input)
G3	T1A (Timer T1 I/O)
G4	SO (MICROWIRE™ Serial Data Output)
G5	SK (MICROWIRE Serial Clock)
G6	SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1	WDOUT WatchDog and/or Clock Monitor dedicated output
G7	CKO Oscillator dedicated output or general purpose input

Port I is an 8-bit Hi-Z input port. The 40-pin device does not have a full complement of Port I pins. Pins 15 and 16 on this package must be connected to GND.

The 28-pin device has four I pins (I0, I1, I4, I5). The user should pay attention when reading port I to the fact that I4 and I5 are in bit positions 4 and 5 rather than 2 and 3.

The unavailable pins (I4–I7) are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes into account by either masking or restricting the accesses to bit operations. The unterminated port I pins will draw power only when addressed.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CL is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CL architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_C) cycle time.

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)
PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CL consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CL vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The COP888CL has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

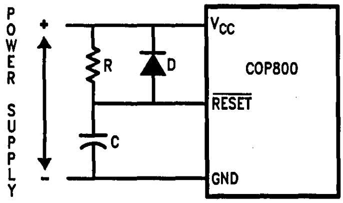
The instruction set of the COP888CL permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CL (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Reset

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the data and configuration registers for Ports L, G, and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is initialized high with $\overline{\text{RESET}}$. The PC, PSW, CNTRL, ICNTRL, and T2CNTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The Stack Pointer, SP, is initialized to 06F Hex.

The COP888CL comes out of reset with both the WatchDog logic and the Clock Monitor detector armed, and with both the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor detector circuits are inhibited during reset. The WatchDog service window bits are initialized to the maximum WatchDog service window of 64k t_C clock cycles. The Clock Monitor bit is initialized high, and will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16–32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 4* should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.



RC > 5 × Power Supply Rise Time TL/DD/9766-7

FIGURE 4. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.

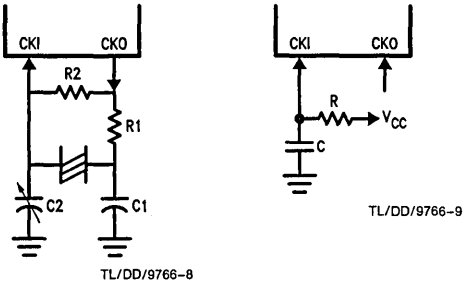


FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4	V_{CC} = 2.5V
0	1	200	100–150	0.455	V_{CC} = 2.5V

TABLE B. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8 to 2.2	3.6 to 4.5	V_{CC} = 5V
5.6	100	1.5 to 1.1	6.7 to 9	V_{CC} = 5V
6.8	100	1.1 to 0.8	9 to 12.5	V_{CC} = 2.5V

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. Clock Monitor current when enabled—I6

Thus the total current drain, It, is given as

$$It = I1 + I2 + I3 + I4 + I5 + I6$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Some sample current drain values at V_{CC} = 5V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)		0.005

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

SL1 & SL0	Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
IEDG	External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
MSEL	Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
T1C0	Timer T1 Start/Stop control in timer modes 1 and 2 Timer T1 Underflow Interrupt Pending Flag in timer mode 3
T1C1	Timer T1 mode control bit
T1C2	Timer T1 mode control bit
T1C3	Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0

Bit 7 Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt
BUSY	MICROWIRE/PLUS busy shifting flag
EXPND	External interrupt pending
T1ENA	Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge

Control Registers (Continued)

T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)

C Carry Flag

HC Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE

Bit 7 Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB Timer T1 Interrupt Enable for T1B Input capture edge

T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge

μWEN Enable MICROWIRE/PLUS interrupt

μWPND MICROWIRE/PLUS interrupt pending

T0EN Timer T0 Interrupt Enable (Bit 12 toggle)

T0PND Timer T0 Interrupt pending

LPEN L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PNDB	T1ENB

Bit 7 Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB Timer T2 Interrupt Enable for T2B Input capture edge

T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge

T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge

T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)

T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

T2C1 Timer T2 mode control bit

T2C2 Timer T2 mode control bit

T2C3 Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB

Bit 7 Bit 0

ROMless Mode

The COP888CL can address up to 32 kbytes of address space. If at power up, D2 is held at ground, the COP888CL executes from external memory. Port D is used to interface to external program memory. The address comes out in a serial fashion and the data from the external program memory is read back in a serial fashion. The Port D pins perform the following functions.

D0 Shifts in ROM data

D1 Shifts out lower eight bits of PC

D2 Places the μC in the ROMless mode if grounded at reset

D3 Shifts out upper eight bits of PC

D4 Data Shift Clock

D5 HALT Mask Option select pin (D5 = 0) for HALT enable, D5 = 1 for HALT disable)

D6 Load Clock

D7 Shifts out recreated Port D data

The most significant bit of the data to come out on the D3 pin is a status signal.. It is used by the MOLE development system. This "lost" output port (D0–D7) can be accurately reconstructed with external components as shown in *Figure 6*, providing an accurate emulation.

The 44-pin and 40-pin versions of the COP888CL have a full complement of the D Port pins and can be used in the ROMless mode.

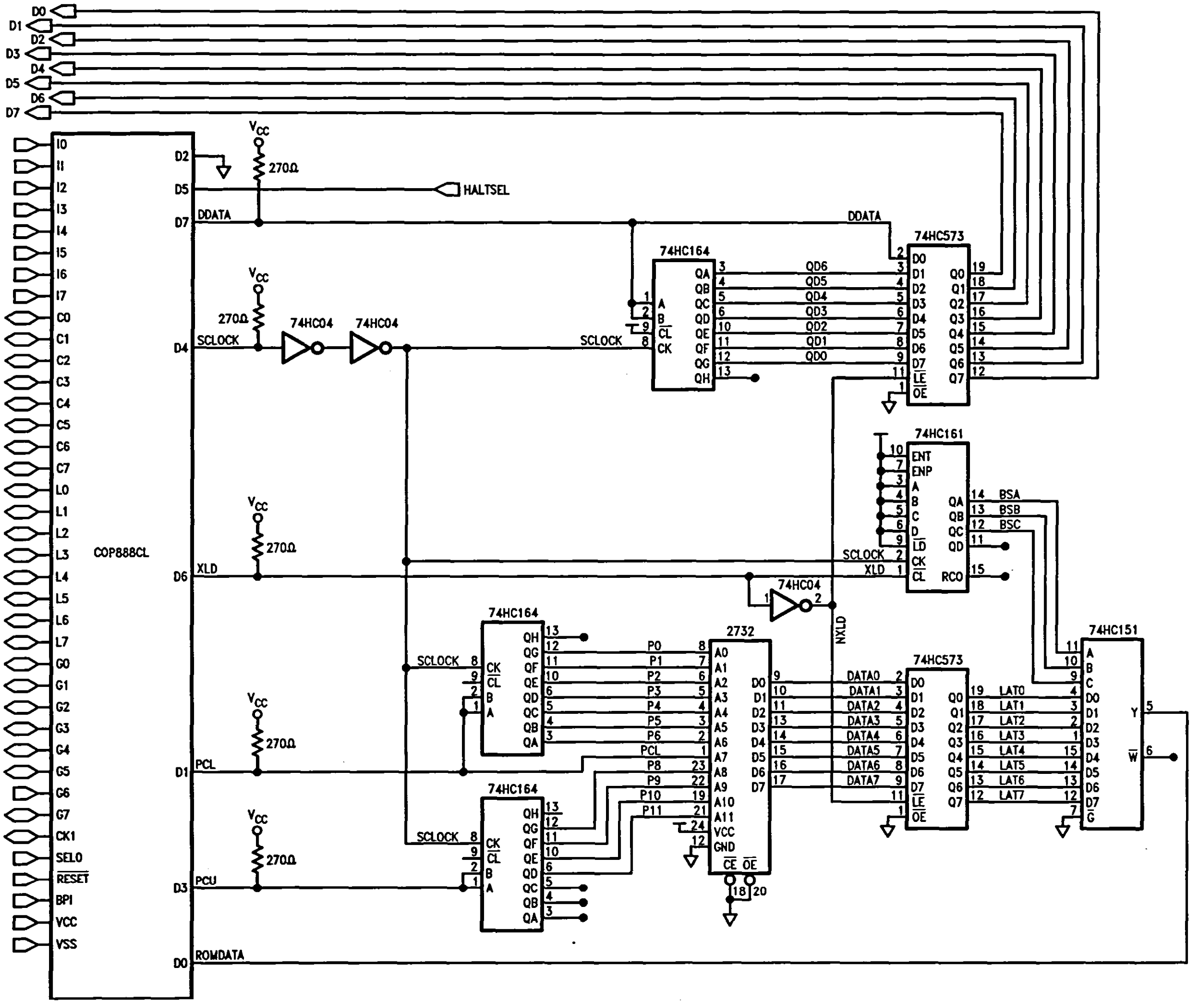
The 28-pin part cannot be used for emulation since it does not have the full complement of 8 D Port pins necessary for entering the ROMless mode.

Note that in the ROMless mode the D Port is recreated one full CKI clock cycle behind the normal port timings.

Note: Standard parts used in the ROMless mode will operate only at a reduced frequency (to be defined).

The COP888CL device has a spare D pin (D5) in the ROMless mode since only seven pins are required for emulation and recreation. This pin D5 is used in the ROMless mode to enable or disable the HALT mask option feature.

Figure 6 shows the COP888CL ROMless Mode Schematic.



TL/DD/9766–10

FIGURE 6. COP888CL ROMless Mode Schematic

Timers

The COP888CL contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

Figure 7 shows a block diagram for the timers on the COP888CL.

TIMER T0 (IDLE TIMER)

The COP888CL supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_C. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

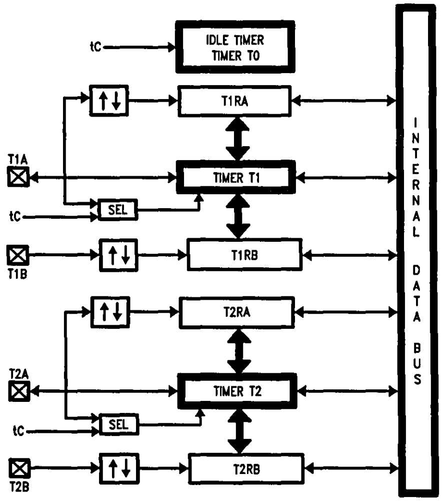
Exit out of the Idle Mode (See Idle Mode description)
WatchDog logic (See WatchDog description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_C = 1\ \mu s$). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1 AND TIMER T2

The COP888CL has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2, are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CL to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.



TL/DD/9766-11

FIGURE 7. Timers for the COP888CL

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CL to generate a PWM signal with very minimal user intervention.

The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

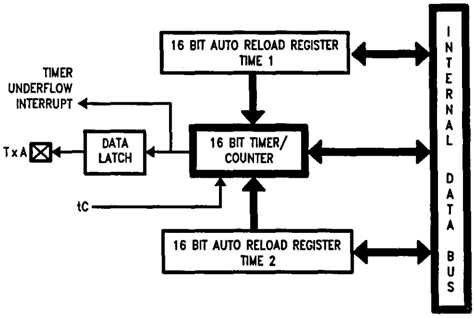
The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode.

The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.



TL/DD/9766-13

FIGURE 8. Timer in PWM Mode

Timers (Continued)

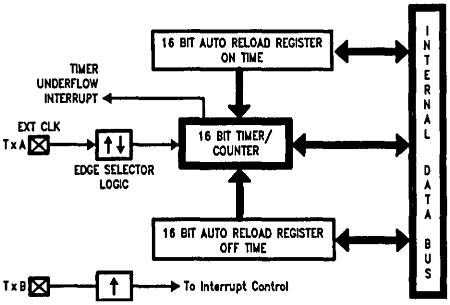
Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.


16 BIT AUTO RELOAD REGISTER ON TIME
TIMER UNDERFLOW INTERRUPT
EXT CLK
TxA
EDGE SELECTOR LOGIC
16 BIT TIMER/ COUNTER
16 BIT AUTO RELOAD REGISTER OFF TIME
TxB
To Interrupt Control
INTERNAL DATA BUS

TL/DD/9766-14

FIGURE 9. Timer In External Event Counter Mode

Mode 3. Input Capture Mode

The COP888CL can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_c rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

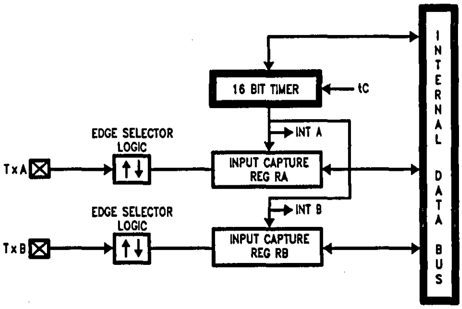
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

The timers T1 and T2 have indentical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control



TL/DD/9766-15

FIGURE 10. Timer In Input Capture Mode

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_C
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_C
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C

Power Save Modes

The COP888CL offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CL is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock, timers, are stopped. The WatchDog logic on the COP888CL is disabled during the HALT mode. However, the clock monitor circuitry, if enabled, remains active. In the HALT mode, the power requirements of the COP888CL are minimal and the applied voltage (V_{CC}) may be decreased to V_r (V_r = 2.0V) without altering the state of the machine.

The COP888CL supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the $\overline{\text{RESET}}$ pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_C instruction cycle clock. The t_C clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

Power Save Modes (Continued)

The COP888CL has two mask options associated with the HALT mode. The first mask option enables the HALT mode feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CL will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CL cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit, if enabled, remains active during HALT mode in order to ensure a clock monitor error if the COP888CL inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CL is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the WatchDog logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CL can be returned to normal operation with a reset, or with a Multi-Input Wake-up from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_C = 1\ \mu s$) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

The user has the option of being interrupted with a transition on the twelfth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the COP888CL will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CL will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

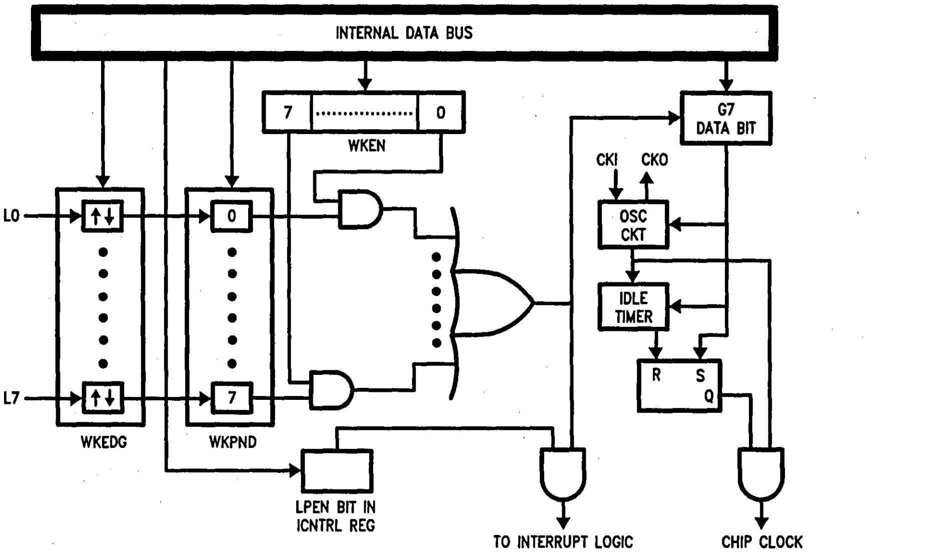


FIGURE 11. Multi-Input Wake Up Logic

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CL from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 11 shows the Multi-Input Wakeup logic for the COP888CL microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CL to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CL will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

The WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG

Multi-Input Wakeup (Continued)

specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CL out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CL will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CL will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CL to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

Interrupts

The COP888CL supports a vectored interrupt scheme. It supports a total of ten interrupt sources. The following table lists all the possible COP888CL interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	Underflow	0yF8–0yF9
(4)	Timer T1	T1A/Underflow	0yF6–0yF7
(5)	Timer T1	T1B	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
	Reserved	for UART	0yEE–0yEF
	Reserved	for UART	0yEC–0yED
(7)	Timer T2	T2A/Underflow	0yEA–0yEB
(8)	Timer T2	T2B	0yE8–0yE9
	Reserved	for Future Use	0yE6–0yE7
	Reserved	for Future Use	0yE4–0yE5
(9)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(10) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page, y ≠ 0.

Interrupts (Continued)

instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_C cycles to execute.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

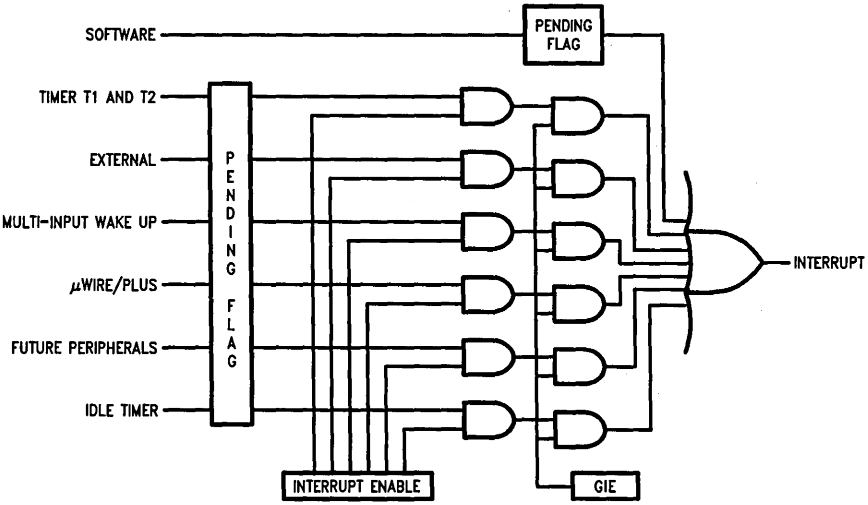
VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 12 shows the COP888CL Interrupt block diagram.



TL/DD/9766-18

FIGURE 12. COP888CL Interrupt Block Diagram

Interrupts (Continued)

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit **(not accessible by the user)** is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WatchDog

The COP888CL contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table I shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table II shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE I. WatchDog Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE II. WatchDog Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k-8k t_C Cycles
0	1	2k-16k t_C Cycles
1	0	2k-32k t_C Cycles
1	1	2k-64k t_C Cycles

Clock Monitor

The Clock Monitor aboard the COP888CL can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_C$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WatchDog Operation

The WatchDog and Clock Monitor are disabled during reset. The COP888CL comes out of reset with the WatchDog armed, the WatchDog Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WatchDog service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WatchDog service window and match the WatchDog key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WatchDog service window value and the key data (bits 7 through 1) in the WDSVR Register. Table III shows the sequence of events that can occur.

The user must service the WatchDog at least once before the upper limit of the serivce window expires. The WatchDog may not be serviced more than once in every lower limit of the service window. The user may service the WatchDog as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WatchDog service.

The WatchDog has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WatchDog, the logic will pull the WDOUT (G1) pin low for an additional 16 t_C–32 t_C cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CL will stop forcing the WDOUT output low.

The WatchDog service window will restart when the WDOUT pin goes high It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WatchDog service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WatchDog will time out and WDOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_C–32 t_C clock cycles. The Clock Monitor generates a continual Clock Moni-

WatchDog Operation (Continued)

TABLE III. WatchDog Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WatchDog Output
Mismatch	Don't Care	Don't Care	Error: Generate WatchDog Output
Don't Care	Don't Care	Mismatch	Error: Generate WatchDog Output

TABLE IV. MICROWIRE/PLUS Master Mode Clock Select

SL1	SL0	SK
0	0	$2 \times t_C$
0	1	$4 \times t_C$
1	x	$8 \times t_C$

Where t_C is the instruction cycle clock

tor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_C > 10$ kHz—No clock rejection.

$1/t_C < 10$ Hz—Guaranteed clock rejection.

Detection of Illegal Conditions

The COP888CL can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F Hex is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

a. Executing from undefined ROM

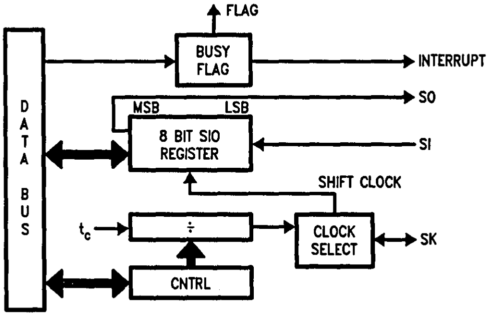
b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CL to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E^{2}PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 13* shows a block diagram of the MICROWIRE logic.

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE arrangement with an external shift clock is called the Slave mode of operation.



TL/DD/9766-20

FIGURE 13. MICROWIRE/PLUS Block Diagram

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table IV details the different clock rates that may be selected.

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The COP888CL may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 14* shows how two COP888CL microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. The SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock forthe SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CL. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table V summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

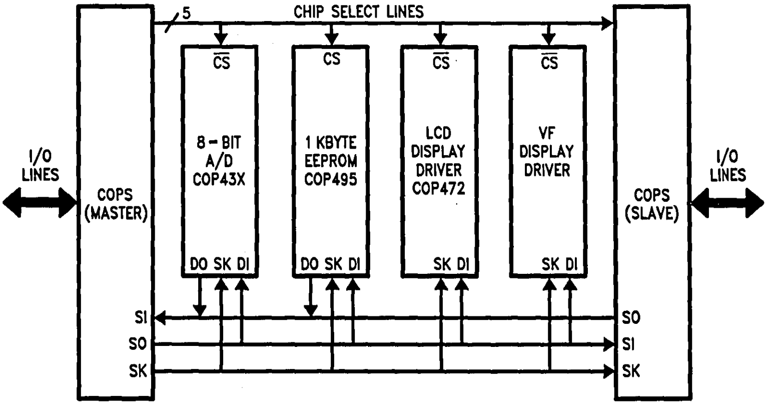
The COP888CL allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE V

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave



TL/DD/9766-21

FIGURE 14. MICROWIRE/PLUS Application

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space

Address	Contents
00 to 6F	On-Chip RAM bytes
70 to BF	Unused RAM Address Space
C0	Timer T2 Lower Byte
C1	Timer T2 Upper Byte
C2	Timer T2 Autoload Register T2RA Lower Byte
C3	Timer T2 Autoload Register T2RA Upper Byte
C4	Timer T2 Autoload Register T2RB Lower Byte
C5	Timer T2 Autoload Register T2RB Upper Byte
C6	Timer T2 Control Register
C7	WatchDog Service Register (Reg:WDSVR)
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	Reserved
CC	Reserved
CD to CF	Reserved
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved

Reading memory locations 70-7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

The COP888CL has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CL. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the LBI instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
Meml	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
⟷	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,MemI	ADD	A ← A + MemI

ADC	A,MemI	ADD with Carry	A ← A + MemI + C, C ← Carry
			HC ← Half Carry

SUBC	A,MemI	Subtract with Carry	A ← A − $\overline{\text{MemI}}$ + C, C ← Carry
			HC ← Half Carry

AND	A,MemI	Logical AND	A ← A and MemI
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,MemI	Logical OR	A ← A or MemI
XOR	A,MemI	Logical EXclusive OR	A ← A xor MemI
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,MemI	IF EQual	Compare A and MemI, Do next if A = MemI
IFNE	A,MemI	IF Not Equal	Compare A and MemI, Do next if A ≠ MemI
IFGT	A,MemI	IF Greater Than	Compare A and MemI, Do next if A > MemI
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg ← Reg− 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A ←→ Mem
X	A,[X]	EXchange A with Memory [X]	A ←→ [X]
LD	A,MemI	LoaD A with Memory	A ← MemI
LD	A,[X]	LoaD A with Memory [X]	A ← [X]
LD	B,Imm	LoaD B with Immed.	B ← Imm
LD	Mem,Imm	LoaD Memory Immed	Mem ← Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg ← Imm
X	A, [B ±]	EXchange A with Memory [B]	A ←→ [B], (B ← B ± 1)
X	A, [X ±]	EXchange A with Memory [X]	A ←→ [X], (X ← ±1)
LD	A, [B±]	LoaD A with Memory [B]	A ← [B], (B ← B ± 1)
LD	A, [X±]	LoaD A with Memory [X]	A ← [X], (X ← X ± 1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] ← Imm, (B ← ±1)
CLR	A	CLeaR A	A ← 0
INC	A	INCrement A	A ← A + 1
DEC	A	DECrementA	A ← A − 1
LAID		Load A InDirect from ROM	A ← ROM (PU,A)
DCOR	A	Decimal CORrect A	A ← BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C ←→ A7 ←→ ... ←→ A0 ←→ C
RLC	A	Rotate A Left thru C	C ← A7 ← ... ← A0 ← C
SWAP	A	SWAP nibbles of A	A7...A4 ←→ A3...A0
SC		Set C	C ← 1, HC ← 1
RC		Reset C	C ← 0, HC ← 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP ← SP + 1, A ← [SP]
PUSH	A	PUSH A onto the stack	[SP] ← A, SP ← SP − 1
VIS		Vector to Interrupt Service Routine	PU ← [VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9...0 ← i (i = 12 bits)
JP	Disp.	Jump relative short	PC ← PC + r (r is −31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] ← PL, [SP−1] ← PU,SP−2, PC ← ii
JSR	Addr	Jump SubRoutine	[SP] ← PL, [SP−1] ← PU,SP−2, PC9...0 ← i
JID		Jump InDirect	PL ← ROM (PU,A)
RET		RETurn from subroutine	SP+2, PL ← [SP], PU ← [SP−1]
RETSK		RETurn and SKip	SP+2, PL ← [SP],PU ← [SP−1]
RETI		RETurn from Interrupt	SP+2, PL ← [SP],PU ← [SP−1],GIE ← 1
INTR		Generate an Interrupt	[SP] ← PL, [SP−1] ← PU, SP−2, PC ← 0FF
NOP		No OPeration	PC ← PC+1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time (1 µs at 10 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 µs at 10 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B, Imm				1/1			(IF B < 16)
LD B, Imm				2/2			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

* = > Memory location addressed by B or X or directly.

COP888CL Opcode Table

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP −15	JP −31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	0
JP −14	JP −30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP −13	JP −29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	2
JP −12	JP −28	LD 0F3, # i	DRSZ 0F3	X A, [X−]	X A,[B−]	IFGT A,#i	IFGT A,[B]	3
JP −11	JP −27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	4
JP −10	JP −26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	5
JP −9	JP −25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	6
JP −8	JP −24	LD 0F7, # i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	7
JP −7	JP −23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	8
JP −6	JP −22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	9
JP −5	JP −21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+],#i	INCA	A
JP −4	JP −20	LD 0FB, # i	DRSZ 0FB	LD A,[X−]	LD A,[B−]	LD [B−],#i	DECA	B
JP −3	JP −19	LD 0FC, # i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	C
JP −2	JP −18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP −1	JP −17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B],#i	RET	E
JP −0	JP −16	LD 0FF, # i	DRSZ 0FF	*	*	LD B,#i	RETI	F

COP888CL Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B,#0F	IFBNE 0	JSR x000-x0FF	JMP x000-x0FF	JP +17	INTR	0
IFBIT 1,[B]	*	LD B,#0E	IFBNE 1	JSR x100-x1FF	JMP x100-x1FF	JP +18	JP + 2	1
IFBIT 2,[B]	*	LD B,#0D	IFBNE 2	JSR x200-x2FF	JMP x200-x2FF	JP +19	JP + 3	2
IFBIT 3,[B]	*	LD B,#0C	IFBNE 3	JSR x300-x3FF	JMP x300-x3FF	JP +20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400-x4FF	JMP x400-x4FF	JP +21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500-x5FF	JMP x500-x5FF	JP +22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600-x6FF	JMP x600-x6FF	JP +23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700-x7FF	JMP x700-x7FF	JP +24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800-x8FF	JMP x800-x8FF	JP +25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900-x9FF	JMP x900-x9FF	JP +26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00-xAFF	JMP xA00-xAFF	JP +27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00-xBFF	JMP xB00-xBFF	JP +28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00-xCFF	JMP xC00-xCFF	JP +29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00-xDFF	JMP xD00-xDFF	JP +30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00-xEFF	JMP xE00-xEFF	JP +31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00-xFFF	JMP xF00-xFFF	JP +32	JP + 16	F

Where,

i is the immediate data
Md is a directly addressed memory location
* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CL mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

```
OPTION 1: CLOCK CONFIGURATION
  = 1     Crystal Oscillator (CKI/10)
              G7 (CKO) is clock generator
              output to crystal/resonator
              CKI is the clock input
  = 2     Single-pin RC controlled
              oscillator (CKI/10)
              G7 is available as a HALT
              restart and/or general purpose
              input
OPTION 2: HALT
  = 1     Enable HALT mode
  = 2     Disable HALT mode
OPTION 3: COP888CL BONDING
  = 1     44-Pin PCC
  = 2     40-Pin DIP
  = 3     28-Pin PCC
  = 4     28-Pin DIP
```

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System and additionally provides the capability of remotely accessing the MOLE development system at a customer site.

Information System

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

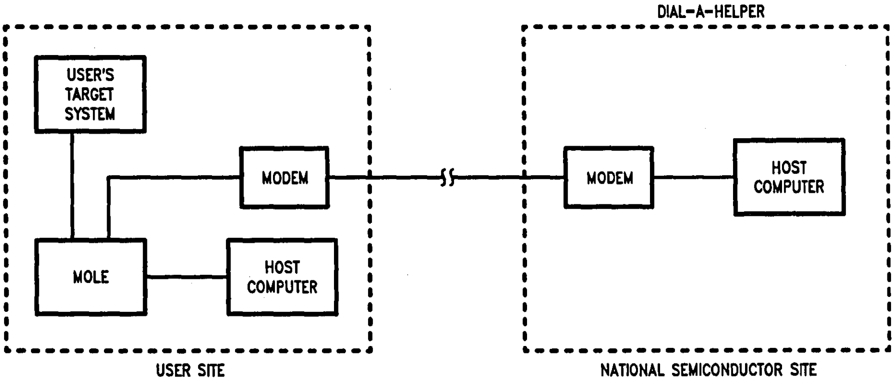
Order P/N: MOLE-DIAL-A-HLP

Information System Package Contents
- Dial-A-Helper User Manual P/N
- Public Domain Communications Software

Factory Applications Support

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 baud
Set-up: Length: 8-Bit
Parity: None
Stop Bit 1
Operation: 24 Hours, 7 Days



TL/DD/9766-24

National Semiconductor

PRELIMINARY

COP888CF Single-Chip microCMOS Microcontroller

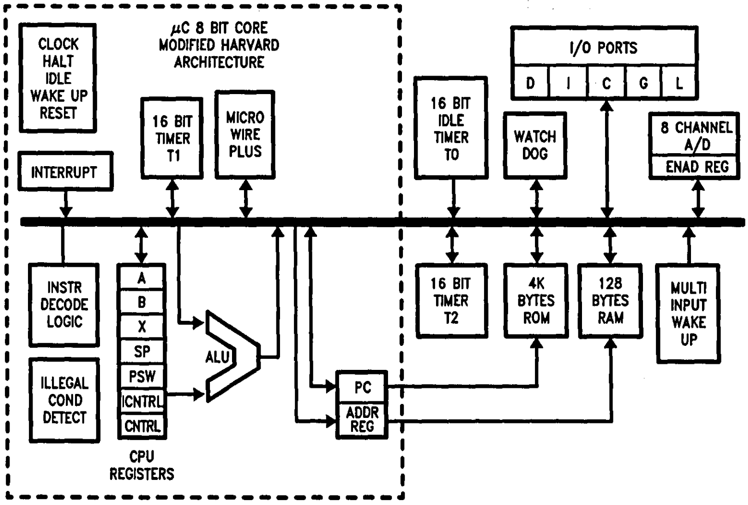
General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M2CMOS™ process technology. The COP888CF is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 µs instruction cycle time
- 4096 bytes on-board ROM
- 128 bytes on-board RAM
- Single supply operation: 2.5V–6V
- 8-channel A/D converter with prescaler and both differential and single ended modes
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
  - External Interrupt
  - Idle Timer T0
  - Two Timers (Each with 2 Interrupts)
  - MICROWIRE/PLUS
  - Multi-Input Wake Up
  - Software Trap
  - Default VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
  - Processor Independent PWM mode
  - External Event counter mode
  - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC or 40 N or 28 N or 28 PCC
  - 44 PCC with 37 I/O pins
  - 40 N with 33 I/O pins
  - 28 PCC or 28 N, each with 21 I/O pins
- Software selectable I/O options
  - TRI-STATE® Output
  - Push-Pull Output
  - Weak Pull Up Input
  - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: −40°C to + 85°C
−55°C to +125°C
- ROMless mode for accurate emulation and external program memory capability
- Single chip COP888CFP piggy back emulation device
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram



TL/DD/9425-1

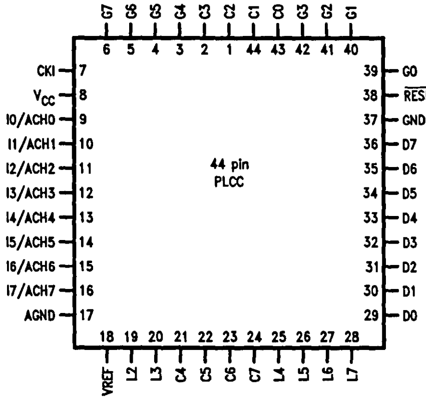
FIGURE 1. COP888CF Block Diagram

General Description (Continued)

It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), an 8-channel, 8-bit A/D converter with both differential and single ended modes, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CF operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The COP888CF may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Connection Diagrams

Plastic Chip Carrier

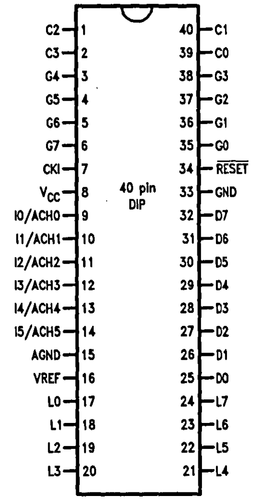


TL/DD/9425-2

Top View

Order Number COP888CF-XXX/V
See NS Plastic Chip Package Number V44A

Dual-In-Line Package

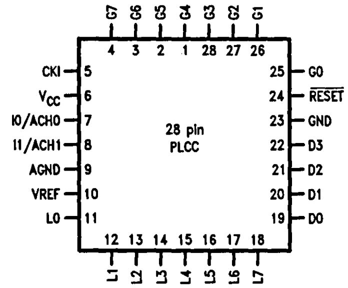


TL/DD/9425-4

Top View

Order Number COP888F-XXX/N
See NS Molded Package Number N40A

Plastic Chip Carrier

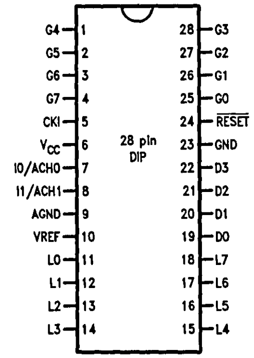


TL/DD/9425-3

Top View

Order Number COP884CF-XXX/V
See NS Plastic Chip Package Number V28A

Dual-In-Line Package



TL/DD/9425-5

Top View

Order Number COP884CF-XXX/N
See NS Molded Package Number N28A

FIGURE 2. COP888CF Connection Diagrams

COP888CF Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	—
L1	I/O	MIWU		12	18	—
L2	I/O	MIWU		13	19	19
L3	I/O	MIWU		14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU		17	23	27
L7	I/O	MIWU		18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O	ROM DATA*		19	25	29
D1	O	PCL*		20	26	30
D2	O	EMUL*		21	27	31
D3	O	PCU*		22	28	32
I0	I	ACH0		7	9	9
I1	I	ACH1		8	10	10
I2	I	ACH2			11	11
I3	I	ACH3			12	12
I4	I	ACH4			13	13
I5	I	ACH5			14	14
I6	I	ACH6				15
I7	I	ACH7				16
D4	O	S CLOCK*			29	33
D5	O	HALTSEL*			30	34
D6	O	LOAD*			31	35
D7	O	D DATA*			32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V_{REF}	$+V_{REF}$			10	16	18
AGND	AGND			9	15	17
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

* = Only in the ROMless Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	$-65°C$ to $+140°C$

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \leq T_A \leq +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V$, $t_c = 1\ \mu s$			15	mA
CKI = 4 MHz	$V_{CC} = 2.5V$, $t_c = 2.5\ \mu s$			2	mA
HALT Current (Note 3)	$V_{CC} = 6V$, CKI = 0 MHz		<26		μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 6V$, $t_c = 1\ \mu s$			5	mA
CKI = 4 MHz	$V_{CC} = 2.5V$, $t_c = 2.5\ \mu s$			0.6	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	−2		+2	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V$, $V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V$, $V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4V$, $V_{OL} = 1V$	10			mA
	$V_{CC} = 2.5V$, $V_{OL} = 0.4V$	2.0			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V$, $V_{OH} = 2.7V$	10		100	μA
	$V_{CC} = 2.5V$, $V_{OH} = 1.8V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4V$, $V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V$, $V_{OH} = 1.8V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4V$, $V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V$, $V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage		−2		+2	μA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. If the A/D is not being used and minimum standby current is desired, V_{REF} should be tied to AGND (effectively shorting the Reference resistor). The clock monitor is disabled.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				 15 3	 mA mA
Maximum Input Current without Latchup (Note 7)	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications $V_{CC} = 5V \pm 10\%$ $(V_{SS} - 0.050V) \le$ Any Input $\le (V_{CC} + 0.050V)$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			±1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the Best Straight Line			±½	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			±½	LSB
Input Reference Resistance		1.6		4.8	kΩ
Common Mode Input Range (Note 8)		AGND		V_{REF}	V
DC Common Mode Error				±¼	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
A/D Clock Frequency (Note 6)		0.1		1.67	MHz
Conversion Time (Note 5)			12		A/D Clock Cycles

Note 5: Conversion Time includes sample and hold time.

Note 6: See Prescaler description.

Note 7: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

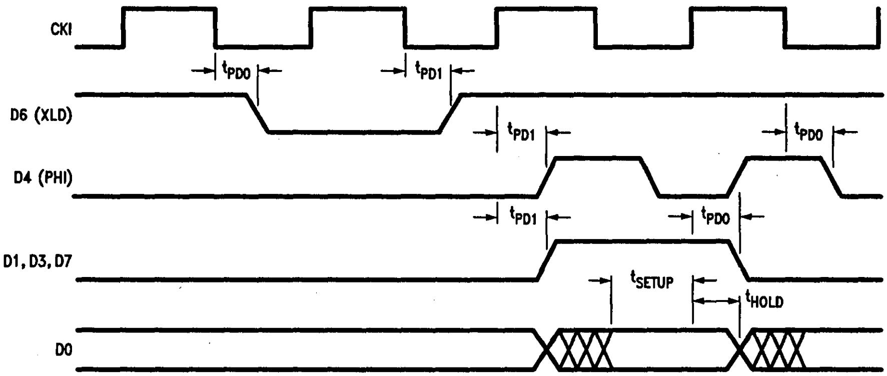
Note 8: For $V_{IN}(-) \ge V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

AC Electrical Characteristics $-40°C \leq T_A \leq +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator	$4V \leq V_{CC} \leq 6V$	1		DC	μs
	$2.5V \leq V_{CC} < 4V$	2.5		DC	μs
R/C Oscillator	$4V \leq V_{CC} \leq 6V$	3		DC	μs
	$2.5V \leq V_{CC} < 4V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 9)	f_r = Max	40		60	%
Rise Time (Note 9)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 9)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4V \leq V_{CC} \leq 6V$	200			ns
	$2.5V \leq V_{CC} < 4V$	500			ns
t_{HOLD}	$4V \leq V_{CC} \leq 6V$	60			ns
	$2.5V \leq V_{CC} < 4V$	150			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4V \leq V_{CC} \leq 6V$			0.7	μs
	$2.5V \leq V_{CC} < 4V$			1.75	μs
All Others	$4V \leq V_{CC} \leq 6V$			1	μs
	$2.5V \leq V_{CC} < 4V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

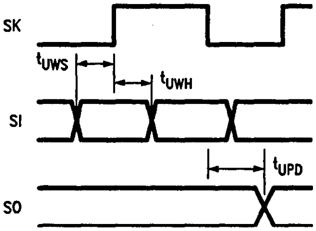
Note 9: Parameter sample but not 100% tested.

AC Electrical Characteristics (Continued)



TL/DD/9425-25

FIGURE 2a. AC Timing Diagrams in ROMless Mode



TL/DD/9425-26

FIGURE 2b. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

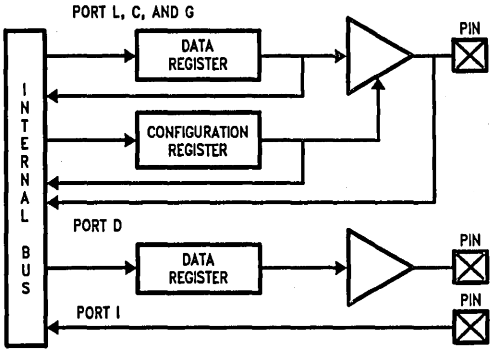
V_{REF} and AGND are the reference voltage pins for the on-board A/D converter.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset Description section.

The COP888CF contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CF memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CF. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/9425-6

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Port L supports Multi-Input Wakeup (MIWU) on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B. L0 and L1 are not available on the 44-pin version of the COP888CF, since they are replaced by V_{REF} and AGND. L0 and L1 are not terminated on the 44-pin version. Consequently, reading L0 or L1 as inputs will return unreliable data with the 44-pin package, so this data should be masked out with user software when the L port is read for input data. It is recommended that the pins be configured as outputs.

Port L has the following alternate features:

L0	MIWU
L1	MIWU
L2	MIWU
L3	MIWU
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU
L7	MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin, but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin or general purpose input (R/C clock configuration), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0	INTR (External Interrupt Input)
G2	T1B (Timer T1 Capture Input)
G3	T1A (Timer T1 I/O)
G4	SO (MICROWIRE™ Serial Data Output)
G5	SK (MICROWIRE Serial Clock)
G6	SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1	WDOUT WatchDog and/or Clock Monitor dedicated output
G7	CKO Oscillator dedicated output or general purpose input

Port I is an 8-bit Hi-Z input port, and also provides the analog inputs to the A/D converter. The 28-pin and 40 de-

Pin Descriptions (Continued)

vices do not have a full complement of Port I pins. The unavailable pins are not terminated (i.e. they are floating). A read operation from these unterminated pins will return unpredictable values. The user should ensure that the software takes this into account by either masking out these inputs, or else restricting the accesses to bit operations only. If unterminated, Port I pins will draw power only when addressed.

Port D is an 8-bit output port that is preset high when RESET goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CF is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CF architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_c) cycle time.

There are five CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)
PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CF consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CF vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X and SP pointers.

The COP888CF has 128 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, and B are memory mapped into this space at address locations 0FC to 0FE Hex respectively, with the other registers (other than reserved register 0FF) being available for general usage.

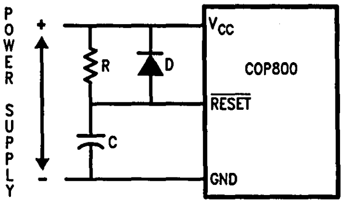
The instruction set of the COP888CF permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CF (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Reset

The $\overline{\text{RESET}}$ input when pulled low initializes the microcontroller. Initialization will occur whenever the $\overline{\text{RESET}}$ input is pulled low. Upon initialization, the data and configuration registers for Ports L, G, and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is initialized high with $\overline{\text{RESET}}$. The PC, PSW, CNTRL, ICNTRL, and T2CNTRL control registers are cleared. The Multi-Input Wakeup registers WKEN, WKEDG, and WKPND are cleared. The A/D control register ENAD is cleared, resulting in the ADC being powered down initially. The Stack Pointer, SP, is initialized to 06F Hex.

The COP888CF comes out of reset with both the WatchDog logic and the Clock Monitor detector armed, and with both the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor detector circuits are inhibited during reset. The WatchDog service window bits are initialized to the maximum WatchDog service window of 64k t_c clock cycles. The Clock Monitor bit is initialized high, and will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16–32 t_c clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 4* should be used to ensure that the $\overline{\text{RESET}}$ pin is held low until the power supply to the chip stabilizes.



TL/DD/9425-7

RC > 5 × Power Supply Rise Time

FIGURE 4. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.

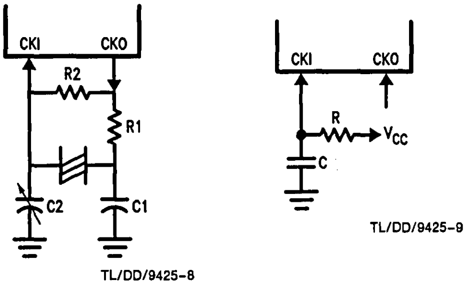


FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4	V_{CC} = 2.5V
0	1	200	100–150	0.455	V_{CC} = 2.5V

TABLE B. R/C Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8 to 2.2	3.6 to 4.5	V_{CC} = 5V
5.6	100	1.5 to 1.1	6.7 to 9	V_{CC} = 5V
6.8	100	1.1 to 0.8	9 to 12.5	V_{CC} = 2.5V

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. DC reference current contribution from the A/D converter—I6
7. Clock Monitor current when enabled—I7

Thus the total current drain, It, is given as

$$It = I1 + I2 + I3 + I4 + I5 + I6 + I7$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Some sample current drain values at V_{CC} = 5V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)	—	0.005

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

SL1 & SL0	Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
IEDG	External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
MSEL	Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
T1C0	Timer T1 Start/Stop control in timer modes 1 and 2 Timer T1 Underflow Interrupt Pending Flag in timer mode 3
T1C1	Timer T1 mode control bit
T1C2	Timer T1 mode control bit
T1C3	Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7							Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE	Global interrupt enable (enables interrupts)
EXEN	Enable external interrupt

Control Registers (Continued)

BUSY MICROWIRE/PLUS busy shifting flag

EXPND External interrupt pending

T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge

T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)

C Carry Flag

HC Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE

Bit 7 Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB Timer T1 Interrupt Enable for T1B Input capture edge

T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge

μWEN Enable MICROWIRE/PLUS interrupt

μWPND MICROWIRE/PLUS interrupt pending

T0EN Timer T0 Interrupt Enable (Bit 12 toggle)

T0PND Timer T0 Interrupt pending

LPEN L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PNDB	T1ENB

Bit 7 Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB Timer T2 Interrupt Enable for T2B Input capture edge

T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge

T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge

T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)

T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

T2C1 Timer T2 mode control bit

T2C2 Timer T2 mode control bit

T2C3 Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB

Bit 7 Bit 0

ROMless Mode

The COP888CF can address up to 32 kbytes of address space. If at power up, D2 is held at ground, the COP888CF executes from external memory. Port D is used to interface to external program memory. The address comes out in a serial fashion and the data from the external program memory is read back in a serial fashion. The Port D pins perform the following functions.

D0 Shifts in ROM data

D1 Shifts out lower eight bits of PC

D2 Places the μC in the ROMless mode if grounded at reset

D3 Shifts out upper eight bits of PC

D4 Data Shift Clock

D5 HALT Mask Option select pin
(D5 = 0) for HALT enable, D5 = 1 for HALT disable)

D6 Load Clock

D7 Shifts out recreated Port D data

The most significant bit of the data to come out on the D3 pin is a status signal.. It is used by the MOLE development system. This "lost" output port (D0–D7) can be accurately reconstructed with external components as shown in *Figure 6*, providing an accurate emulation.

The 44-pin and 40-pin versions of the COP888CF have a full complement of the D Port pins and can be used in the ROMless mode. However, it should be noted that the 44-pin device can only emulate itself and not the 40-pin or 28-pin devices as it has only 6 Port L pins while the other two devices have a full complement of Port L pins.

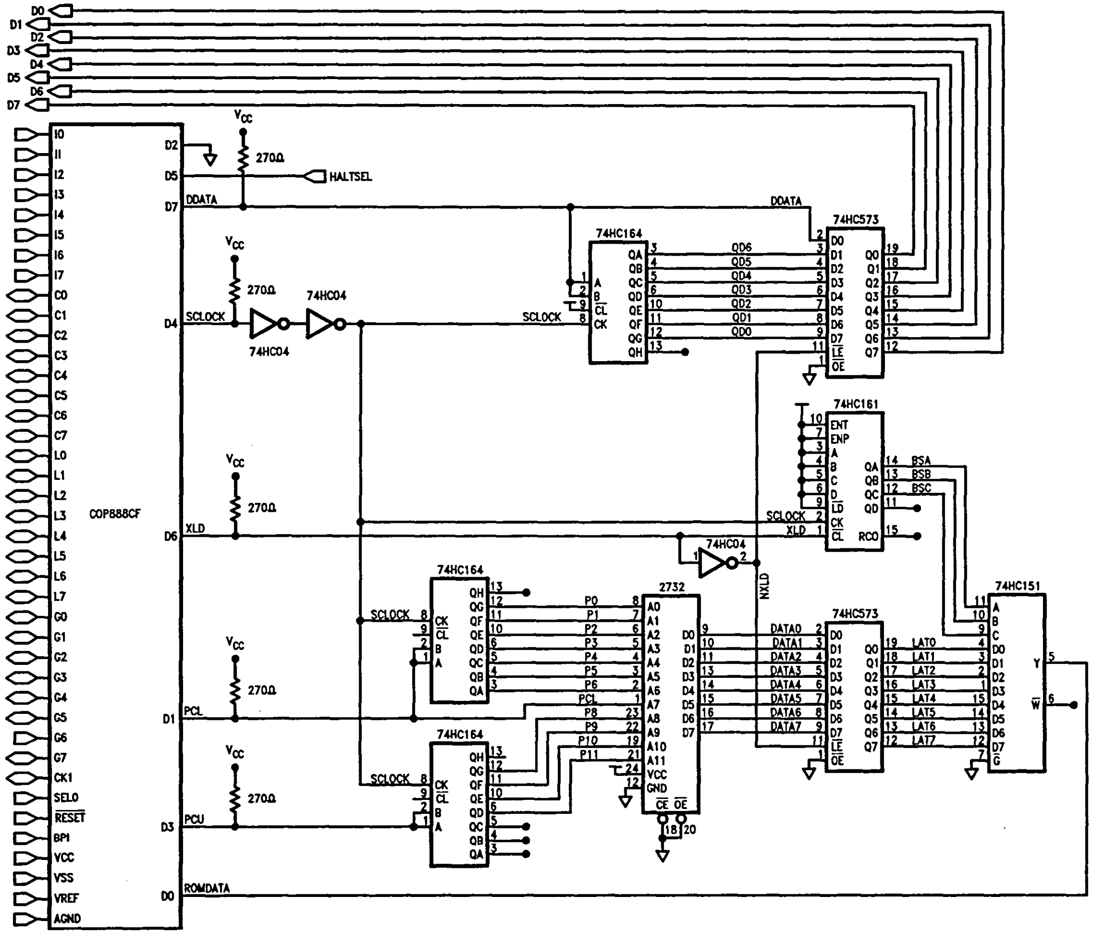
The 28-pin part cannot be used for emulation since it does not have the full complement of 8 D Port pins necessary for entering the ROMless mode.

Note that in the ROMless mode the D Port is recreated one full CKI clock cycle behind the normal port timings.

Note: Standard parts used in the ROMless mode will operate only at a reduced frequency (to be defined).

The COP888CF device has a spare D pin (D5) in the ROMless mode since only seven pins are required for emulation and recreation. This pin D5 is used in the ROMless mode to enable or disable the HALT mask option feature.

Figure 6 shows the COP888CF ROMless Mode Schematic.



TL/DD/9425–10

FIGURE 6. COP888CF ROMless Mode Schematic

Timers (Continued)

The COP888CF contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

Figure 7 shows a block diagram for the timers on the COP888CF.

TIMER T0 (IDLE TIMER)

The COP888CF supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_C. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

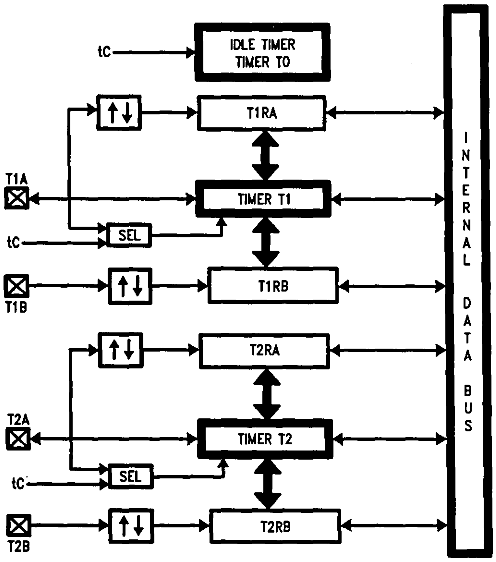
Exit out of the Idle Mode (See Idle Mode description)
WatchDog logic (See WatchDog description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_C = 1$ μs). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1 AND TIMER T2

The COP888CF has a set of two powerful timer/counter blocks, T1 and T2. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the two timer blocks, T1 and T2, are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CF to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.



TL/DD/9425-11

FIGURE 7. Timers for the COP888CF

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CF to generate a PWM signal with very minimal user intervention. The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

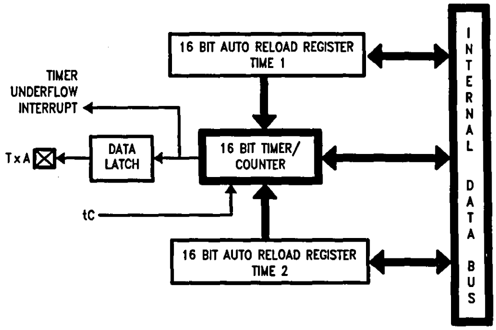
The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode.

The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.



TL/DD/9425-13

FIGURE 8. Timer In PWM Mode

Timers (Continued)

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

Mode 3. Input Capture Mode

The COP888CF can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_C rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

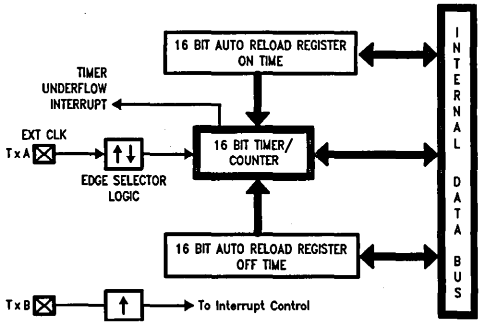
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

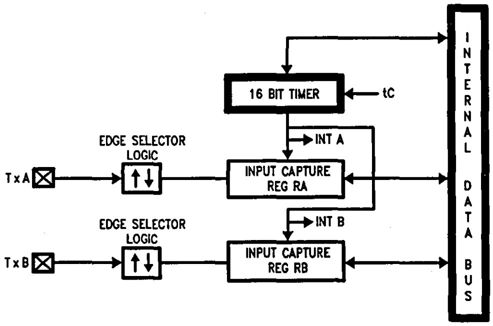
The timers T1 and T2 have indentical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control


16 BIT AUTO RELOAD REGISTER ON TIME
TIMER UNDERFLOW INTERRUPT
EXT CLK
TxA
EDGE SELECTOR LOGIC
16 BIT TIMER/ COUNTER
16 BIT AUTO RELOAD REGISTER OFF TIME
TxB
To Interrupt Control
INTERNAL DATA BUS

TL/DD/9425-14

FIGURE 9. Timer in External Event Counter Mode



TL/DD/9425-15

FIGURE 10. Timer in Input Capture Mode

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_c
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_c
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_c
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_c

Power Save Modes

The COP888CF offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CF is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock, timers, and A/D converter, are stopped. The WatchDog logic on the COP888CF is disabled during the HALT mode. However, the clock monitor circuitry if enabled remains active. In the HALT mode, the power requirements of the COP888CF are minimal and the applied voltage (V_{CC}) may be decreased to V_r (V_r = 2.0V) without altering the state of the machine.

The COP888CF supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the RESET pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_c instruction cycle clock. The t_c clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

The COP888CF has two mask options associated with the HALT mode. The first mask option enables the HALT mode

Power Save Modes (Continued)

feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CF will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CF cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the COP888CF inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CF is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activity, except the associated on-board oscillator circuitry, the WatchDog logic, the clock monitor and the IDLE Timer T0, is stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CF can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, $t_C = 1\ \mu s$) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the COP888CF will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CF will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is used to return (wakeup) the COP888CF from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 11 shows the Multi-Input Wakeup logic for the COP888CF microcontroller.

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CF to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

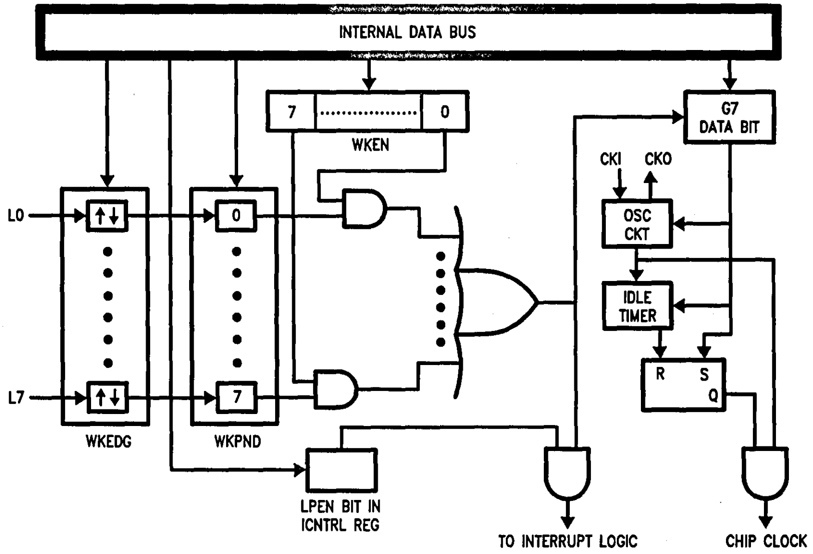
The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the as-

Multi-Input Wakeup (Continued)



TL/DD/9425-16

FIGURE 11. Multi-Input Wake Up Logic

sociated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CF will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

The WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG

Multi-Input Wakeup (Continued)

specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (global interrupt enable) bit enables the interrupt function. A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CF out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CF will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CF will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CF to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

A/D Converter

The COP888CF contains an 8-channel, multiplexed input, successive approximation, A/D converter. Two dedicated pins, V_{REF} and AGND are provided for voltage reference.

OPERATING MODES

The A/D converter supports ratiometric measurements. It supports both Single Ended and Differential modes of operation.

Four specific analog channel selection modes are supported. These are as follows:

Allow any specific channel to be selected at one time. The A/D converter performs the specific conversion requested and stops.

Allow any specific channel to be scanned continuously. In other words, the user will specify the channel and the A/D converter will keep on scanning it continuously. The user can come in at any arbitrary time and immediately read the result of the last conversion. The user does not have to wait for the current conversion to be completed.

Allow any differential channel pair to be selected at one time. The A/D converter performs the specific differential conversion requested and stops.

Allow any differential channel pair to be scanned continuously. In other words, the user will specify the differential channel pair and the A/D converter will keep on scanning it continuously. The user can come in at any arbitrary time and immediately read the result of the last differential conversion. The user does not have to wait for the current conversion to be completed.

The A/D converter is supported by two memory mapped registers, the result register and the mode control register. When the COP888CF is reset, the control register is cleared and the A/D is powered down. The A/D result register has unknown data following reset.

A/D Control Register

A control register, Reg: ENAD, contains 3 bits for channel selection, 3 bits for prescaler selection, and 2 bits for mode selection. An A/D conversion is initiated by writing to the ENAD control register. The result of the conversion is available to the user from the A/D result register, Reg: ADRSLT.

Reg: ENAD

CHANNEL SELECT	MODE SELECT	PRESCALER SELECT
Bits 7, 6, 5	Bits 4,3	Bits 2, 1, 0

CHANNEL SELECT

This 3-bit field selects one of eight channels to be the V_{IN+}. The mode selection determines the V_{IN-} input.

Single Ended mode:

Bit 7	Bit 6	Bit 5	Channel No.
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

A/D Converter (Continued)

Differential mode:

Bit 7	Bit 6	Bit 5	Channel Pairs (+. −)
0	0	0	0, 1
0	0	1	1, 0
0	1	0	2, 3
0	1	1	3, 2
1	0	0	4, 5
1	0	1	5, 4
1	1	0	6, 7
1	1	1	7, 6

MODE SELECT

This 2-bit field is used to select the mode of operation (single conversion, continuous conversions, differential, single ended) as shown in the following table.

Bit 4	Bit 3	Mode
0	0	Single Ended mode, single conversion
0	1	Single Ended mode, continuous scan of a single channel into the result register
1	0	Differential mode, single conversion
1	1	Differential mode, continuous scan of a channel pair into the result register

PRESCALER SELECT

This 3-bit field is used to select one of the seven prescaler clocks for the A/D converter. The prescaler also allows the A/D clock inhibit power saving mode to be selected. The following table shows the various prescaler options.

Bit 2	Bit 1	Bit 0	Clock Select
0	0	0	Inhibit A/D clock
0	0	1	Divide by 1
0	1	0	Divide by 2
0	1	1	Divide by 4
1	0	0	Divide by 6
1	0	1	Divide by 12
1	1	0	Divide by 8
1	1	1	Divide by 16

ADC Operation

The A/D converter interface works as follows. Writing to the A/D control register ENAD initiates an A/D conversion unless the prescaler value is set to 0, in which case the ADC clock is stopped and the ADC is powered down. The conversion sequence starts at the beginning of the write to ENAD operation powering up the ADC. At the first falling edge of the converter clock following the write operation (not counting the falling edge if it occurs at the same time as the write operation ends), the sample signal turns on for two clock cycles. The ADC is selected in the middle of the sample period. If the ADC is in single conversion mode, the conversion complete signal from the ADC will generate a power down for the A/D converter. If the ADC is in continuous mode, the conversion complete signal will restart the conversion sequence by deselecting the ADC for one converter clock cycle before starting the next sample. The ADC 8-bit result is loaded into the A/D result register (ADRSLT) except during LOAD clock high, which prevents transient data (resulting from the ADC writing a new result over an old one) being read from ADRSLT.

PRESCALER

The COP888CF A/D Converter (ADC) contains a prescaler option which allows seven different clock selections. The A/D clock frequency is equal to CKI divided by the prescaler value. Note that the prescaler value must be chosen such that the A/D clock falls within the specified range. The maximum A/D frequency is 1.67 MHz. This equates to a 600 ns ADC clock cycle.

The A/D converter takes 12 ADC clock cycles to complete a conversion. Thus the minimum ADC conversion time for the COP888CF is 7.2 μs when a prescaler of 6 has been selected. These 12 ADC clock cycles necessary for a conversion consist of 1 cycle at the beginning for reset, 2 cycles for sampling, 8 cycles for converting, and 1 cycle for loading the result into the COP888CF A/D result register (ADRSLT). This A/D result register is a read-only register. The COP888CF cannot write into ADRSLT.

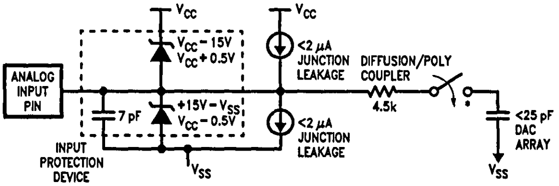
The prescaler also allows an A/D clock inhibit option, which saves power by powering down the A/D when it is not in use.

Note: The A/D converter is also powered down when the COP888CF is in either the HALT or IDLE modes. If the ADC is running when the COP888CF enters the HALT or IDLE modes, the ADC will power down during the HALT or IDLE, and then will reinitialize the conversion when the COP888CF comes out of the HALT or IDLE modes.

Analog Input and Source Resistance Considerations

Figure 12 shows the A/D pin model for the COP888CF in single ended mode. The differential mode has similiar A/D pin model. The leads to the analog inputs should be kept as short as possible. Both noise and digital clock coupling to an A/D input can cause conversion errors. The clock lead should be kept away from the analog input line to reduce coupling. The A/D channel input pins do not have any internal output driver circuitry connected to them because this circuitry would load the analog input signals due to output buffer leakage current.

Source impedances greater than 1 kΩ on the analog input lines will adversely affect internal RC charging time during input sampling. As shown in *Figure 12*, the analog switch to the DAC array is closed only during the 2 A/D cycle sample time. Large source impedances on the analog inputs may result in the DAC array not being charged to the correct voltage levels, causing scale errors.



TL/DD/9425-28

*The analog switch is closed only during the sample time.

FIGURE 12. A/D Pin Model (Single Ended Mode)

Interrupts (Continued)

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	Underflow	0yF8–0yF9
(4)	Timer T1	T1A/Underflow	0yF6–0yF7
(5)	Timer T1	T1B	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
	Reserved	for UART	0yEE–0yEF
	Reserved	for UART	0yEC–0yED
(7)	Timer T2	T2A/Underflow	0yEA–0yEB
(8)	Timer T2	T2B	0yE8–0yE9
	Reserved	for Future Use	0yE6–0yE7
	Reserved	for Future Use	0yE4–0yE5
(9)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(10) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page, $y \neq 0$

If large source resistance is necessary, the recommended solution is to slow down the A/D clock speed in proportion to the source resistance. The A/D converter may be operated at the maximum speed for R_S less than 1 kΩ. For R_S greater than 1 kΩ, A/D clock speed needs to be reduced. For example, with $R_S = 2$ kΩ, the A/D converter may be operated at half the maximum speed. A/D converter clock speed may be slowed down by either increasing the A/D prescaler divide-by or decreasing the CKI clock frequency. The A/D clock speed may be reduced to its minimum frequency of 100 kHz.

Interrupts

The COP888CF supports a vectored interrupt scheme. It supports a total of ten interrupt sources. The following table lists all the possible COP888CF interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

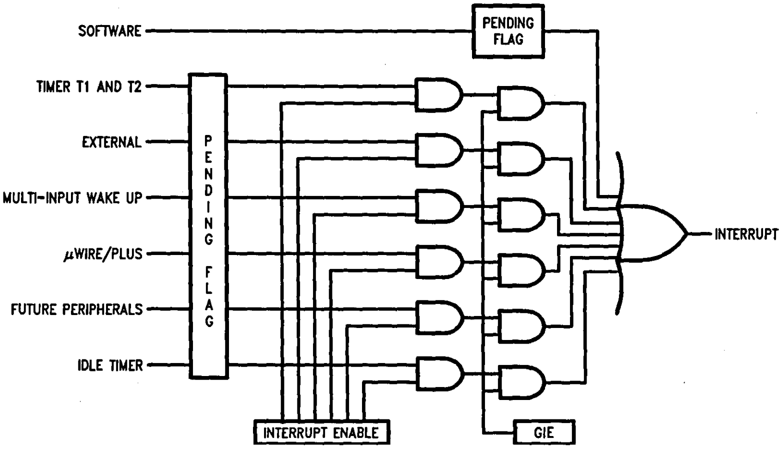
1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_c cycles to execute.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

Interrupts (Continued)



TL/DD/9425-18

FIGURE 13. COP888CF Interrupt Block Diagram

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block.

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 13 shows the COP888CF Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to RESET, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit **(not accessible by the user)** is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WatchDog

The COP888CF contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table I shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table II shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

WatchDog (Continued)

TABLE I. WatchDog Service Register

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE II. WatchDog Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k-8k t_c Cycles
0	1	2k-16k t_c Cycles
1	0	2k-32k t_c Cycles
1	1	2k-64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the COP888CF can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WatchDog Operation

The WatchDog and Clock Monitor are disabled during reset. The COP888CF comes out of reset with the WatchDog armed, the WatchDog Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WatchDog service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WatchDog service window and match the WatchDog key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WatchDog service window value and the key data (bits 7 through 1) in the WDSVR Register. Table III shows the sequence of events that can occur.

The user must service the WatchDog at least once before the upper limit of the service window expires. The WatchDog may not be serviced more than once in every lower limit of the service window. The user may service the WatchDog as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WatchDog service.

The WatchDog has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WatchDog, the logic will pull the WDOUT (G1) pin low for an additional 16 t_c–32 t_c cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CF will stop forcing the WDOUT output low.

The WatchDog service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WatchDog service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WatchDog will time out and WDOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_c–32 t_c clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10$ kHz—No clock rejection.

$1/t_c < 10$ Hz—Guaranteed clock rejection.

Detection of Illegal Conditions

The COP888CF can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location

TABLE III. WatchDog Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WatchDog Output
Mismatch	Don't Care	Don't Care	Error: Generate WatchDog Output
Don't Care	Don't Care	Mismatch	Error: Generate WatchDog Output

TABLE IV. MICROWIRE/PLUS Master Mode Clock Selection

SL1	SL0	SK
0	0	2 × t_c
0	1	4 × t_c
1	x	8 × t_c

Where t_c is the instruction cycle clock

Detection of Illegal Conditions

(Continued)

06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F Hex is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

a. Executing from undefined ROM

b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures).

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CF to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E^2PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 14* shows a block diagram of the MICROWIRE/PLUS logic.

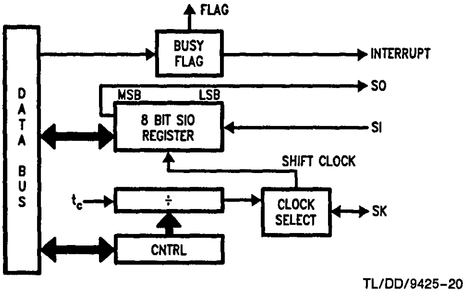


FIGURE 14. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table IV details the different clock rates that may be selected.

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The COP888CF may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 15* shows how two COP888CF microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock forthe SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CF. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table V summarizes the bit settings required for Master mode of operation.

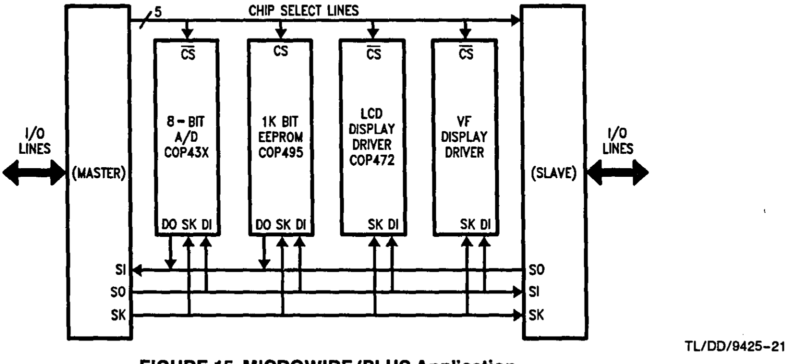


FIGURE 15. MICROWIRE/PLUS Application

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table V summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

The COP888CF allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock in the normal mode. In the alternate SK phase mode the SIO register is shifted on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE V

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space

Address	Contents
00 to 6F	On-Chip RAM bytes
70 to BF	Unused RAM Address Space
C0	Timer T2 Lower Byte
C1	Timer T2 Upper Byte
C2	Timer T2 Autoload Register T2RA Lower Byte
C3	Timer T2 Autoload Register T2RA Upper Byte
C4	Timer T2 Autoload Register T2RB Lower Byte
C5	Timer T2 Autoload Register T2RB Upper Byte
C6	Timer T2 Control Register
C7	WatchDog Service Register (Reg:WDSVR)
C8	MIWU Edge Select Register (Reg:WKEDG)
C9	MIWU Enable Register (Reg:WKEN)
CA	MIWU Pending Register (Reg:WKPND)
CB	A/D Converter Control Register (Reg:ENAD)
CC	A/D Converter Result Register (Reg: ADRSLT)
CD to CF	Reserved
D0	Port L Data Register
D1	Port L Configuration Register
D2	Port L Input Pins (Read Only)
D3	Reserved for Port L
D4	Port G Data Register
D5	Port G Configuration Register
D6	Port G Input Pins (Read Only)
D7	Port I Input Pins (Read Only)
D8	Port C Data Register
D9	Port C Configuration Register
DA	Port C Input Pins (Read Only)
DB	Reserved for Port C
DC	Port D Data Register
DD to DF	Reserved for Port D
E0 to E5	Reserved
E6	Timer T1 Autoload Register T1RB Lower Byte
E7	Timer T1 Autoload Register T1RB Upper Byte
E8	ICNTRL Register
E9	MICROWIRE Shift Register
EA	Timer T1 Lower Byte
EB	Timer T1 Upper Byte
EC	Timer T1 Autoload Register T1RA Lower Byte
ED	Timer T1 Autoload Register T1RA Upper Byte
EE	CNTRL Control Register
EF	PSW Register
F0 to FB	On-Chip RAM Mapped as Registers
FC	X Register
FD	SP Register
FE	B Register
FF	Reserved

Reading memory locations 70-7F Hex will return all ones. Reading other unused memory locations will return undefined data.

Addressing Modes

The COP888CF has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CF. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the LBI instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
MemI	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
⟷	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,Meml	ADD	A ← A + Meml
ADC	A,Meml	ADD with Carry	A ← A + Meml + C, C ← Carry
			HC ← Half Carry
SUBC	A,Meml	Subtract with Carry	A ← A $\overline{\text{Meml}}$ + C, C ← Carry
			HC ← Half Carry
AND	A,Meml	Logical AND	A ← A and Meml
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,Meml	Logical OR	A ← A or Meml
XOR	A,Meml	Logical EXclusive OR	A ← A xor Meml
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,Meml	IF EQual	Compare A and Meml, Do next if A = Meml
IFNE	A,Meml	IF Not Equal	Compare A and Meml, Do next if A ≠ Meml
IFGT	A,Meml	IF Greater Than	Compare A and Meml, Do next if A > Meml
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg ← Reg− 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A ⟷ Mem
X	A,[X]	EXchange A with Memory [X]	A ⟷ [X]
LD	A,Meml	LoaD A with Memory	A ← Meml
LD	A,[X]	LoaD A with Memory [X]	A ← [X]
LD	B,Imm	LoaD B with Immed.	B ← Imm
LD	Mem,Imm	LoaD Memory Immed	Mem ← Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg ← Imm
X	A, [B ±]	EXchange A with Memory [B]	A ⟷ [B], (B ← B ±1)
X	A, [X ±]	EXchange A with Memory [X]	A ⟷ [X], (X ← ±1)
LD	A, [B±]	LoaD A with Memory [B]	A ← [B], (B ← B ±1)
LD	A, [X±]	LoaD A with Memory [X]	A ← [X], (X ← X±1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] ← Imm, (B ← B±1)
CLR	A	CLeaR A	A ← 0
INC	A	INCrement A	A ← A + 1
DEC	A	DECrementA	A ← A − 1
LAID		Load A InDirect from ROM	A ← ROM (PU,A)
DCOR	A	Decimal CORrect A	A ← BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C → A7 → ... → A0 → C
RLC	A	Rotate A Left thru C	C ← A7 ← ... ← A0 ← C
SWAP	A	SWAP nibbles of A	A7...A4 ⟷ A3...A0
SC		Set C	C ← 1, HC ← 1
RC		Reset C	C ← 0, HC ← 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP ← SP + 1, A ← [SP]
PUSH	A	PUSH A onto the stack	[SP] ← A, SP ← SP − 1
VIS		Vector to Interrupt Service Routine	PU ← [VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9...0 ← i (i = 12 bits)
JP	Disp.	Jump relative short	PC ← PC + r (r is −31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] ← PL, [SP−1] ← PU,SP−2, PC ← ii
JSR	Addr	Jump SubRoutine	[SP] ← PL, [SP−1] ← PU,SP−2, PC9...0 ← i
JID		Jump InDirect	PL ← ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL ← [SP], PU ← [SP−1]
RETSK		RETurn and SKip	SP + 2, PL ← [SP],PU ← [SP−1]
RETI		RETurn from Interrupt	SP + 2, PL ← [SP],PU ← [SP−1],GIE ← 1
INTR		Generate an Interrupt	[SP] ← PL, [SP−1] ← PU, SP−2, PC ← 0FF
NOP		No OPeration	PC ← PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time (1 µs at 10 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 µs at 10 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B, Imm				1/1			(IF B < 16)
LD B, Imm				2/2			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

* = > Memory location addressed by B or X or directly.

COP888CF Opcode Table

Upper Nibble Along X-Axis
Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP −15	JP −31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	0
JP −14	JP −30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP −13	JP −29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	2
JP −12	JP −28	LD 0F3, # i	DRSZ 0F3	X A, [X−]	X A,[B−]	IFGT A,#i	IFGT A,[B]	3
JP −11	JP −27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	4
JP −10	JP −26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	5
JP −9	JP −25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	6
JP −8	JP −24	LD 0F7, # i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	7
JP −7	JP −23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	8
JP −6	JP −22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	9
JP −5	JP −21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+],#i	INCA	A
JP −4	JP −20	LD 0FB, # i	DRSZ 0FB	LD A,[X−]	LD A,[B−]	LD [B−],#i	DECA	B
JP −3	JP −19	LD 0FC, # i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	C
JP −2	JP −18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP −1	JP −17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B],#i	RET	E
JP −0	JP −16	LD 0FF, # i	DRSZ 0FF	*	*	LD B,#i	RETI	F

COP888CF Opcode Table (Continued)

Upper Nibble Along X-Axis

Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B,#0F	IFBNE 0	JSR x000–x0FF	JMP x000–x0FF	JP +17	INTR	0
IFBIT 1,[B]	*	LD B,#0E	IFBNE 1	JSR x100–x1FF	JMP x100–x1FF	JP +18	JP + 2	1
IFBIT 2,[B]	*	LD B,#0D	IFBNE 2	JSR x200–x2FF	JMP x200–x2FF	JP +19	JP + 3	2
IFBIT 3,[B]	*	LD B,#0C	IFBNE 3	JSR x300–x3FF	JMP x300–x3FF	JP +20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400–x4FF	JMP x400–x4FF	JP +21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500–x5FF	JMP x500–x5FF	JP +22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600–x6FF	JMP x600–x6FF	JP +23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700–x7FF	JMP x700–x7FF	JP +24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800–x8FF	JMP x800–x8FF	JP +25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900–x9FF	JMP x900–x9FF	JP +26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00–xAFF	JMP xA00–xAFF	JP +27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00–xBFF	JMP xB00–xBFF	JP +28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00–xCFF	JMP xC00–xCFF	JP +29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00–xDFF	JMP xD00–xDFF	JP +30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00–xEFF	JMP xE00–xEFF	JP +31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00–xFFF	JMP xF00–xFFF	JP +32	JP + 16	F

Where,

i is the immediate data
Md is a directly addressed memory location
* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CF mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

OPTION 1: CLOCK CONFIGURATION

= 1 Crystal Oscillator (CKI/10)
G7 (CKO) is clock generator output to crystal/resonator
CKI is the clock input

= 2 Single-pin RC controlled oscillator (CKI/10)
G7 is available as a HALT restart and/or general purpose input

OPTION 2: HALT

= 1 Enable HALT mode

= 2 Disable HALT mode

OPTION 3: COP888CF BONDING

= 1 44-Pin PLCC

= 2 40-Pin DIP

= 3 28-Pin PLCC

= 4 28-Pin DIP

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

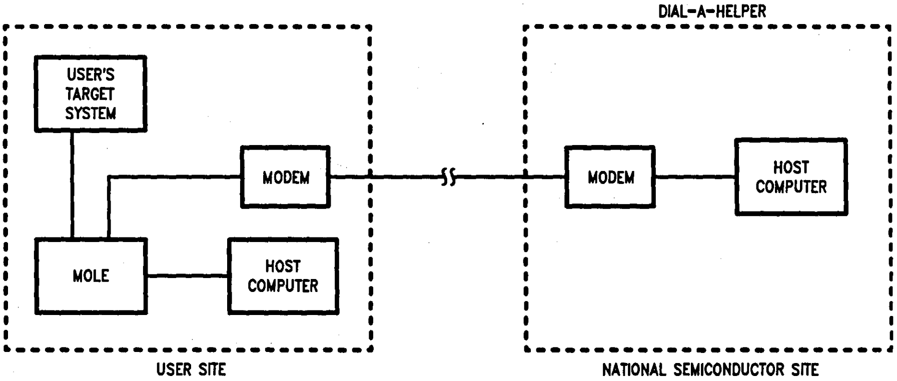
Order P/N: MOLE-DIAL-A-HLP

Information System Package Contents
- Dial-A-Helper User Manual
- Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 Baud	
	Set-Up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hours, 7 Days	



TL/DD/9425-27

National Semiconductor

PRELIMINARY

COP888CG Single-Chip microCMOS Microcontroller

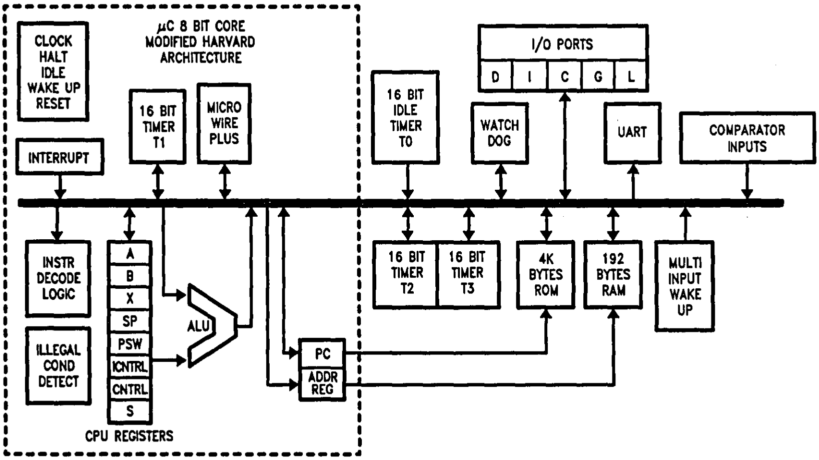
General Description

The COP888 family of microcontrollers uses an 8-bit single chip core architecture fabricated with National Semiconductor's M2CMOS™ process technology. The COP888CG is a member of this expandable 8-bit core processor family of microcontrollers. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 4096 bytes on-board ROM
- 192 bytes on-board RAM
- Single supply operation: 2.5V–6V
- Full duplex UART
- Two analog comparators
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Fourteen multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Three Timers (Each with 2 Interrupts)
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — UART (2)
  - — Default VIS
- Three 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC or 40 N or 28 N or 28 PCC
  - — 44 PCC with 39 I/O pins
  - — 40 N with 35 I/O pins
  - — 28 PCC or 28 N, each with 23 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Temperature ranges: −40°C to +85°C,
  −55°C to +125°C
- ROMless mode for accurate emulation and external program memory capability
- Single chip COP888CGP piggy back emulation device
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram



TL/DD/9765–1

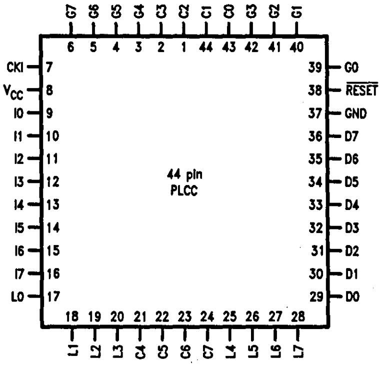
FIGURE 1. COP888CG Block Diagram

General Description (Continued)

It is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CG operates over a voltage range of 2.5V to 6V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate. The COP888CG may be operated in the ROMless mode to provide for accurate emulation and for applications requiring external program memory.

Connection Diagrams

Plastic Chip Carrier

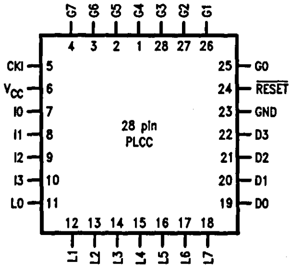


TL/DD/9765-2

Top View

Order Number COP888CG-XXX/V
See NS Plastic Chip Package Number V44A

Plastic Chip Carrier

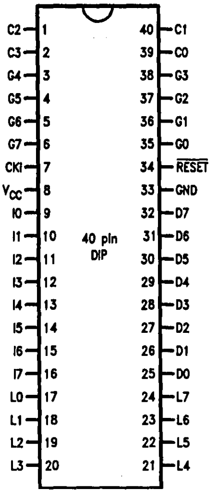


TL/DD/9765-3

Top View

Order Number COP884CG-XXX/V
See NS Plastic Chip Package Number V28A

Dual-In-Line Package

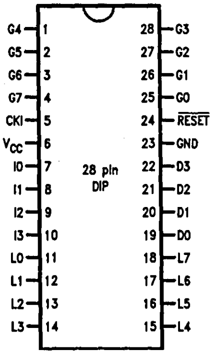


TL/DD/9765-4

Top View

Order Number COP888G-XXX/N
See NS Molded Package Number N40A

Dual-In-Line Package



TL/DD/9765-5

Top View

Order Number COP884CG-XXX/N
See NS Molded Package Number N28A

FIGURE 2a. COP888CG Connection Diagrams

Connection Diagrams (Continued)

COP888CG Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.	44-Pin Pack.
L0	I/O	MIWU		11	17	17
L1	I/O	MIWU	CKX	12	18	18
L2	I/O	MIWU	TDX	13	19	19
L3	I/O	MIWU	RDX	14	20	20
L4	I/O	MIWU	T2A	15	21	25
L5	I/O	MIWU	T2B	16	22	26
L6	I/O	MIWU	T3A	17	23	27
L7	I/O	MIWU	T3B	18	24	28
G0	I/O	INT		25	35	39
G1	WDOUT			26	36	40
G2	I/O	T1B		27	37	41
G3	I/O	T1A		28	38	42
G4	I/O	SO		1	3	3
G5	I/O	SK		2	4	4
G6	I	SI		3	5	5
G7	I/CKO	HALT Restart		4	6	6
D0	O	ROM DATA*		19	25	29
D1	O	PCL*		20	26	30
D2	O	EMUL*		21	27	31
D3	O	PCU*		22	28	32
I0	I			7	9	9
I1	I	COMP1IN−		8	10	10
I2	I	COMP1IN+		9	11	11
I3	I	COMP1OUT		10	12	12
I4	I	COMP2IN−			13	13
I5	I	COMP2IN+			14	14
I6	I	COMP2OUT			15	15
I7	I				16	16
D4	O	S CLOCK*			29	33
D5	O	HALTSEL*			30	34
D6	O	LOAD*			31	35
D7	O	D DATA*			32	36
C0	I/O				39	43
C1	I/O				40	44
C2	I/O				1	1
C3	I/O				2	2
C4	I/O					21
C5	I/O					22
C6	I/O					23
C7	I/O					24
V_{CC}				6	8	8
GND				23	33	37
CKI				5	7	7
$\overline{\text{RESET}}$				24	34	38

* = Only in the ROMless Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	7V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $-40°C \le T_A \le +85°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		2.5		6	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1\ \mu s$			15	mA
CKI = 4 MHz	$V_{CC} = 2.5V, t_c = 2.5\ \mu s$			2	mA
HALT Current (Note 3)	$V_{CC} = 6V$, CKI = 0 MHz		<1		μA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 6V, t_c = 1\ \mu s$			5	mA
CKI = 4 MHz	$V_{CC} = 2.5V, t_c = 2.5\ \mu s$			0.6	mA
Input Levels					
$\overline{RESET}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 6V$	−2		+2	μA
Input Pullup Current	$V_{CC} = 6V$	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink	$V_{CC} = 4V, V_{OL} = 1V$	10			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	2.0			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4V, V_{OH} = 2.7V$	10		100	μA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	2.5		33	μA
Source (Push-Pull Mode)	$V_{CC} = 4V, V_{OH} = 3.3V$	0.4			mA
	$V_{CC} = 2.5V, V_{OH} = 1.8V$	0.2			mA
Sink (Push-Pull Mode)	$V_{CC} = 4V, V_{OL} = 0.4V$	1.6			mA
	$V_{CC} = 2.5V, V_{OL} = 0.4V$	0.7			mA
TRI-STATE Leakage		−2		+2	μA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25^{\circ}C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $-40^{\circ}C \leq T_A \leq +85^{\circ}C$ unless otherwise specified

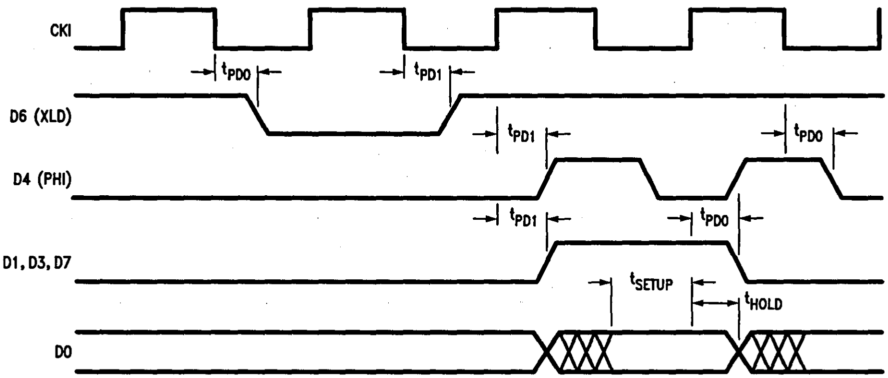
Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator,	$4V \leq V_{CC} \leq 6V$	1		DC	μs
	$2.5V \leq V_{CC} < 4V$	2.5		DC	μs
R/C Oscillator	$4V \leq V_{CC} \leq 6V$	3		DC	μs
	$2.5V \leq V_{CC} < 4V$	7.5		DC	μs
CKI Clock Duty Cycle (Note 5)	f_r = Max	40		60	%
Rise Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4V \leq V_{CC} \leq 6V$	200			ns
	$2.5V \leq V_{CC} < 4V$	500			ns
t_{HOLD}	$4V \leq V_{CC} \leq 6V$	60			ns
	$2.5V \leq V_{CC} < 4V$	150			ns
Output Propagation Delay	$R_L = 2.2k, C_L = 100\ pF$				
t_{PD1}, t_{PD0}					
SO, SK	$4V \leq V_{CC} \leq 6V$			0.7	μs
	$2.5V \leq V_{CC} < 4V$			1.75	μs
All Others	$4V \leq V_{CC} \leq 6V$			1	μs
	$2.5V \leq V_{CC} < 4V$			2.5	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

Note 5: Parameter sampled but not 100% tested.

Note 6: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

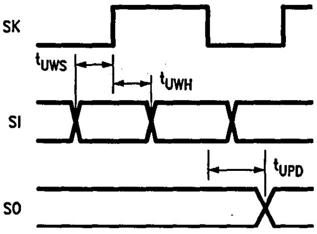
Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25°C$

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \le V_{IN} \le V_{CC} - 1.5V$		±10	±25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current Per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



TL/DD/9765-6

FIGURE 2b. AC Timing Diagrams in ROMless Mode



TL/DD/9765-7

FIGURE 2c. MICROWIRE/PLUS Timing

Pin Descriptions

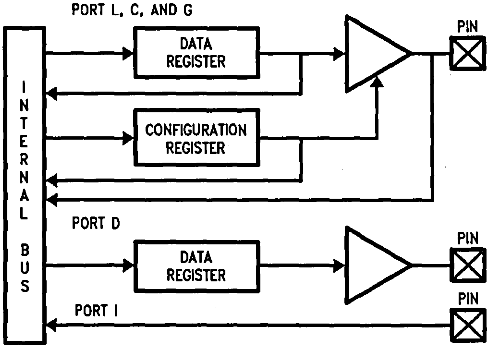
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset Description section.

The COP888CG contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CG memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CG. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/9765-8

FIGURE 3. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

L0	MIWU
L1	MIWU or CKX
L2	MIWU or TDX
L3	MIWU or RDX
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU or T3A
L7	MIWU or T3B

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition on G7. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0	INTR (External Interrupt Input)
G2	T1B (Timer T1 Capture Input)
G3	T1A (Timer T1 I/O)
G4	SO (MICROWIRE™ Serial Data Output)
G5	SK (MICROWIRE Serial Clock)
G6	SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1	WDOUT WatchDog and/or Clock Monitor dedicated output
G7	CKO Oscillator dedicated output or general purpose input

PORT I is an eight-bit Hi-Z input port. The 28-pin device does not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the

Pin Descriptions (Continued)

accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

I1	COMP1−IN (Comparator 1 Negative Input)
I2	COMP1+IN (Comparator 1 Positive Input)
I3	COMP1OUT (Comparator 1 Output)
I4	COMP2−IN (Comparator 2 Negative Input)
I5	COMP2+IN (Comparator 2 Positive Input)
I6	COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when $\overline{\text{RESET}}$ goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Functional Description

The architecture of the COP888CG is modified Harvard architecture. With the Harvard architecture, the control store program memory (ROM) is separated from the data store memory (RAM). Both ROM and RAM have their own separate addressing space with separate address buses. The COP888CG architecture, though based on Harvard architecture, permits transfer of data from ROM to RAM.

CPU REGISTERS

The CPU can do an 8-bit addition, subtraction, logical or shift operation in one instruction (t_C) cycle time.

There are six CPU registers:

A is the 8-bit Accumulator Register

PC is the 15-bit Program Counter Register

PU is the upper 7 bits of the program counter (PC)
PL is the lower 8 bits of the program counter (PC)

B is an 8-bit RAM address pointer, which can be optionally post auto incremented or decremented.

X is an 8-bit alternate RAM address pointer, which can be optionally post auto incremented or decremented.

SP is the 8-bit stack pointer, which points to the subroutine/interrupt stack (in RAM). The SP is initialized to RAM address 06F with reset.

S is the 8-bit Data Segment Address Register used to extend the lower half of the address range (00 to 7F) into 256 data segments of 128 bytes each.

All the CPU registers are memory mapped with the exception of the Accumulator (A) and the Program Counter (PC).

PROGRAM MEMORY

Program memory for the COP888CG consists of 4096 bytes of ROM. These bytes may hold program instructions or constant data (data tables for the LAID instruction, jump vectors for the JID instruction, and interrupt vectors for the VIS instruction). The program memory is addressed by the 15-bit program counter (PC). All interrupts in the COP888CG vector to program memory location 0FF Hex.

DATA MEMORY

The data memory address space includes the on-chip RAM and data registers, the I/O registers (Configuration, Data and Pin), the control registers, the MICROWIRE/PLUS SIO shift register, and the various registers, and counters associated with the timers (with the exception of the IDLE timer). Data memory is addressed directly by the instruction or indirectly by the B, X, SP pointers and S register.

The COP888CG has 192 bytes of RAM. Sixteen bytes of RAM are mapped as "registers" at addresses 0F0 to 0FF Hex. These registers can be loaded immediately, and also decremented and tested with the DRSZ (decrement register and skip if zero) instruction. The memory pointer registers X, SP, B and S are memory mapped into this space at address locations 0FC to 0FF Hex respectively, with the other registers being available for general usage.

The instruction set of the COP888CG permits any bit in memory to be set, reset or tested. All I/O and registers on the COP888CG (except A and PC) are memory mapped; therefore, I/O bits and register bits can be directly and individually set, reset and tested. The accumulator (A) bits can also be directly and individually tested.

Data Memory Segment RAM Extension

Data memory address 0FF is used as a memory mapped location for the Data Segment Address Register (S) in the COP888CG.

The data store memory is either addressed directly by a single byte address within the instruction, or indirectly relative to the reference of the B, X, or SP pointers (each contains a single-byte address). This single-byte address allows an addressing range of 256 locations from 00 to FF hex. The upper bit of this single-byte address divides the data store memory into two separate sections as outlined previously. With the exception of the RAM register memory from address locations 00F0 to 00FF, all RAM memory is memory mapped with the upper bit of the single-byte address being equal to zero. This allows the upper bit of the single-byte address to determine whether or not the base address range (from 0000 to 00FF) is extended. If this upper bit equals one (representing address range 0080 to 00FF), then address extension does not take place. Alternatively, if this upper bit equals zero, then the data segment extension register S is used to extend the base address range (from 0000 to 007F) from XX00 to XX7F, where XX represents the 8 bits from the S register. Thus the 128-byte data segment extensions are located from addresses 0100 to 017F for data segment 1, 0200 to 027F for data segment 2, etc., up to FF00 to FF7F for data segment 255. The base address range from 0000 to 007F represents data segment 0.

Figure 4 illustrates how the S register data memory extension is used in extending the lower half of the base address range (00 to 7F hex) into 256 data segments of 128 bytes each, with a total addressing range of 32 kbytes from XX00 to XX7F. This organization allows a total of 256 data segments of 128 bytes each with an additional upper base segment of 128 bytes. Furthermore, all addressing modes are available for all data segments. The S register must be changed under program control to move from one data segment (128 bytes) to another. However, the upper base segment (containing the 16 memory registers, I/O registers, control registers, etc.) is always available regardless of the

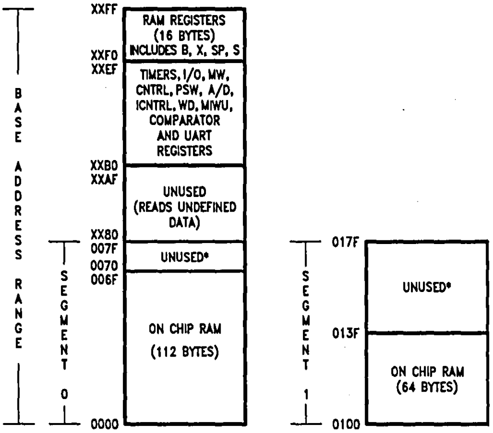
Data Memory Segment RAM Extension (Continued)

contents of the S register, since the upper base segment (address range 0080 to 00FF) is independent of data segment extension.

The instructions that utilize the stack pointer (SP) always reference the stack as part of the base segment (Segment 0), regardless of the contents of the S register. The S register is not changed by these instructions. Consequently, the stack (used with subroutine linkage and interrupts) is always located in the base segment. The stack pointer will be intitialized to point at data memory location 006F as a result of reset.

The 128 bytes of RAM contained in the base segment are split between the lower and upper base segments. The first 116 bytes of RAM are resident from address 0000 to 006F in the lower base segment, while the remaining 16 bytes of RAM represent the 16 data memory registers located at addresses 00F0 to 00FF of the upper base segment. No RAM is located at the upper sixteen addresses (0070 to 007F) of the lower base segment.

Additional RAM beyond these initial 128 bytes, however, will always be memory mapped in groups of 128 bytes (or less) at the data segment address extensions (XX00 to XX7F) of the lower base segment. The additional 64 bytes of RAM in the COP888CG (beyond the initial 128 bytes) are memory mapped at address locations 0100 to 013F hex.


XXFF
RAM REGISTERS (16 BYTES) INCLUDES B, X, SP, S
XXF0
XXEF
TIMERS, I/O, MW, CNTRL, PSW, A/D, ICNTRL, WD, MIWU, COMPARATOR AND UART REGISTERS
XXB0
XXAF
UNUSED (READS UNDEFINED DATA)
XX80
007F
0070
006F
UNUSED*
ON CHIP RAM (112 BYTES)
0000
BASE ADDRESS RANGE
SEGMENT 0
SEGMENT 1
017F
UNUSED*
013F
ON CHIP RAM (64 BYTES)
0100

TL/DD/9765-9

*Reads as all ones.

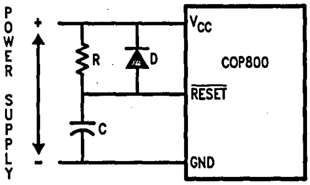
FIGURE 4. RAM Organization

Reset

The RESET input when pulled low initializes the microcontroller. Initialization will occur whenever the RESET input is pulled low. Upon initialization, the data and configuration registers for ports L, G and C are cleared, resulting in these Ports being initialized to the TRI-STATE mode. Pin G1 of the G Port is an exception (as noted below) since pin G1 is dedicated as the WatchDog and/or Clock Monitor error output pin. Port D is set high. The PC, PSW, ICNTRL, CNTRL, T2CNTRL and T3CNTRL control registers are cleared. The UART registers PSR, ENU (except that TBMT bit is set), ENUR and ENUI are cleared. The Comparator Select Register is cleared. The S register is initialized to zero. The Multi-Input Wakeup registers WKEN, WKEDG and WKPND are cleared. The stack pointer, SP, is initialized to 6F Hex.

The COP888CG comes out of reset with both the WatchDog logic and the Clock Monitor detector armed, with the WatchDog service window bits set and the Clock Monitor bit set. The WatchDog and Clock Monitor circuits are inhibited during reset. The WatchDog service window bits being initialized high default to the maximum WatchDog service window of 64k t_C clock cycles. The Clock Monitor bit being initialized high will cause a Clock Monitor error following reset if the clock has not reached the minimum specified frequency at the termination of reset. A Clock Monitor error will cause an active low error output on pin G1. This error output will continue until 16 t_C–32 t_C clock cycles following the clock frequency reaching the minimum specified value, at which time the G1 output will enter the TRI-STATE mode.

The external RC network shown in *Figure 5* should be used to ensure that the RESET pin is held low until the power supply to the chip stabilizes.



TL/DD/9765-10

RC > 5 × Power Supply Rise Time

FIGURE 5. Recommended Reset Circuit

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Figure 6 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

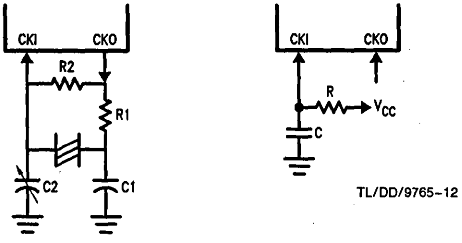
Table A shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table B shows the variation in the oscillator frequencies as functions of the component (R and C) values.

Oscillator Circuits (Continued)



TL/DD/9765-11

TL/DD/9765-12

FIGURE 6. Crystal and R/C Oscillator Diagrams

TABLE A. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30–36	4	V_{CC} = 2.5V
0	1	200	100–150	0.455	V_{CC} = 2.5V

TABLE B. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8 to 2.2	3.6 to 4.5	V_{CC} = 5V
5.6	100	1.5 to 1.1	6.7 to 9	V_{CC} = 5V
6.8	100	1.1 to 0.8	9 to 12.5	V_{CC} = 2.5V

Current Drain

The total current drain of the chip depends on:

1. Oscillator operation mode—I1
2. Internal switching current—I2
3. Internal leakage current—I3
4. Output source current—I4
5. DC current caused by external input not at V_{CC} or GND—I5
6. Comparator DC supply current when enabled—I6
7. Clock Monitor current when enabled—I7

Thus the total current drain, It, is given as

$$It = I1 + I2 + I3 + I4 + I5 + I6 + I7$$

To reduce the total current drain, each of the above components must be minimum.

The chip will draw more current as the CKI input frequency increases up to the maximum 10 MHz value. Operating with a crystal network will draw more current than an external square-wave. Switching current, governed by the equation below, can be reduced by lowering voltage and frequency. Leakage current can be reduced by lowering voltage and temperature. The other two items can be reduced by carefully designing the end-user's system.

$$I2 = C \times V \times f$$

where C = equivalent capacitance of the chip
V = operating voltage
f = CKI frequency

Some sample current drain values at V_{CC} = 5V are:

CKI (MHz)	Inst. Cycle (μs)	It (mA)
10	1	15
3.58	2.8	5.4
2	5	3
0.3	33	0.45
0 (HALT)	—	0.005

Control Registers

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE/PLUS control register contains the following bits:

SL1 & SL0 Select the MICROWIRE/PLUS clock divide by (00 = 2, 01 = 4, 1x = 8)
IEDG External interrupt edge polarity select (0 = Rising edge, 1 = Falling edge)
MSEL Selects G5 and G4 as MICROWIRE/PLUS signals SK and SO respectively
T1C0 Timer T1 Start/Stop control in timer modes 1 and 2
Timer T1 Underflow Interrupt Pending Flag in timer mode 3
T1C1 Timer T1 mode control bit
T1C2 Timer T1 mode control bit
T1C3 Timer T1 mode control bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0

Bit 7 Bit 0

PSW Register (Address X'00EF)

The PSW register contains the following select bits:

GIE Global interrupt enable (enables interrupts)
EXEN Enable external interrupt
BUSY MICROWIRE/PLUS busy shifting flag
EXPND External interrupt pending
T1ENA Timer T1 Interrupt Enable for Timer Underflow or T1A Input capture edge
T1PNDA Timer T1 Interrupt Pending Flag (Autoreload RA in mode 1, T1 Underflow in Mode 2, T1A capture edge in mode 3)
C Carry Flag
HC Half Carry Flag

HC	C	T1PNDA	T1ENA	EXPND	BUSY	EXEN	GIE

Bit 7 Bit 0

The Half-Carry bit is also affected by all the instructions that affect the Carry flag. The SC (Set Carry) and RC (Reset Carry) instructions will respectively set or clear both the carry flags. In addition to the SC and RC instructions, ADC, SUBC, RRC and RLC instructions affect the carry and Half Carry flags.

Control Registers (Continued)

ICNTRL Register (Address X'00E8)

The ICNTRL register contains the following bits:

T1ENB Timer T1 Interrupt Enable for T1B Input capture edge

T1PNDB Timer T1 Interrupt Pending Flag for T1B capture edge

μWEN Enable MICROWIRE/PLUS interrupt

μWPND MICROWIRE/PLUS interrupt pending

T0EN Timer T0 Interrupt Enable (Bit 12 toggle)

T0PND Timer T0 Interrupt pending

LPEN L Port Interrupt Enable (Multi-Input Wakeup/Interrupt)

Bit 7 could be used as a flag

Unused	LPEN	T0PND	T0EN	μWPND	μWEN	T1PNDB	T1ENB
Bit 7							Bit 0

T2CNTRL Register (Address X'00C6)

The T2CNTRL register contains the following bits:

T2ENB Timer T2 Interrupt Enable for T2B Input capture edge

T2PNDB Timer T2 Interrupt Pending Flag for T2B capture edge

T2ENA Timer T2 Interrupt Enable for Timer Underflow or T2A Input capture edge

T2PNDA Timer T2 Interrupt Pending Flag (Autoreload RA in mode 1, T2 Underflow in mode 2, T2A capture edge in mode 3)

T2C0 Timer T2 Start/Stop control in timer modes 1 and 2 Timer T2 Underflow Interrupt Pending Flag in timer mode 3

T2C1 Timer T2 mode control bit

T2C2 Timer T2 mode control bit

T2C3 Timer T2 mode control bit

T2C3	T2C2	T2C1	T2C0	T2PNDA	T2ENA	T2PNDB	T2ENB
Bit 7							Bit 0

T3CNTRL Register (Address X'00B6)

The T3CNTRL register contains the following bits:

T3ENB Timer T3 Interrupt Enable for T3B

T3PNDB Timer T3 Interrupt Pending Flag for T3B pin (T3B capture edge)

T3ENA Timer T3 Interrupt Enable for Timer Underflow or T3A pin

T3PNDA Timer T3 Interrupt Pending Flag (Autoload RA in mode 1, T3 Underflow in mode 2, T3a capture edge in mode 3)

T3C0 Timer T3 Start/Stop control in timer modes 1 and 2

Timer T3 Underflow Interrupt Pending Flag in timer mode 3

T3C1 Timer T3 mode control bit

T3C2 Timer T3 mode control bit

T3C3 Timer T3 mode control bit

T3C3	T3C2	T3C1	T3C0	T3PNDA	T3ENA	T3PNDB	T3ENB
Bit 7							Bit 0

ROMless Mode

The COP888CG can address up to 32 kbytes of address space. If at power up, D2 is held at ground, the COP888CG executes from external memory. Port D is used to interface to external program memory. The address comes out in a serial fashion and the data from the external program memory is read back in a serial fashion. The Port D pins perform the following functions.

D0 Shifts in ROM data

D1 Shifts out lower eight bits of PC

D2 Places the μC in the ROMless mode if grounded at reset

D3 Shifts out upper eight bits of PC

D4 Data Shift Clock

D5 HALT Mask Option select pin
(D5 = 0) for HALT enable, D5 = 1 for HALT disable)

D6 Load Clock

D7 Shifts out recreated Port D data

The most significant bit of the data to come out on the D3 pin is a status signal.. It is used by the MOLE development system. This "lost" output port (D0–D7) can be accurately reconstructed with external components as shown in *Figure 6*, providing an accurate emulation.

The 44-pin and 40-pin versions of the COP888CG have a full complement of the D Port pins and can be used in the ROMless mode.

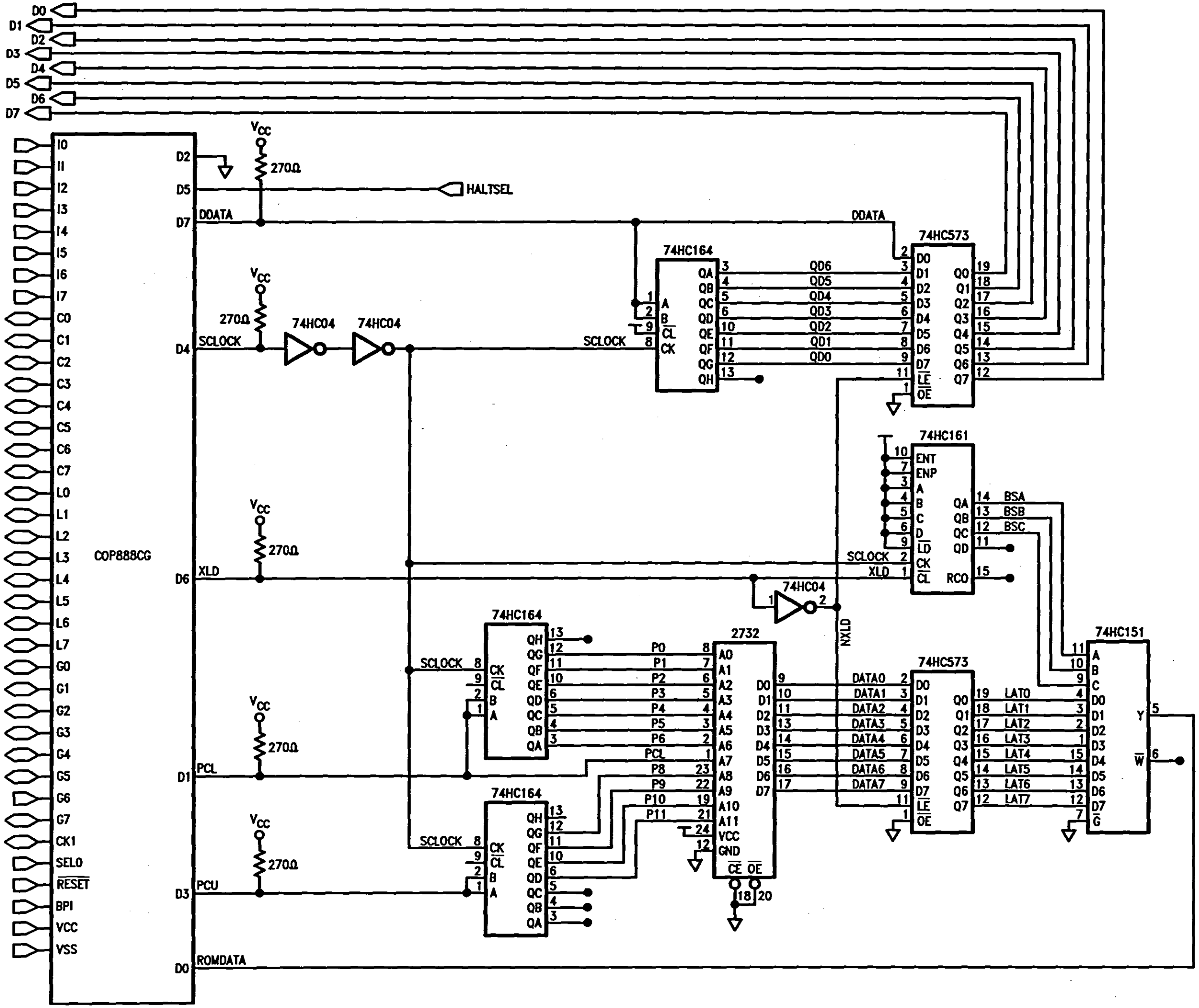
The 28-pin part cannot be used for emulation since it does not have the full complement of 8 D Port pins necessary for entering the ROMless mode.

Note that in the ROMless mode the D Port is recreated one full CKI clock cycle behind the normal port timings.

Note: Standard parts used in the ROMless mode will operate only at a reduced frequency (to be defined).

The COP888CG device has a spare D pin (D5) in the ROMless mode since only seven pins are required for emulation and recreation. This pin D5 is used in the ROMless mode to enable or disable the HALT mask option feature.

Figure 7 shows the COP888CG ROMless Mode Schematic.



TL/DD/9765–13

FIGURE 7. COP888CG ROMless Mode Schematic

Timers

The COP888CG contains a very versatile set of timers (T0, T1, T2). All timers and associated autoreload/capture registers power up containing random data.

TIMER T0 (IDLE TIMER)

The COP888CG supports applications that require maintaining real time and low power with the IDLE mode. This IDLE mode support is furnished by the IDLE timer T0, which is a 16-bit timer. The Timer T0 runs continuously at the fixed rate of the instruction cycle clock, t_C. The user cannot read or write to the IDLE Timer T0, which is a count down timer.

The Timer T0 supports the following functions:

Exit out of the Idle Mode (See Idle Mode description)
WatchDog logic (See WatchDog description)
Start up delay out of the HALT mode

The IDLE Timer T0 can generate an interrupt when the thirteenth bit toggles. This toggle is latched into the T0PND pending flag, and will occur every 4 ms at the maximum clock frequency ($t_C = 1\ \mu s$). A control flag T0EN allows the interrupt from the thirteenth bit of Timer T0 to be enabled or disabled. Setting T0EN will enable the interrupt, while resetting it will disable the interrupt.

TIMER T1, TIMER T2 AND TIMER T3

The COP888CG has a set of two powerful timer/counter blocks, T1, T2 and T3. The associated features and functioning of a timer block are described by referring to the timer block Tx. Since the three timer blocks, T1, T2 and T3 are identical, all comments are equally applicable to either timer block.

Each timer block consists of a 16-bit timer, Tx, and two supporting 16-bit autoreload/capture registers, RxA and RxB. Each timer block has two pins associated with it, TxA and TxB. The pin TxA supports I/O required by the timer block, while the pin TxB is an input to the timer block. The powerful and flexible timer block allows the COP888CG to easily perform all timer functions with minimal software overhead. The timer block has three operating modes: Processor Independent PWM mode, External Event Counter mode, and Input Capture mode.

The control bits TxC3, TxC2, and TxC1 allow selection of the different modes of operation.

Mode 1. Processor Independent PWM Mode

As the name suggests, this mode allows the COP888CG to generate a PWM signal with very minimal user intervention.

The user only has to define the parameters of the PWM signal (ON time and OFF time). Once begun, the timer block will continuously generate the PWM signal completely independent of the microcontroller. The user software services the timer block only when the PWM parameters require updating.

In this mode the timer Tx counts down at a fixed rate of t_C. Upon every underflow the timer is alternately reloaded with the contents of supporting registers, RxA and RxB. The very first underflow of the timer causes the timer to reload from the register RxA. Subsequent underflows cause the timer to be reloaded from the registers alternately beginning with the register RxB.

The Tx Timer control bits, TxC3, TxC2 and TxC1 set up the timer for PWM mode operation.

Figure 8 shows a block diagram of the timer in PWM mode.

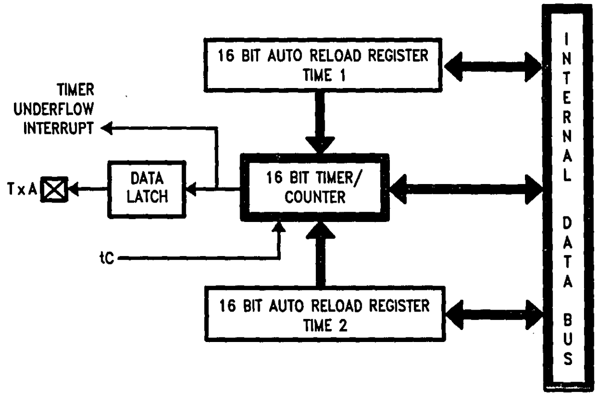
The underflows can be programmed to toggle the TxA output pin. The underflows can also be programmed to generate interrupts.

Underflows from the timer are alternately latched into two pending flags, TxPNDA and TxPNDB. The user must reset these pending flags under software control. Two control enable flags, TxENA and TxENB, allow the interrupts from the timer underflow to be enabled or disabled. Setting the timer enable flag TxENA will cause an interrupt when a timer underflow causes the RxA register to be reloaded into the timer. Setting the timer enable flag TxENB will cause an interrupt when a timer underflow causes the RxB register to be reloaded into the timer. Resetting the timer enable flags will disable the associated interrupts.

Either or both of the timer underflow interrupts may be enabled. This gives the user the flexibility of interrupting once per PWM period on either the rising or falling edge of the PWM output. Alternatively, the user may choose to interrupt on both edges of the PWM output.

Mode 2. External Event Counter Mode

This mode is quite similar to the processor independent PWM mode described above. The main difference is that the timer, Tx, is clocked by the input signal from the TxA pin. The Tx timer control bits, TxC3, TxC2 and TxC1 allow the timer to be clocked either on a positive or negative edge from the TxA pin. Underflows from the timer are latched into the TxPNDA pending flag. Setting the TxENA control flag will cause an interrupt when the timer underflows.



TL/DD/9765-14

FIGURE 8. Timer in PWM Mode

Timers (Continued)

In this mode the input pin TxB can be used as an independent positive edge sensitive interrupt input if the TxENB control flag is set. The occurrence of a positive edge on the TxB input pin is latched into the TxPNDB flag.

Figure 9 shows a block diagram of the timer in External Event Counter mode.

Note: The PWM output is not available in this mode since the TxA pin is being used as the counter input clock.

Mode 3. Input Capture Mode

The COP888CG can precisely measure external frequencies or time external events by placing the timer block, Tx, in the input capture mode.

In this mode, the timer Tx is constantly running at the fixed t_C rate. The two registers, RxA and RxB, act as capture registers. Each register acts in conjunction with a pin. The register RxA acts in conjunction with the TxA pin and the register RxB acts in conjunction with the TxB pin.

The timer value gets copied over into the register when a trigger event occurs on its corresponding pin. Control bits, TxC3, TxC2 and TxC1, allow the trigger events to be specified either as a positive or a negative edge. The trigger condition for each input pin can be specified independently.

The trigger conditions can also be programmed to generate interrupts. The occurrence of the specified trigger condition on the TxA and TxB pins will be respectively latched into the pending flags, TxPNDA and TxPNDB. The control flag TxENA allows the interrupt on TxA to be either enabled or disabled. Setting the TxENA flag enables interrupts to be generated when the selected trigger condition occurs on the TxA pin. Similarly, the flag TxENB controls the interrupts from the TxB pin.

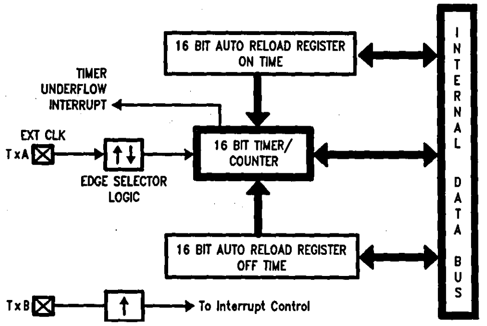
Underflows from the timer can also be programmed to generate interrupts. Underflows are latched into the timer TxC0 pending flag (the TxC0 control bit serves as the timer underflow interrupt pending flag in the Input Capture mode). Consequently, the TxC0 control bit should be reset when entering the Input Capture mode. The timer underflow interrupt is enabled with the TxENA control flag. When a TxA interrupt occurs in the Input Capture mode, the user must check both the TxPNDA and TxC0 pending flags in order to determine whether a TxA input capture or a timer underflow (or both) caused the interrupt.

Figure 10 shows a block diagram of the timer in Input Capture mode.

TIMER CONTROL FLAGS

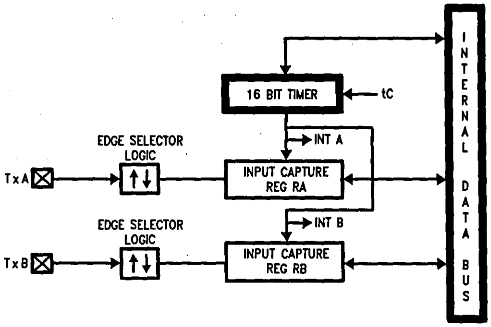
The timers T1, T2 and T3 have indentical control structures. The control bits and their functions are summarized below.

TxC0	Timer Start/Stop control in Modes 1 and 2 (Processor Independent PWM and External Event Counter), where 1 = Start, 0 = Stop Timer Underflow Interrupt Pending Flag in Mode 3 (Input Capture)
TxPNDA	Timer Interrupt Pending Flag
TxPNDB	Timer Interrupt Pending Flag
TxENA	Timer Interrupt Enable Flag
TxENB	Timer Interrupt Enable Flag 1 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled
TxC3	Timer mode control
TxC2	Timer mode control
TxC1	Timer mode control


16 BIT AUTO RELOAD REGISTER ON TIME
TIMER UNDERFLOW INTERRUPT
EXT CLK
TxA
16 BIT TIMER/ COUNTER
EDGE SELECTOR LOGIC
16 BIT AUTO RELOAD REGISTER OFF TIME
TxB
To Interrupt Control
INTERNAL DATA BUS

TL/DD/9765-15

FIGURE 9. Timer in External Event Counter Mode



TL/DD/9765-16

FIGURE 10. Timer in Input Capture Mode

Timers (Continued)

The timer mode control bits (TxC3, TxC2 and TxC1) are detailed below:

TxC3	TxC2	TxC1	Timer Mode	Interrupt A Source	Interrupt B Source	Timer Counts On
0	0	0	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Pos. Edge
0	0	1	MODE 2 (External Event Counter)	Timer Underflow	Pos. TxB Edge	TxA Neg. Edge
1	0	1	MODE 1 (PWM) TxA Toggle	Autoreload RA	Autoreload RB	t_C
1	0	0	MODE 1 (PWM) No TxA Toggle	Autoreload RA	Autoreload RB	t_C
0	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Pos. Edge	Pos. TxA Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	0	MODE 3 (Capture) Captures: TxA Pos. Edge TxB Neg. Edge	Pos. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C
0	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Pos. Edge	Neg. TxB Edge or Timer Underflow	Pos. TxB Edge	t_C
1	1	1	MODE 3 (Capture) Captures: TxA Neg. Edge TxB Neg. Edge	Neg. TxA Edge or Timer Underflow	Neg. TxB Edge	t_C

Power Save Modes

The COP888CG offers the user two power save modes of operation: HALT and IDLE. In the HALT mode, all microcontroller activities are stopped. In the IDLE mode, the on-board oscillator circuitry the WatchDog logic, the Clock Monitor and timer T0 are active but all other microcontroller activities are stopped. In either mode, all on-board RAM, registers, I/O states, and timers (with the exception of T0) are unaltered.

HALT MODE

The COP888CG is placed in the HALT mode by writing a "1" to the HALT flag (G7 data bit). All microcontroller activities, including the clock and timers, are stopped. The WatchDog logic on the COP888CG is disabled during the HALT mode. However, the clock monitor circuitry if enabled remains active. In the HALT mode, the power requirements of the COP888CG are minimal and the applied voltage (V_{CC}) may be decreased to V_r (V_r = 2.0V) without altering the state of the machine.

The COP888CG supports three different ways of exiting the HALT mode. The first method of exiting the HALT mode is with the Multi-Input Wakeup feature on the L port. The second method is with a low to high transition on the CKO (G7) pin. This method precludes the use of the crystal clock configuration (since CKO becomes a dedicated output), and so may be used with an RC clock configuration. The third method of exiting the HALT mode is by pulling the $\overline{\text{RESET}}$ pin low.

Since a crystal or ceramic resonator may be selected as the oscillator, the Wakeup signal is not allowed to start the chip running immediately since crystal oscillators and ceramic resonators have a delayed start up time to reach full amplitude and frequency stability. The IDLE timer is used to generate a fixed delay to ensure that the oscillator has indeed stabilized before allowing instruction execution. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry is enabled. The IDLE timer is loaded with a value of 256 and is clocked with the t_C instruction cycle clock. The t_C clock is derived by dividing the oscillator clock down by a factor of 10. The Schmitt trigger following the CKI inverter on the chip ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

If an RC clock option is being used, the fixed delay is introduced optionally. A control bit, CLKDLY, mapped as configuration bit G7, controls whether the delay is to be introduced or not. The delay is included if CLKDLY is set, and excluded if CLKDLY is reset. The CLKDLY bit is cleared on reset.

The COP888CG has two mask options associated with the HALT mode. The first mask option enables the HALT mode

Power Save Modes (Continued)

feature, while the second mask option disables the HALT mode. With the HALT mode enable mask option, the COP888CG will enter and exit the HALT mode as described above. With the HALT disable mask option, the COP888CG cannot be placed in the HALT mode (writing a "1" to the HALT flag will have no effect).

The WatchDog detector circuit is inhibited during the HALT mode. However, the clock monitor circuit if enabled remains active during HALT mode in order to ensure a clock monitor error if the COP888CG inadvertently enters the HALT mode as a result of a runaway program or power glitch.

IDLE MODE

The COP888CG is placed in the IDLE mode by writing a "1" to the IDLE flag (G6 data bit). In this mode, all activities, except the associated on-board oscillator circuitry, the WatchDog logic, the clock monitor and the IDLE Timer T0, are stopped. The power supply requirements of the microcontroller in this mode of operation are typically around 30% of normal power requirement of the microcontroller.

As with the HALT mode, the COP888CG can be returned to normal operation with a reset, or with a Multi-Input Wakeup from the L Port. Alternately, the microcontroller resumes normal operation from the IDLE mode when the thirteenth bit (representing 4.096 ms at internal clock frequency of 1 MHz, t_c = 1 μs) of the IDLE Timer toggles.

This toggle condition of the thirteenth bit of the IDLE Timer T0 is latched into the T0PND pending flag.

The user has the option of being interrupted with a transition on the thirteenth bit of the IDLE Timer T0. The interrupt can be enabled or disabled via the T0EN control bit. Setting the T0EN flag enables the interrupt and vice versa.

The user can enter the IDLE mode with the Timer T0 interrupt enabled. In this case, when the T0PND bit gets set, the COP888CG will first execute the Timer T0 interrupt service routine and then return to the instruction following the "Enter Idle Mode" instruction.

Alternatively, the user can enter the IDLE mode with the IDLE Timer T0 interrupt disabled. In this case, the COP888CG will resume normal operation with the instruction immediately following the "Enter IDLE Mode" instruction.

Note: It is necessary to program two NOP instructions following both the set HALT mode and set IDLE mode instructions. These NOP instructions are necessary to allow clock resynchronization following the HALT or IDLE modes.

Multi-Input Wakeup

The Multi-Input Wakeup feature is ued to return (wakeup) the COP888CG from either the HALT or IDLE modes. Alternately Multi-Input Wakeup/Interrupt feature may also be used to generate up to 8 edge selectable external interrupts.

Figure 11 shows the Multi-Input Wakeup logic for the COP888CG microcontroller.

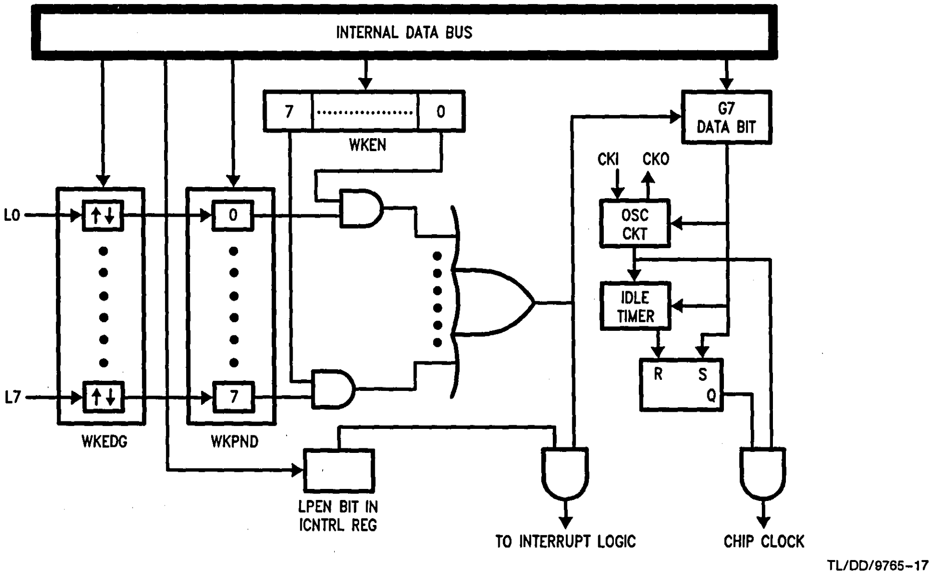


FIGURE 11. Multi-Input Wake Up Logic

Multi-Input Wakeup (Continued)

The Multi-Input Wakeup feature utilizes the L Port. The user selects which particular L port bit (or combination of L Port bits) will cause the COP888CG to exit the HALT or IDLE modes. The selection is done through the Reg: WKEN. The Reg: WKEN is an 8-bit read/write register, which contains a control bit for every L port bit. Setting a particular WKEN bit enables a Wakeup from the associated L port pin.

The user can select whether the trigger condition on the selected L Port pin is going to be either a positive edge (low to high transition) or a negative edge (high to low transition). This selection is made via the Reg: WKEDG, which is an 8-bit control register with a bit assigned to each L Port pin. Setting the control bit will select the trigger condition to be a negative edge on that particular L Port pin. Resetting the bit selects the trigger condition to be a positive edge. Changing an edge select entails several steps in order to avoid a pseudo Wakeup condition as a result of the edge change. First, the associated WKEN bit should be reset, followed by the edge select change in WKEDG. Next, the associated WKPND bit should be cleared, followed by the associated WKEN bit being re-enabled.

An example may serve to clarify this procedure. Suppose we wish to change the edge select from positive (low going high) to negative (high going low) for L Port bit 5, where bit 5 has previously been enabled for an input interrupt. The program would be as follows:

```
RBIT  5, WKEN
SBIT  5, WKEDG
RBIT  5, WKPND
SBIT  5, WKEN
```

If the L port bits have been used as outputs and then changed to inputs with Multi-Input Wakeup/Interrupt, a safety procedure should also be followed to avoid inherited pseudo wakeup conditions. After the selected L port bits have been changed from output to input but before the associated WKEN bits are enabled, the associated edge select bits in WKEDG should be set or reset for the desired edge selects, followed by the associated WKPND bits being cleared.

This same procedure should be used following reset, since the L port inputs are left floating as a result of reset.

The occurrence of the selected trigger condition for Multi-Input Wakeup is latched into a pending register called WKPND. The respective bits of the WKPND register will be set on the occurrence of the selected trigger edge on the corresponding Port L pin. The user has the responsibility of clearing these pending flags. Since WKPND is a pending register for the occurrence of selected wakeup conditions, the COP888CG will not enter the HALT mode if any Wakeup bit is both enabled and pending. Consequently, the user has the responsibility of clearing the pending flags before attempting to enter the HALT mode.

WKEN, WKPND and WKEDG are all read/write registers, and are cleared at reset.

PORT L INTERRUPTS

Port L provides the user with an additional eight fully selectable, edge sensitive interrupts which are all vectored into the same service subroutine.

The interrupt from Port L shares logic with the wake up circuitry. The register WKEN allows interrupts from Port L to be individually enabled or disabled. The register WKEDG specifies the trigger condition to be either a positive or a negative edge. Finally, the register WKPND latches in the pending trigger conditions.

The GIE (Global Interrupt Enable) bit enables the interrupt function.

A control flag, LPEN, functions as a global interrupt enable for Port L interrupts. Setting the LPEN flag will enable interrupts and vice versa. A separate global pending flag is not needed since the register WKPND is adequate.

Since Port L is also used for waking the COP888CG out of the HALT or IDLE modes, the user can elect to exit the HALT or IDLE modes either with or without the interrupt enabled. If he elects to disable the interrupt, then the COP888CG will restart execution from the instruction immediately following the instruction that placed the microcontroller in the HALT or IDLE modes. In the other case, the COP888CG will first execute the interrupt service routine and then revert to normal operation.

The Wakeup signal will not start the chip running immediately since crystal oscillators or ceramic resonators have a finite start up time. The IDLE Timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the COP888CG to execute instructions. In this case, upon detecting a valid Wakeup signal, only the oscillator circuitry and the IDLE Timer T0 are enabled. The IDLE Timer is loaded with a value of 256 and is clocked from the t_c instruction cycle clock. The t_c clock is derived by dividing down the oscillator clock by a factor of 10. A Schmitt trigger following the CKI on-chip inverter ensures that the IDLE timer is clocked only when the oscillator has a sufficiently large amplitude to meet the Schmitt trigger specifications. This Schmitt trigger is not part of the oscillator closed loop. The startup timeout from the IDLE timer enables the clock signals to be routed to the rest of the chip.

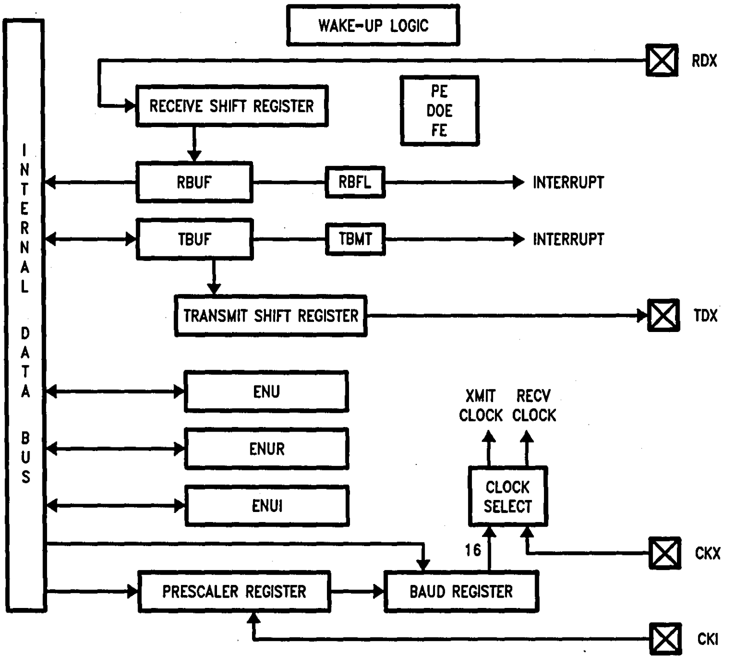
Multi-Input Wakeup (Continued)

If the RC clock option is used, the fixed delay is under software control. A control flag, CLKDLY, in the G7 configuration bit allows the clock start up delay to be optionally inserted. Setting CLKDLY flag high will cause clock start up delay to be inserted and resetting it will exclude the clock start up delay. The CLKDLY flag is cleared during reset, so the clock start up delay is not present following reset with the RC clock options.

UART

The COP888CG contains a full-duplex software programmable UART. The UART *(Figure 12)* consists of a transmit shift register, a receiver shift register and seven addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR), a UART interrupt and clock source register (ENUI), a prescaler select register (PSR) and baud (BAUD) register. The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (7, 8 or 9 bits), the value of the ninth bit in transmission, and parity selection bits. The ENUR register flags framming, data overrun and parity errors while the UART is receiving.

Other functions of the ENUR register include saving the ninth bit received in the data frame, enabling or disabling the UART's attention mode of operation and providing additional receiver/transmitter status information via RCVG and XMTG bits. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts. A control flag in this register can also select the UART mode of operation: asynchronous or synchronous.



TL/DD/9765-18

FIGURE 12. UART Block Diagram

UART (Continued)

UART CONTROL AND STATUS REGISTERS

The operation of the UART is programmed through three registers: ENU, ENUR and ENUI. The function of the individual bits in these registers is as follows:

ENU-UART Control and Status Register (Address at 0BA)

PEN	PSEL1	XBIT9/PSEL0	CHL1	CHL0	ERR	RBFL	TBMT
0RW	0RW	0RW	0RW	0RW	0R	0R	1R

Bit 7 Bit 0

ENUR-UART Receive Control and Status Register (Address at 0BB)

DOE	FE	PE	SPARE	RBIT9	ATTN	XMTG	RCVG
0RD	0RD	0RD	0RW*	0R	0RW	0R	0R

Bit7 Bit0

ENUI-UART Interrupt and Clock Source Register (Address at 0BC)

STP2	STP78	ETDX	SSEL	XRCLK	XTCLK	ERI	ETI
0RW	0RW	0RW	0RW	0RW	0RW	0RW	0RW

Bit7 Bit0

*Bit is not used.

0 Bit is cleared on reset.

1 Bit is set to one on reset.

R Bit is read-only; it cannot be written by software.

RW Bit is read/write.

D Bit is cleared on read; when read by software as a one, it is cleared automatically. Writing to the bit does not affect its state.

DESCRIPTION OF UART REGISTER BITS

ENU—UART CONTROL AND STATUS REGISTER

TBMT: This bit is set when the UART transfers a byte of data from the TBUF register into the TSFT register for transmission. It is automatically reset when software writes into the TBUF register.

RBFL: This bit is set when the UART has received a complete character and has copied it into the RBUF register. It is automatically reset when software reads the character from RBUF.

ERR: This bit is a global UART error flag which gets set if any or a combination of the errors (DOE, FE, PE) occur.

CHL1, CHL0: These bits select the character frame format. Parity is not included and is generated/verified by hardware.

CHL1 = 0, CHL0 = 0 The frame contains eight data bits.

CHL1 = 0, CHL0 = 1 The frame contains seven data bits.

CHL1 = 1, CHL0 = 0 The frame contains nine data bits.

CHL1 = 1, CHL0 = 1 Loopback Mode selected. Transmitter output internally looped back to receiver input. Nine bit framing format is used.

XBIT9/PSEL0: Programs the ninth bit for transmission when the UART is operating with nine data bits per frame. For seven or eight data bits per frame, this bit in conjunction with PSEL1 selects parity.

PSEL1, PSEL0: Parity select bits.

PSEL1 = 0, PSEL0 = 0 Odd Parity (if Parity enabled)

PSEL1 = 0, PSEL0 = 1 Even Parity (if Parity enabled)

PSEL1 = 1, PSEL0 = 0 Mark(1) (if Parity enabled)

PSEL1 = 1, PSEL0 = 1 Space(0) (if Parity enabled)

PEN: This bit enables/disables Parity (7- and 8-bit modes only).

PEN = 0 Parity disabled.

PEN = 1 Parity enabled.

ENUR—UART RECEIVE CONTROL AND STATUS REGISTER

RCVG: This bit is set high whenever a framing error occurs and goes low when RDX goes high.

XMTG: This bit is set to indicate that the UART is transmitting. It gets reset at the end of the last frame (end of last Stop bit).

ATTN: ATTENTION Mode is enabled while this bit is set. This bit is cleared automatically on receiving a character with data bit nine set.

RBIT9: Contains the ninth data bit received when the UART is operating with nine data bits per frame.

SPARE: Reserved for future use.

PE: Flags a Parity Error.

PE = 0 Indicates no Parity Error has been detected since the last time the ENUR register was read.

PE = 1 Indicates the occurence of a Parity Error.

FE: Flags a Framing Error.

FE = 0 Indicates no Framing Error has been detected since the last time the ENUR register was read.

FE = 1 Indicates the occurence of a Framing Error.

DOE: Flags a Data Overrun Error.

DOE = 0 Indicates no Data Overrun Error has been detected since the last time the ENUR register was read.

DOE = 1 Indicates the occurence of a Data Overrun Error.

ENUI—UART INTERRUPT AND CLOCK SOURCE REGISTER

ETI: This bit enables/disables interrupt from the transmitter section.

ETI = 0 Interrupt from the transmitter is disabled.

ETI = 1 Interrupt from the transmitter is enabled.

ERI: This bit enables/disables interrupt from the receiver section.

ERI = 0 Interrupt from the receiver is disabled.

ERI = 1 Interrupt from the receiver is enabled.

XTCLK: This bit selects the clock source for the transmitter-section.

XTCLK = 0 The clock source is selected through the PSR and BAUD registers.

XTCLK = 1 Signal on CKX (L1) pin is used as the clock.

XRCLK: This bit selects the clock source for the receiver section.

XRCLK = 0 The clock source is selected through the PSR and BAUD registers.

XRCLK = 1 Signal on CKX (L1) pin is used as the clock.

SSEL: UART mode select.

SSEL = 0 Asynchronous Mode.

SSEL = 1 Synchronous Mode.

UART (Continued)

ETDX: TDX (UART Transmit Pin) is the alternate function assigned to Port L pin L2; it is selected by setting ETDX bit. To simulate line break generation, software should reset ETDX bit and output logic zero to TDX pin through Port L data and configuration registers.

STP78: This bit is set to program the last Stop bit to be 7/8th of a bit in length.

STP2: This bit programs the number of Stop bits to be transmitted.

STP2 = 0 One Stop bit transmitted.
STP2 = 1 Two Stop bits transmitted.

Associated I/O Pins

Data is transmitted on the TDX pin and received on the RDX pin. TDX is the alternate function assigned to Port L pin L2; it is selected by setting ETDX (in the ENUI register) to one. RDX is an inherent function of Port L pin L3, requiring no setup.

The baud rate clock for the UART can be generated on-chip, or can be taken from an external source. Port L pin L1 (CKX) is the external clock I/O pin. The CKX pin can be either an input or an output, as determined by Port L Configuration and Data registers (Bit 1). As an input, it accepts a clock signal which may be selected to drive the transmitter and/or receiver. As an output, it presents the internal Baud Rate Generator output.

UART Operation

The UART has two modes of operation: asynchronous mode and synchronous mode.

ASYNCHRONOUS MODE

This mode is selected by resetting the SSEL (in the ENUI register) bit to zero. The input frequency to the UART is 16 times the baud rate.

The TSFT and TBUF registers double-buffer data for transmission. While TSFT is shifting out the current character on the TDX pin, the TBUF register may be loaded by software with the next byte to be transmitted. When TSFT finishes transmitting the current character the contents of TBUF are transferred to the TSFT register and the Transmit Buffer Empty Flag (TBMT in the ENU register) is set. The TBMT flag is automatically reset by the UART when software loads a new character into the TBUF register. There is also the XMTG bit which is set to indicate that the UART is transmitting. This bit gets reset at the end of the last frame (end of last Stop bit). TBUF is a read/write register.

The RSFT and RBUF registers double-buffer data being received. The UART receiver continually monitors the signal on the RDX pin for a low level to detect the beginning of a Start bit. Upon sensing this low level, it waits for half a bit time and samples again. If the RDX pin is still low, the receiver considers this to be a valid Start bit, and the remaining bits in the character frame are each sampled a single time, at the mid-bit position. Serial data input on the RDX pin is shifted into the RSFT register. Upon receiving the complete character, the contents of the RSFT register are copied into the RBUF register and the Received Buffer Full Flag (RBFL) is set. RBFL is automatically reset when software reads the character from the RBUF register. RBUF is a read only register. There is also the RCVG bit which is set high when a framing error occurs and goes low once RDX goes high. TBMT, XMTG, RBFL and RCVG are read only bits.

SYNCHRONOUS MODE

In this mode data is transferred synchronously with the clock. Data is transmitted on the rising edge and received on the falling edge of the synchronous clock.

This mode is selected by setting SSEL bit in the ENUI register. The input frequency to the UART is the same as the baud rate.

When an external clock input is selected at the CKX pin, data transmit and receive are performed synchronously with this clock through TDX/RDX pins.

If data transmit and receive are selected with the CKX pin as clock output, the μC generates the synchronous clock output at the CKX pin. The internal baud rate generator is used to produce the synchronous clock. Data transmit and receive are performed synchronously with this clock.

FRAMING FORMATS

The UART supports several serial framing formats *(Figure 13)*. The format is selected using control bits in the ENU, ENUR and ENUI registers.

The first format (1, 1a, 1b, 1c) for data transmission (CHL0 = 1, CHL1 = 0) consists of Start bit, seven Data bits (excluding parity) and 7/8, one or two Stop bits. In applications using parity, the parity bit is generated and verified by hardware.

The second format (CHL0 = 0, CHL1 = 0) consists of one Start bit, eight Data bits (excluding parity) and 7/8, one or two Stop bits. Parity bit is generated and verified by hardware.

The third format for transmission (CHL0 = 0, CHL1 = 1) consists of one Start bit, nine Data bits and 7/8, one or two Stop bits. This format also supports the UART "ATTENTION" feature. When operating in this format, all eight bits of TBUF and RBUF are used for data. The ninth data bit is transmitted and received using two bits in the ENU and ENUR registers, called XBIT9 and RBIT9. RBIT9 is a read only bit. Parity is not generated or verified in this mode.

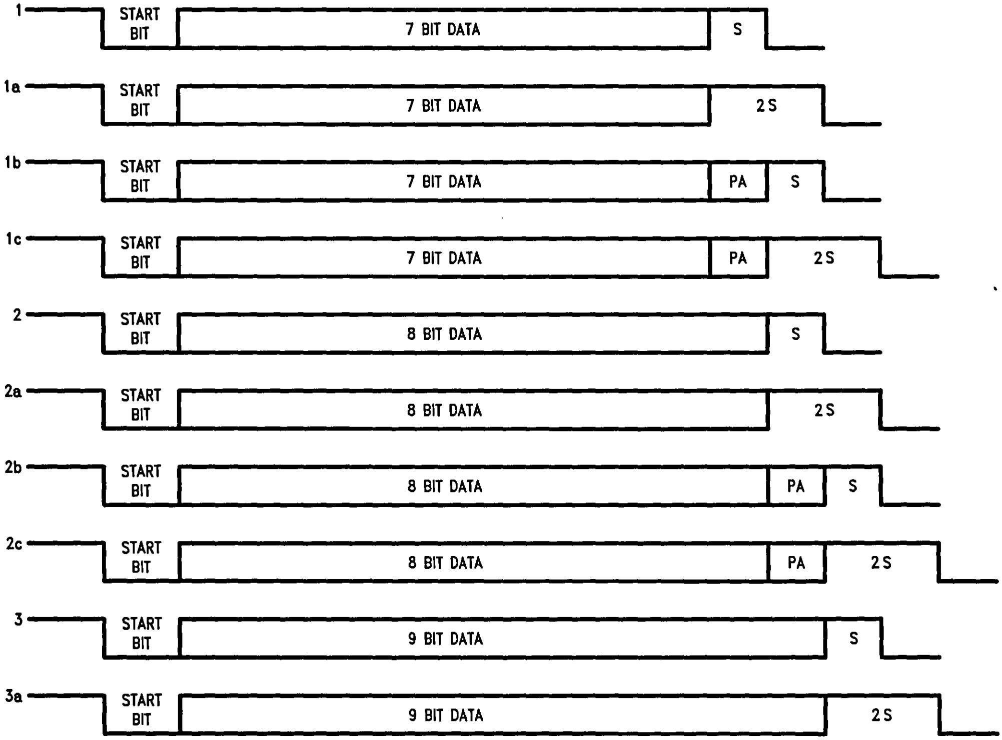
For any of the above framing formats, the last Stop bit can be programmed to be 7/8th of a bit in length. If two Stop bits are selected and the 7/8th bit is set (selected), the second Stop bit will be 7/8th of a bit in length.

The parity is enabled/disabled by PEN bit located in the ENU register. Parity is selected for 7- and 8-bit modes only. If parity is enabled (PEN = 1), the parity selection is then performed by PSEL0 and PSEL1 bits located in the ENU register.

Note that the XBIT9/PSEL0 bit located in the ENU register serves two mutually exclusive functions. This bit programs the ninth bit for transmission when the UART is operating with nine data bits per frame. There is no parity selection in this framing format. For other framing formats XBIT9 is not needed and the bit is PSEL0 used in conjunction with PSEL1 to select parity.

The frame formats for the receiver differ from the transmitter in the number of Stop bits required. The receiver only requires one Stop bit in a frame, regardless of the setting of the Stop bit selection bits in the control register. Note that an implicit assumption is made for full duplex UART operation that the framing formats are the same for the transmitter and receiver.

UART Operation (Continued)



TL/DD/9765-19

FIGURE 13. Framing Formats

UART INTERRUPTS

The UART is capable of generating interrupts. Interrupts are generated on Receive Buffer Full and Transmit Buffer Empty. Both interrupts have individual interrupt vectors. Two bytes of program memory space are reserved for each interrupt vector. The two vectors are located at addresses 0xEC to 0xEF Hex in the program memory space. The interrupts can be individually enabled or disabled using Enable Transmit Interrupt (ETI) and Enable Receive Interrupt (ERI) bits in the ENUI register.

The interrupt from the Transmitter is set pending, and remains pending, as long as both the TBMT and ETI bits are set. To remove this interrupt, software must either clear the ETI bit or write to the TBUF register (thus clearing the TBMT bit).

The interrupt from the receiver is set pending, and remains pending, as long as both the RBFL and ERI bits are set. To remove this interrupt, software must either clear the ERI bit or read from the RBUF register (thus clearing the RBFL bit).

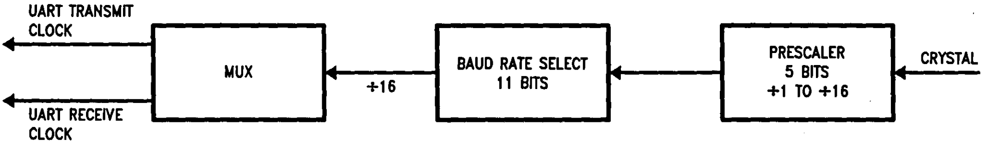
Baud Clock Generation

The clock inputs to the transmitter and receiver sections of the UART can be individually selected to come either from an external source at the CKX pin (port L, pin L1) or from a source selected in the PSR and BAUD registers. Internally, the basic baud clock is created from the oscillator frequency through a two-stage divider chain consisting of a 1–16 (increments of 0.5) prescaler and an 11-bit binary counter. *(Figure 14)* The divide factors are specified through two read/write registers shown in *Figure 15*. Note that the 11-bit Baud Rate Divisor spills over into the Prescaler Select Register (PSR). PSR is cleared upon reset.

As shown in Table I, a Prescaler Factor of 0 corresponds to NO CLOCK. NO CLOCK condition is the UART power down mode where the UART clock is turned off for power saving purpose. The user must also turn the UART clock off when a different baud rate is chosen.

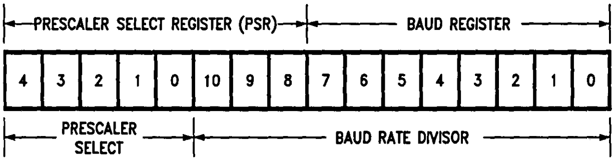
The correspondences between the 5-bit Prescaler Select and Prescaler factors are shown in Table I. Therer are many ways to calculate the two divisor factors, but one particularly effective method would be to achieve a 1.8432 MHz frequency coming out of the first stage. The 1.8432 MHz prescaler output is then used to drive the software programmable baud rate counter to create a x16 clock for the following baud rates: 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, 9600, 19200 and 38400 (Table II). Other baud rates may be created by using appropriate divisors. The x16 clock is then divided by 16 to provide the rate for the serial shift registers of the transmitter and receiver.

Baud Clock Generation (Continued)



TL/DD/9765-20

FIGURE 14. UART BAUD Clock Generation



TL/DD/9765-21

FIGURE 15. UART BAUD Clock Divisor Registers

TABLE I. Prescaler Factors

Prescaler Select	Prescaler Factor
00000	NO CLOCK
00001	1
00010	1.5
00011	2
00100	2.5
00101	3
00110	3.5
00111	4
01000	4.5
01001	5
01010	5.5
01011	6
01100	6.5
01101	7
01110	7.5
01111	8
10000	8.5
10001	9
10010	9.5
10011	10
10100	10.5
10101	11
10110	11.5
10111	12
11000	12.5
11001	13
11010	13.5
11011	14
11100	14.5
11101	15
11110	15.5
11111	16

TABLE II. Baud Rate Divisors (1.8432 MHz Prescaler Output)

Baud Rate	Baud Rate Divisor − 1 (N-1)
110 (110.03)	1046
134.5 (134.58)	855
150	767
300	383
600	191
1200	95
1800	63
2400	47
3600	31
4800	23
7200	15
9600	11
19200	5
38400	2

The entries in Table II assume a prescaler output of 1.8432 MHz. In the asynchronous mode the baud rate could be as high as 625k.

As an example, considering the Asynchronous Mode and a CKI clock of 4.608 MHz, the prescaler factor selected is:

$$4.608/1.8432 = 2.5$$

The 2.5 entry is available in Table I. The 1.8432 MHz prescaler output is then used with proper Baud Rate Divisor (Table II) to obtain different baud rates. For a baud rate of 19200 e.g., the entry in Table II is 5.

$N - 1 = 5$ (N − 1 is the value from Table II)

$N = 6$ (N is the Baud Rate Divisor)

Baud Rate = 1.8432 MHz/(16 × 6) = 19200

The divide by 16 is performed because in the asynchronous mode, the input frequency to the UART is 16 times the baud rate. The equation to calculate baud rates is given below.

The actual Baud Rate may be found from:

$$BR = Fc/(16 \times N \times P)$$

Baud Clock Generation (Continued)

Where:

BR is the Baud Rate

Fc is the CKI frequency

N is the Baud Rate Divisor (Table II).

P is the Prescaler Divide Factor selected by the value in the Prescaler Select Register (Table I)

Note: In the Synchronous Mode, the divisor 16 is replaced by one.

Example:

Asynchronous Mode:

$$\text{Crystal Frequency} = 5\ \text{MHz}$$

$$\text{Desired baud rate} = 9600$$

Using the above equation N × P can be calculated first.

$$N \times P = (5 \times 10^6)/(16 \times 9600) = 32.552$$

Now 32.552 is divided by each Prescaler Factor (Table II) to obtain a value closest to an integer. This factor happens to be 6.5 (P = 6.5).

$$N = 32.552/6.5 = 5.008\ (N = 5)$$

The programmed value (from Table II) should be 4 (N − 1).

Using the above values calculated for N and P:

$$BR = (5 \times 10^6)/(16 \times 5 \times 6.5) = 9615.384$$

$$\%\ \text{error} = (9615.385 - 9600)/9600 = 0.16$$

Effect of HALT/IDLE

The UART logic is reinitialized when either the HALT or IDLE modes are entered. This reinitialization sets the TBMT flag and resets all read only bits in the UART control and status registers. Read/Write bits remain unchanged. The Transmit Buffer (TBUF) is not affected, but the Transmit Shift register (TSFT) bits are set to one. The receiver registers RBUF and RSFT are not affected.

The μC will exit from the HALT/IDLE modes when the Start bit of a character is detected at the RDX (L3) pin. This feature is obtained by using the Multi-Input Wakeup scheme provided on the μC.

Before entering the HALT or IDLE modes the user program must select the Wakeup source to be on the RDX pin. This selection is done by setting bit 3 of WKEN (Wakeup Enable) register. The Wakeup trigger condition is then selected to be high to low transition. This is done via the WKEDG register (Bit 3 is zero.)

If the microcontroller is halted and crystal oscillator is used, the Wakeup signal will not start the chip running immediately because of the finite start up time requirement of the crystal oscillator. The idle timer (T0) generates a fixed delay to ensure that the oscillator has indeed stabilized before allowing the μC to execute code. The user has to consider this delay when data transfer is expected immediately after exiting the HALT mode.

Diagnostic

Bits CHARL0 and CHARL1 in the ENU register provide a loopback feature for diagnostic testing of the UART. When these bits are set to one, the following occur: The receiver input pin (RDX) is internally connected to the transmitter output pin (TDX); the output of the Transmitter Shift Register is "looped back" into the Receive Shift Register input. In this mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive data paths of the UART.

Note that the framing format for this mode is the nine bit format; one Start bit, nine data bits, and 7/8, one or two Stop bits. Parity is not generated or verified in this mode.

Attention Mode

The UART Receiver section supports an alternate mode of operation, referred to as ATTENTION Mode. This mode of operation is selected by the ATTN bit in the ENUR register. The data format for transmission must also be selected as having nine Data bits and either 7/8, one or two Stop bits.

The ATTENTION mode of operation is intended for use in networking the COP888CG with other processors. Typically in such environments the messages consists of device addresses, indicating which of several destinations should receive them, and the actual data. This Mode supports a scheme in which addresses are flagged by having the ninth bit of the data field set to a 1. If the ninth bit is reset to a zero the byte is a Data byte.

While in ATTENTION mode, the UART monitors the communication flow, but ignores all characters until an address character is received. Upon receiving an address character, the UART signals that the character is ready by setting the RBFL flag, which in turn interrupts the processor if UART Receiver interrupts are enabled. The ATTN bit is also cleared automatically at this point, so that data characters as well as address characters are recognized. Software examines the contents of the RBUF and responds by deciding either to accept the subsequent data stream (by leaving the ATTN bit reset) or to wait until the next address character is seen (by setting the ATTN bit again).

Operation of the UART Transmitter is not affected by selection of this Mode. The value of the ninth bit to be transmitted is programmed by setting XBIT9 appropriately. The value of the ninth bit received is obtained by reading RBIT9. Since this bit is located in ENUR register where the error flags reside, a bit operation on it will reset the error flags.

Comparators

The COP888CG contains two differential comparators, each with a pair of inputs (positive and negative) and an output. Ports I1–I3 and I4–I6 are used for the comparators. The following is the Port I assignment:

I1	Comparator1 negative input
I2	Comparator1 positive input
I3	Comparator1 output
I4	Comparator2 negative input
I5	Comparator2 positive input
I6	Comparator2 output

A Comparator Select Register (CMPSL) is used to enable the comparators, read the outputs of the comparators internally, and enable the outputs of the comparators to the pins. Two control bits (enable and output enable) and one result bit are associated with each comparator. The comparator result bits (CMP1RD and CMP2RD) are read only bits which will read as zero if the associated comparator is not enabled. The Comparator Select Register is cleared with reset, resulting in the comparators being disabled. The comparators should also be disabled before entering either the HALT or IDLE modes in order to save power. The configuration of the CMPSL register is as follows:

Comparators (Continued)

CMPSL REGISTER (ADDRESS X'00B7)

The CMPSL register contains the following bits:

CMP1EN Enable comparator 1

CMP1RD Comparator 1 result (this is a read only bit, which will read as 0 if the comparator is not enabled)

CMP10E Selects pin I3 as comparator 1 output provided that CMPIEN is set to enable the comparator

CMP2EN Enable comparator 2

CMP2RD Comparator 2 result (this is a read only bit, which will read as 0 if the comparator is not enabled)

CMP20E Selects pin I6 as comparator 2 output provided that CMP2EN is set to enable the comparator

Unused	CMP20E	CMP2RD	CMP2EN	CMP10E	CMP1RD	CMP1EN	Unused
Bit 7							Bit 0

Note that the two unused bits of CMPSL may be used as software flags.

Comparator outputs have the same spec as Ports L and G except that the rise and fall times are symmetrical.

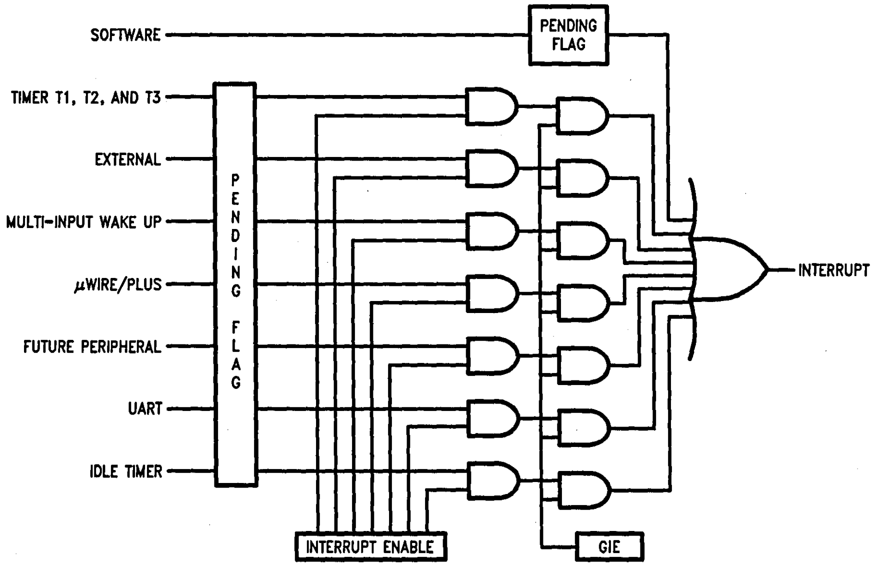
Interrupts

The COP888CG supports a vectored interrupt scheme. It supports a total of fourteen interrupt sources. The following table lists all the possible COP888CG interrupt sources, their arbitration ranking and the memory locations reserved for the interrupt vector for each source.

Two bytes of program memory space are reserved for each interrupt source. All interrupt sources except the software interrupt are maskable. Each of the maskable interrupts have an Enable bit and a Pending bit. A maskable interrupt is active if its associated enable and pending bits are set. If GIE = 1 and an interrupt is active, then the processor will be interrupted as soon as it is ready to start executing an instruction except if the above conditions happen during the Software Trap service routine. This exception is described in the Software Trap sub-section.

The interruption process is accomplished with the INTR instruction (opcode 00), which is jammed inside the Instruction Register and replaces the opcode about to be executed. The following steps are performed for every interrupt:

1. The GIE (Global Interrupt Enable) bit is reset.
2. The address of the instruction about to be executed is pushed into the stack.
3. The PC (Program Counter) branches to address 00FF. This procedure takes 7 t_C cycles to execute.



TL/DD/9765-22

FIGURE 16. COP888CG Interrupt Block Diagram

Interrupts (Continued)

Arbitration Ranking	Source	Description	Vector Address Hi-Low Byte
(1) Highest	Software	INTR Instruction	0yFE–0yFF
	Reserved	for Future Use	0yFC–0yFD
(2)	External	Pin G0 Edge	0yFA–0yFB
(3)	Timer T0	Underflow	0yF8–0yF9
(4)	Timer T1	T1A/Underflow	0yF6–0yF7
(5)	Timer T1	T1B	0yF4–0yF5
(6)	MICROWIRE/PLUS	BUSY Goes Low	0yF2–0yF3
	Reserved	for Future Use	0yF0–0yF1
(7)	UART	Receive	0yEE–0yEF
(8)	UART	Transmit	0yEC–0yED
(9)	Timer T2	T2A/Underflow	0yEA–0yEB
(10)	Timer T2	T2B	0yE8–0yE9
(11)	Timer T3	T3A/Underflow	0yE6–0yE7
(12)	Timer T3	T3B	0yE4–0yE5
(13)	Port L/Wakeup	Port L Edge	0yE2–0yE3
(14) Lowest	Default	VIS Instr. Execution without Any Interrupts	0yE0–0yE1

y is VIS page, y ≠ 0.

At this time, since GIE = 0, other maskable interrupts are disabled. The user is now free to do whatever context switching is required by saving the context of the machine in the stack with PUSH instructions. The user would then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS. Note that this is not necessarily the interrupt that caused the branch to address location 00FF Hex prior to the context switching.

Thus, if an interrupt with a higher rank than the one which caused the interruption becomes active before the decision of which interrupt to service is made by the VIS, then the interrupt with the higher rank will override any lower ones and will be acknowledged. The lower priority interrupt(s) are still pending, however, and will cause another interrupt immediately following the completion of the interrupt service routine associated with the higher priority interrupt just serviced. This lower priority interrupt will occur immediately following the RETI (Return from Interrupt) instruction at the end of the interrupt service routine just completed.

Inside the interrupt service routine, the associated pending bit has to be cleared by software. The RETI (Return from Interrupt) instruction at the end of the interrupt service routine will set the GIE (Global Interrupt Enable) bit, allowing the processor to be interrupted again if another interrupt is active and pending.

The VIS instruction looks at all the active interrupts at the time it is executed and performs an indirect jump to the beginning of the service routine of the one with the highest rank.

The addresses of the different interrupt service routines, called vectors, are chosen by the user and stored in ROM in a table starting at 01E0 (assuming that VIS is located between 00FF and 01DF). The vectors are 15-bit wide and therefore occupy 2 ROM locations.

VIS and the vector table must be located in the same 256-byte block (0y00 to 0yFF) except if VIS is located at the last address of a block. In this case, the table must be in the next block. The vector table cannot be inserted in the first 256-byte block (y ≠ 0).

The vector of the maskable interrupt with the lowest rank is located at 0yE0 (Hi-Order byte) and 0yE1 (Lo-Order byte) and so forth in increasing rank number. The vector of the maskable interrupt with the highest rank is located at 0yFA (Hi-Order byte) and 0yFB (Lo-Order byte).

The Software Trap has the highest rank and its vector is located at 0yFE and 0yFF.

If, by accident, a VIS gets executed and no interrupt is active, then the PC (Program Counter) will branch to a vector located at 0yE0–0yE1. This vector can point to the Software Trap (ST) interrupt service routine, or to another special service routine as desired.

Figure 16 shows the COP888CG Interrupt block diagram.

SOFTWARE TRAP

The Software Trap (ST) is a special kind of non-maskable interrupt which occurs when the INTR instruction (used to acknowledge interrupts) is fetched from ROM and placed inside the instruction register. This may happen when the PC is pointing beyond the available ROM address space or when the stack is over-popped.

Interrupts (Continued)

When an ST occurs, the user can re-initialize the stack pointer and do a recovery procedure (similar to reset, but not necessarily containing all of the same initialization procedures) before restarting.

The occurrence of an ST is latched into the ST pending bit. The GIE bit is not affected and the ST pending bit **(not accessible by the user)** is used to inhibit other interrupts and to direct the program to the ST service routine with the VIS instruction. The RPND instruction is used to clear the software interrupt pending bit. This pending bit is also cleared on reset.

The ST has the highest rank among all interrupts.

Nothing (except another ST) can interrupt an ST being serviced.

WatchDog

The COP888CG contains a WatchDog and clock monitor. The WatchDog is designed to detect the user program getting stuck in infinite loops resulting in loss of program control or "runaway" programs. The Clock Monitor is used to detect the absence of a clock or a very slow clock below a specified rate on the CKI pin.

The WatchDog consists of two independent logic blocks: WD UPPER and WD LOWER. WD UPPER establishes the upper limit on the service window and WD LOWER defines the lower limit of the service window.

Servicing the WatchDog consists of writing a specific value to a WatchDog Service Register named WDSVR which is memory mapped in the RAM. This value is composed of three fields, consisting of a 2-bit Window Select, a 5-bit Key Data field, and the 1-bit Clock Monitor Select field. Table III shows the WDSVR register.

The lower limit of the service window is fixed at 2048 instruction cycles. Bits 7 and 6 of the WDSVR register allow the user to pick an upper limit of the service window.

Table IV shows the four possible combinations of lower and upper limits for the WatchDog service window. This flexibility in choosing the WatchDog service window prevents any undue burden on the user software.

Bits 5, 4, 3, 2 and 1 of the WDSVR register represent the 5-bit Key Data field. The key data is fixed at 01100. Bit 0 of the WDSVR Register is the Clock Monitor Select bit.

TABLE III. Watchdog Service Register (WDSVR)

Window Select		Key Data					Clock Monitor
X	X	0	1	1	0	0	Y
7	6	5	4	3	2	1	0

TABLE IV. Watchdog Service Window Select

WDSVR Bit 7	WDSVR Bit 6	Service Window (Lower-Upper Limits)
0	0	2k–8k t_c Cycles
0	1	2k–16k t_c Cycles
1	0	2k–32k t_c Cycles
1	1	2k–64k t_c Cycles

Clock Monitor

The Clock Monitor aboard the COP888CG can be selected or deselected under program control. The Clock Monitor is guaranteed not to reject the clock if the instruction cycle clock ($1/t_c$) is greater or equal to 10 kHz. This equates to a clock input rate on CKI of greater or equal to 100 kHz.

WatchDog Operation

The WatchDog and Clock Monitor are disabled during reset. The COP888CG comes out of reset with the WatchDog armed, the WatchDog Window Select bits (bits 6, 7 of the WDSVR Register) set, and the Clock Monitor bit (bit 0 of the WDSVR Register) enabled. Thus, a Clock Monitor error will occur after coming out of reset, if the instruction cycle clock frequency has not reached a minimum specified value, including the case where the oscillator fails to start.

The WDSVR register can be written to only once after reset and the key data (bits 5 through 1 of the WDSVR Register) must match to be a valid write. This write to the WDSVR register involves two irrevocable choices: (i) the selection of the WatchDog service window (ii) enabling or disabling of the Clock Monitor. Hence, the first write to WDSVR Register involves selecting or deselecting the Clock Monitor, select the WatchDog service window and match the WatchDog key data. Subsequent writes to the WDSVR register will compare the value being written by the user to the WatchDog service window value and the key data (bits 7 through 1) in the WDSVR Register. Table V shows the sequence of events that can occur.

The user must service the WatchDog at least once before the upper limit of the serivce window expires. The WatchDog may not be serviced more than once in every lower limit of the service window. The user may service the WatchDog as many times as wished in the time period between the lower and upper limits of the service window. The first write to the WDSVR Register is also counted as a WatchDog service.

The WatchDog has an output pin associated with it. This is the WDOUT pin, on pin 1 of the port G. WDOUT is active low. The WDOUT pin is in the high impedance state in the inactive state. Upon triggering the WatchDog, the logic will pull the WDOUT (G1) pin low for an additional 16 t_c–32 t_c cycles after the signal level on WDOUT pin goes below the lower Schmitt trigger threshold. After this delay, the COP888CG will stop forcing the WDOUT output low.

The WatchDog service window will restart when the WDOUT pin goes high. It is recommended that the user tie the WDOUT pin back to V_{CC} through a resistor in order to pull WDOUT high.

A WatchDog service while the WDOUT signal is active will be ignored. The state of the WDOUT pin is not guaranteed on reset, but if it powers up low then the WatchDog will time out and WDOUT will enter high impedance state.

The Clock Monitor forces the G1 pin low upon detecting a clock frequency error. The Clock Monitor error will continue until the clock frequency has reached the minimum specified value, after which the G1 output will enter the high impedance TRI-STATE mode following 16 t_c–32 t_c clock cycles. The Clock Monitor generates a continual Clock Monitor error if the oscillator fails to start, or fails to reach the minimum specified frequency. The specification for the Clock Monitor is as follows:

$1/t_c > 10$ kHz—No clock rejection.

$1/t_c < 10$ Hz—Guaranteed clock rejection.

Detection of Illegal Conditions

The COP888CG can detect various illegal conditions resulting from coding errors, transient noise, power supply voltage drops, runaway programs, etc.

Reading of undefined ROM gets zeros. The opcode for software interrupt is zero. If the program fetches instructions from undefined ROM, this will force a software interrupt, thus signaling that an illegal condition has occurred.

The subroutine stack grows down for each call (jump to subroutine), interrupt, or PUSH, and grows up for each return or POP. The stack pointer is initialized to RAM location 06F Hex during reset. Consequently, if there are more returns than calls, the stack pointer will point to addresses 070 and 071 Hex (which are undefined RAM). Undefined RAM from addresses 070 to 07F (Segment 0), 140 to 17F (Segment 1), and all other segments (i.e., Segments 3 ... etc.) is read as all 1's, which in turn will cause the program to return to address 7FFF Hex. This is an undefined ROM location and the instruction fetched (all 0's) from this location will generate a software interrupt signaling an illegal condition.

Thus, the chip can detect the following illegal conditions:

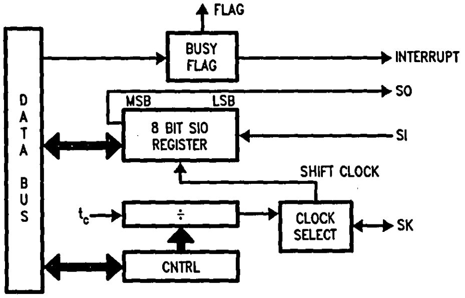
a. Executing from undefined ROM

b. Over "POP"ing the stack by having more returns than calls.

When the software interrupt occurs, the user can re-initialize the stack pointer and do a recovery procedure before restarting (this recovery program is probably similar to that following reset, but might not contain the same program initialization procedures). The recovery program should reset the software interrupt pending bit using the RPND instruction.

MICROWIRE/PLUS

MICROWIRE/PLUS is a serial synchronous communications interface. The MICROWIRE/PLUS capability enables the COP888CG to interface with any of National Semiconductor's MICROWIRE peripherals (i.e. A/D converters, display drivers, E^{2}PROMs etc.) and with other microcontrollers which support the MICROWIRE interface. It consists of an 8-bit serial shift register (SIO) with serial data input (SI), serial data output (SO) and serial shift clock (SK). *Figure 13* shows a block diagram of the MICROWIRE/PLUS logic.



TL/DD/9765-23

FIGURE 17. MICROWIRE/PLUS Block Diagram

The shift clock can be selected from either an internal source or an external source. Operating the MICROWIRE/PLUS arrangement with the internal clock source is called the Master mode of operation. Similarly, operating the MICROWIRE/PLUS arrangement with an external shift clock is called the Slave mode of operation.

The CNTRL register is used to configure and control the MICROWIRE/PLUS mode. To use the MICROWIRE/PLUS, the MSEL bit in the CNTRL register is set to one. In the master mode, the SK clock rate is selected by the two bits, SL0 and SL1, in the CNTRL register. Table VI details the different clock rates that may be selected.

TABLE V. Watchdog Service Actions

Key Data	Window Data	Clock Monitor	Action
Match	Match	Match	Valid Service: Restart Service Window
Don't Care	Mismatch	Don't Care	Error: Generate WatchDog Output
Mismatch	Don't Care	Don't Care	Error: Generate WatchDog Output
Don't Care	Don't Care	Mismatch	Error: Generate WatchDog Output

TABLE VI. MICROWIRE/PLUS Master Mode Clock Select

SL1	SL0	SK
0	0	$2 \times t_c$
0	1	$4 \times t_c$
1	x	$8 \times t_c$

Where t_c is the instruction cycle clock

MICROWIRE/PLUS (Continued)

MICROWIRE/PLUS OPERATION

Setting the BUSY bit in the PSW register causes the MICROWIRE/PLUS to start shifting the data. It gets reset when eight data bits have been shifted. The user may reset the BUSY bit by software to allow less than 8 bits to shift. If enabled, an interrupt is generated when eight data bits have been shifted. The COP888CG may enter the MICROWIRE/PLUS mode either as a Master or as a Slave. *Figure 14* shows how two COP888CG microcontrollers and several peripherals may be interconnected using the MICROWIRE/PLUS arrangements.

Warning:

The SIO register should only be loaded when the SK clock is low. Loading the SIO register while the SK clock is high will result in undefined data in the SIO register. SK clock is normally low when not shifting.

Setting the BUSY flag when the input SK clock is high in the MICROWIRE/PLUS slave mode may cause the current SK clock forthe SIO shift register to be narrow. For safety, the BUSY flag should only be set when the input SK clock is low.

MICROWIRE/PLUS Master Mode Operation

In the MICROWIRE/PLUS Master mode of operation the shift clock (SK) is generated internally by the COP888CG. The MICROWIRE Master always initiates all data exchanges. The MSEL bit in the CNTRL register must be set to enable the SO and SK functions onto the G Port. The SO and SK pins must also be selected as outputs by setting appropriate bits in the Port G configuration register. Table VII summarizes the bit settings required for Master mode of operation.

MICROWIRE/PLUS Slave Mode Operation

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be selected as an input and the SO pin is selected as an output pin by setting and resetting the appropriate bit in the Port G configuration register. Table VII summarizes the settings required to enter the Slave mode of operation.

The user must set the BUSY flag immediately upon entering the Slave mode. This will ensure that all data bits sent by the Master will be shifted properly. After eight clock pulses the BUSY flag will be cleared and the sequence may be repeated.

Alternate SK Phase Operation

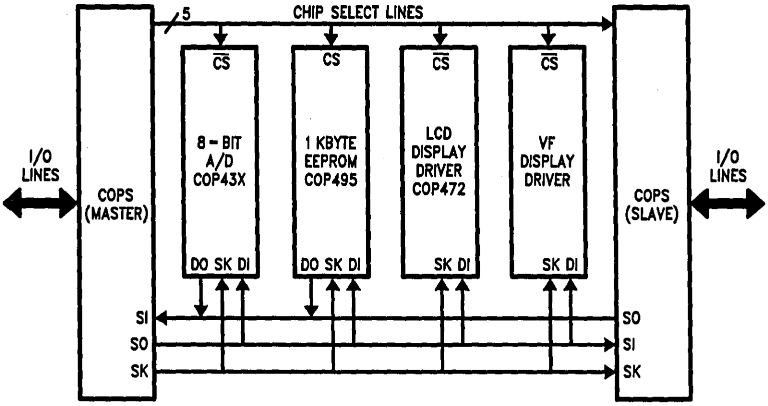
The COP888CG allows either the normal SK clock or an alternate phase SK clock to shift data in and out of the SIO register. In both the modes the SK is normally low. In the normal mode data is shifted in on the rising edge of the SK clock and the data is shifted out on the falling edge of the SK clock. The SIO register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and shifted out on the rising edge of the SK clock.

A control flag, SKSEL, allows either the normal SK clock or the alternate SK clock to be selected. Resetting SKSEL causes the MICROWIRE/PLUS logic to be clocked from the normal SK signal. Setting the SKSEL flag selects the alternate SK clock. The SKSEL is mapped into the G6 configuration bit. The SKSEL flag will power up in the reset condition, selecting the normal SK signal.

TABLE VII

This table assumes that the control flag MSEL is set.

G4 (SO) Config. Bit	G5 (SK) Config. Bit	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE/PLUS Master
0	1	TRI-STATE	Int. SK	MICROWIRE/PLUS Master
1	0	SO	Ext. SK	MICROWIRE/PLUS Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE/PLUS Slave



TL/DD/9765-24

FIGURE 18. MICROWIRE/PLUS Application

Memory Map

All RAM, ports and registers (except A and PC) are mapped into data memory address space.

Address S/ADD REG	Contents
0000 to 006F	On-Chip RAM bytes (112 bytes)
0070 to 007F	Unused RAM Address Space (Reads As All Ones)
xx80 to xxAF	Unused RAM Address Space (Reads Undefined Data)
xxB0	Timer T3 Lower Byte
XXB1	Timer T3 Upper Byte
xxB2	Timer T3 Autoload Register T3RA Lower Byte
xxB3	Timer T3 Autoload Register T3RA Upper Byte
xxB4	Timer T3 Autoload Register T3RB Lower Byte
xxB5	Timer T3 Autoload Register T3RB Upper Byte
xxB6	Timer T3 Control Register
xxB7	Comparator Select Register (CMPSL)
xxB8	UART Transmit Buffer (TBUF)
xxB9	UART Receive Buffer (RBUF)
xxBA	UART Control and Status Register (ENU)
xxBB	UART Receive Control and Status Register (ENUR)
xxBC	UART Interrupt and Clock Source Register (ENUI)
xxBD	UART Baud Register (BAUD)
xxBE	UART Prescale Select Register (PSR)
xxBF	Reserved for UART
xxC0	Timer T2 Lower Byte
xxC1	Timer T2 Upper Byte
xxC2	Timer T2 Autoload Register T2RA Lower Byte
xxC3	Timer T2 Autoload Register T2RA Upper Byte
xxC4	Timer T2 Autoload Register T2RB Lower Byte
xxC5	Timer T2 Autoload Register T2RB Upper Byte
xxC6	Timer T2 Control Register
xxC7	WatchDog Service Register (Reg:WDSVR)
xxC8	MIWU Edge Select Register (Reg:WKEDG)
xxC9	MIWU Enable Register (Reg:WKEN)
xxCA	MIWU Pending Register (Reg:WKPND)
xxCB	Reserved
xxCC	Reserved
xxCD to xxCF	Reserved

Address S/ADD REG	Contents
xxD0	Port L Data Register
xxD1	Port L Configuration Register
xxD2	Port L Input Pins (Read Only)
xxD3	Reserved for Port L
xxD4	Port G Data Register
xxD5	Port G Configuration Register
xxD6	Port G Input Pins (Read Only)
xxD7	Port I Input Pins (Read Only)
xxD8	Port C Data Register
xxD9	Port C Configuration Register
xxDA	Port C Input Pins (Read Only)
xxDB	Reserved for Port C
xxDC	Port D
xxDD to DF	Reserved for Port D
xxE0 to xxE5	Reserved for EE Control Registers
xxE6	Timer T1 Autoload Register T1RB Lower Byte
xxE7	Timer T1 Autoload Register T1RB Upper Byte
xxE8	ICNTRL Register
xxE9	MICROWIRE/PLUS Shift Register
xxEA	Timer T1 Lower Byte
xxEB	Timer T1 Upper Byte
xxEC	Timer T1 Autoload Register T1RA Lower Byte
xxED	Timer T1 Autoload Register T1RA Upper Byte
xxEE	CNTRL Control Register
xxEF	PSW Register
xxF0 to FB	On-Chip RAM Mapped as Registers
xxFC	X Register
xxFD	SP Register
xxFE	B Register
xxFF	S Register
0100-013F	On-Chip RAM Bytes (64 bytes)

Reading memory locations 0070H-007FH (Segment 0) will return all ones. Reading unused memory locations 0080H-00AFH (Segment 0) will return undefined data. Reading unused memory locations 0140-017F (Segment 1) will return all ones. Reading memory locations from other Segments (i.e., Segment 2, Segment 3, ... etc.) will return all ones.

Addressing Modes

The COP888CG has ten addressing modes, six for operand addressing and four for transfer of control.

OPERAND ADDRESSING MODES

Register Indirect

This is the "normal" addressing mode for the COP888CG. The operand is the data memory addressed by the B pointer or X pointer.

Register Indirect (with auto post increment or decrement of pointer)

This addressing mode is used with the LD and X instructions. The operand is the data memory addressed by the B pointer or X pointer. This is a register indirect mode that automatically post increments or decrements the B or X register after executing the instruction.

Direct

The instruction contains an 8-bit address field that directly points to the data memory for the operand.

Immediate

The instruction contains an 8-bit immediate field as the operand.

Short Immediate

This addressing mode is used with the LBI instruction. The instruction contains a 4-bit immediate field as the operand.

Indirect

This addressing mode is used with the LAID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a data operand from the program memory.

TRANSFER OF CONTROL ADDRESSING MODES

Relative

This mode is used for the JP instruction, with the instruction field being added to the program counter to get the new program location. JP has a range from −31 to +32 to allow a 1-byte relative jump (JP + 1 is implemented by a NOP instruction). There are no "pages" when using JP, since all 15 bits of PC are used.

Absolute

This mode is used with the JMP and JSR instructions, with the instruction field of 12 bits replacing the lower 12 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory segment.

Absolute Long

This mode is used with the JMPL and JSRL instructions, with the instruction field of 15 bits replacing the entire 15 bits of the program counter (PC). This allows jumping to any location in the current 4k program memory space.

Indirect

This mode is used with the JID instruction. The contents of the accumulator are used as a partial address (lower 8 bits of PC) for accessing a location in the program memory. The contents of this program memory location serve as a partial address (lower 8 bits of PC) for the jump to the next instruction.

Note: The VIS is a special case of the Indirect Transfer of Control addressing mode, where the double byte vector associated with the interrupt is transferred from adjacent addresses in the program memory into the program counter (PC) in order to jump to the associated interrupt service routine.

Instruction Set

Register and Symbol Definition

Registers	
A	8-Bit Accumulator Register
B	8-Bit Address Register
X	8-Bit Address Register
SP	8-Bit Stack Pointer Register
PC	15-Bit Program Counter Register
PU	Upper 7 Bits of PC
PL	Lower 8 Bits of PC
C	1 Bit of PSW Register for Carry
HC	1 Bit of PSW Register for Half Carry
GIE	1 Bit of PSW Register for Global Interrupt Enable
VU	Interrupt Vector Upper Byte
VL	Interrupt Vector Lower Byte

Symbols	
[B]	Memory Indirectly Addressed by B Register
[X]	Memory Indirectly Addressed by X Register
MD	Direct Addressed Memory
Mem	Direct Addressed Memory or [B]
MemI	Direct Addressed Memory or [B] or Immediate Data
Imm	8-Bit Immediate Data
Reg	Register Memory: Addresses F0 to FF (Includes B, X and SP)
Bit	Bit Number (0 to 7)
←	Loaded with
↔	Exchanged with

Instruction Set (Continued)

INSTRUCTION SET

ADD	A,MemI	ADD	A ← A + MemI

ADC	A,MemI	ADD with Carry	A ← A + MemI + C, C ← Carry
			HC ← Half Carry

SUBC	A,MemI	Subtract with Carry	A ← A − $\overline{\text{MemI}}$ + C, C ← Carry
			HC ← Half Carry

AND	A,MemI	Logical AND	A ← A and MemI
ANDSZ	A,Imm	Logical AND Immed., Skip if Zero	Skip next if (A and Imm) = 0
OR	A,MemI	Logical OR	A ← A or MemI
XOR	A,MemI	Logical EXclusive OR	A ← A xor MemI
IFEQ	MD,Imm	IF EQual	Compare MD and Imm, Do next if MD = Imm
IFEQ	A,MemI	IF EQual	Compare A and MemI, Do next if A = MemI
IFNE	A,MemI	IF Not Equal	Compare A and MemI, Do next if A ≠ MemI
IFGT	A,MemI	IF Greater Than	Compare A and MemI, Do next if A > MemI
IFBNE	#	If B Not Equal	Do next if lower 4 bits of B ≠ Imm
DRSZ	Reg	Decrement Reg., Skip if Zero	Reg ← Reg − 1, Skip if Reg = 0
SBIT	#,Mem	Set BIT	1 to bit, Mem (bit = 0 to 7 immediate)
RBIT	#,Mem	Reset BIT	0 to bit, Mem
IFBIT	#,Mem	IF BIT	If bit in A or Mem is true do next instruction
RPND		Reset PeNDing Flag	Reset Software Interrupt Pending Flag
X	A,Mem	EXchange A with Memory	A ↔ Mem
X	A,[X]	EXchange A with Memory [X]	A ↔ [X]
LD	A,MemI	LoaD A with Memory	A ← MemI
LD	A,[X]	LoaD A with Memory [X]	A ← [X]
LD	B,Imm	LoaD B with Immed.	B ← Imm
LD	Mem,Imm	LoaD Memory Immed	Mem ← Imm
LD	Reg,Imm	LoaD Register Memory Immed.	Reg ← Imm
X	A, [B ±]	EXchange A with Memory [B]	A ↔ [B], (B ← B ±1)
X	A, [X ±]	EXchange A with Memory [X]	A ↔ [X], (X ← ±1)
LD	A, [B±]	LoaD A with Memory [B]	A ← [B], (B ← B ±1)
LD	A, [X±]	LoaD A with Memory [X]	A ← [X], (X ← X±1)
LD	[B±],Imm	LoaD Memory [B] Immed.	[B] ← Imm, (B ← B±1)
CLR	A	CLeaR A	A ← 0
INC	A	INCrement A	A ← A + 1
DEC	A	DECrementA	A ← A − 1
LAID		Load A InDirect from ROM	A ← ROM (PU,A)
DCOR	A	Decimal CORrect A	A ← BCD correction of A (follows ADC, SUBC)
RRC	A	Rotate A Right thru C	C → A7 → . . . → A0 → C
RLC	A	Rotate A Left thru C	C ← A7 ← . . . ← A0 ← C
SWAP	A	SWAP nibbles of A	A7 . . . A4 ↔ A3 . . . A0
SC		Set C	C ← 1, HC ← 1
RC		Reset C	C ← 0, HC ← 0
IFC		IF C	IF C is true, do next instruction
IFNC		IF Not C	If C is not true, do next instruction
POP	A	POP the stack into A	SP ← SP + 1, A ← [SP]
PUSH	A	PUSH A onto the stack	[SP] ← A, SP ← SP − 1
VIS		Vector to Interrupt Service Routine	PU ← [VU], PL ← [VL]
JMPL	Addr.	Jump absolute Long	PC ← ii (ii = 15 bits, 0 to 32k)
JMP	Addr.	Jump absolute	PC9 . . . 0 ← i (i = 12 bits)
JP	Disp.	Jump relative short	PC ← PC + r (r is −31 to +32, except 1)
JSRL	Addr.	Jump SubRoutine Long	[SP] ← PL, [SP−1] ← PU,SP−2, PC ← ii
JSR	Addr	Jump SubRoutine	[SP] ← PL, [SP−1] ← PU,SP−2, PC9 . . . 0 ← i
JID		Jump InDirect	PL ← ROM (PU,A)
RET		RETurn from subroutine	SP + 2, PL ← [SP], PU ← [SP−1]
RETSK		RETurn and SKip	SP + 2, PL ← [SP],PU ← [SP−1]
RETI		RETurn from Interrupt	SP + 2, PL ← [SP],PU ← [SP−1],GIE ← 1
INTR		Generate an Interrupt	[SP] ← PL, [SP−1] ← PU, SP−2, PC ← 0FF
NOP		No OPeration	PC ← PC + 1

Instruction Execution Time

Most instructions are single byte (with immediate addressing mode instructions taking two bytes).

Most single byte instructions take one cycle time (1 μs at 10 MHz) to execute.

See the BYTES and CYCLES per INSTRUCTION table for details.

Bytes and Cycles per Instruction

The following table shows the number of bytes and cycles for each instruction in the format of byte/cycle (a cycle is 1 μs at 10 MHz).

	[B]	Direct	Immed.
ADD	1/1	3/4	2/2
ADC	1/1	3/4	2/2
SUBC	1/1	3/4	2/2
AND	1/1	3/4	2/2
OR	1/1	3/4	2/2
XOR	1/1	3/4	2/2
IFEQ	1/1	3/4	2/2
IFNE	1/1	3/4	2/2
IFGT	1/1	3/4	2/2
IFBNE	1/1		
DRSZ		1/3	
SBIT	1/1	3/4	
RBIT	1/1	3/4	
IFBIT	1/1	3/4	

RPND	1/1

Instructions Using A & C

CLRA	1/1
INCA	1/1
DECA	1/1
LAID	1/3
DCOR	1/1
RRCA	1/1
RLCA	1/1
SWAPA	1/1
SC	1/1
RC	1/1
IFC	1/1
IFNC	1/1
PUSHA	1/3
POPA	1/3
ANDSZ	2/2

Transfer of Control Instructions

JMPL	3/4
JMP	2/3
JP	1/3
JSRL	3/5
JSR	2/5
JID	1/3
VIS	1/5
RET	1/5
RETSK	1/5
RETI	1/5
INTR	1/7
NOP	1/1

Memory Transfer Instructions

	Register Indirect		Direct	Immed.	Register Indirect Auto Incr. & Decr.		
	[B]	[X]			[B+, B−]	[X+, X−]	
X A,*	1/1	1/3	2/3		1/2	1/3	
LD A,*	1/1	1/3	2/3	2/2	1/2	1/3	
LD B, Imm				1/1			(IF B < 16)
LD B, Imm				2/2			(IF B > 15)
LD Mem, Imm	2/2		3/3		2/2		
LD Reg, Imm			2/3				
IFEQ MD, Imm			3/3				

* = > Memory location addressed by B or X or directly.

COP888CG Opcode Table

Upper Nibble Along X-Axis
Lower Nibble Along Y-Axis

F	E	D	C	B	A	9	8	
JP −15	JP −31	LD 0F0, # i	DRSZ 0F0	RRCA	RC	ADC A,#i	ADC A,[B]	0
JP −14	JP −30	LD 0F1, # i	DRSZ 0F1	*	SC	SUBC A, #i	SUB A,[B]	1
JP −13	JP −29	LD 0F2, # i	DRSZ 0F2	X A, [X+]	X A,[B+]	IFEQ A,#i	IFEQ A,[B]	2
JP −12	JP −28	LD 0F3, # i	DRSZ 0F3	X A, [X−]	X A,[B−]	IFGT A,#i	IFGT A,[B]	3
JP −11	JP −27	LD 0F4, # i	DRSZ 0F4	VIS	LAID	ADD A,#i	ADD A,[B]	4
JP −10	JP −26	LD 0F5, # i	DRSZ 0F5	RPND	JID	AND A,#i	AND A,[B]	5
JP −9	JP −25	LD 0F6, # i	DRSZ 0F6	X A,[X]	X A,[B]	XOR A,#i	XOR A,[B]	6
JP −8	JP −24	LD 0F7, # i	DRSZ 0F7	*	*	OR A,#i	OR A,[B]	7
JP −7	JP −23	LD 0F8, # i	DRSZ 0F8	NOP	RLCA	LD A,#i	IFC	8
JP −6	JP −22	LD 0F9, # i	DRSZ 0F9	IFNE A,[B]	IFEQ Md,#i	IFNE A,#i	IFNC	9
JP −5	JP −21	LD 0FA, # i	DRSZ 0FA	LD A,[X+]	LD A,[B+]	LD [B+],#i	INCA	A
JP −4	JP −20	LD 0FB, # i	DRSZ 0FB	LD A,[X−]	LD A,[B−]	LD [B−],#i	DECA	B
JP −3	JP −19	LD 0FC, # i	DRSZ 0FC	LD Md,#i	JMPL	X A,Md	POPA	C
JP −2	JP −18	LD 0FD, # i	DRSZ 0FD	DIR	JSRL	LD A,Md	RETSK	D
JP −1	JP −17	LD 0FE, # i	DRSZ 0FE	LD A,[X]	LD A,[B]	LD [B],#i	RET	E
JP −0	JP −16	LD 0FF, # i	DRSZ 0FF	*	*	LD B,#i	RETI	F

COP888CG Opcode Table (Continued)

Upper Nibble Along X-Axis
Lower Nibble Along Y-Axis

7	6	5	4	3	2	1	0	
IFBIT 0,[B]	ANDSZ A, #i	LD B,#0F	IFBNE 0	JSR x000–x0FF	JMP x000–x0FF	JP +17	INTR	0
IFBIT 1,[B]	*	LD B,#0E	IFBNE 1	JSR x100–x1FF	JMP x100–x1FF	JP +18	JP + 2	1
IFBIT 2,[B]	*	LD B,#0D	IFBNE 2	JSR x200–x2FF	JMP x200–x2FF	JP +19	JP + 3	2
IFBIT 3,[B]	*	LD B,#0C	IFBNE 3	JSR x300–x3FF	JMP x300–x3FF	JP +20	JP + 4	3
IFBIT 4,[B]	CLRA	LD B,#0B	IFBNE 4	JSR x400–x4FF	JMP x400–x4FF	JP +21	JP + 5	4
IFBIT 5,[B]	SWAPA	LD B,#0A	IFBNE 5	JSR x500–x5FF	JMP x500–x5FF	JP +22	JP + 6	5
IFBIT 6,[B]	DCORA	LD B,#09	IFBNE 6	JSR x600–x6FF	JMP x600–x6FF	JP +23	JP + 7	6
IFBIT 7,[B]	PUSHA	LD B,#08	IFBNE 7	JSR x700–x7FF	JMP x700–x7FF	JP +24	JP + 8	7
SBIT 0,[B]	RBIT 0,[B]	LD B,#07	IFBNE 8	JSR x800–x8FF	JMP x800–x8FF	JP +25	JP + 9	8
SBIT 1,[B]	RBIT 1,[B]	LD B,#06	IFBNE 9	JSR x900–x9FF	JMP x900–x9FF	JP +26	JP + 10	9
SBIT 2,[B]	RBIT 2,[B]	LD B,#05	IFBNE 0A	JSR xA00–xAFF	JMP xA00–xAFF	JP +27	JP + 11	A
SBIT 3,[B]	RBIT 3,[B]	LD B,#04	IFBNE 0B	JSR xB00–xBFF	JMP xB00–xBFF	JP +28	JP + 12	B
SBIT 4,[B]	RBIT 4,[B]	LD B,#03	IFBNE 0C	JSR xC00–xCFF	JMP xC00–xCFF	JP +29	JP + 13	C
SBIT 5,[B]	RBIT 5,[B]	LD B,#02	IFBNE 0D	JSR xD00–xDFF	JMP xD00–xDFF	JP +30	JP + 14	D
SBIT 6,[B]	RBIT 6,[B]	LD B,#01	IFBNE 0E	JSR xE00–xEFF	JMP xE00–xEFF	JP +31	JP + 15	E
SBIT 7,[B]	RBIT 7,[B]	LD B,#00	IFBNE 0F	JSR xF00–xFFF	JMP xF00–xFFF	JP +32	JP + 16	F

Where,

i is the immediate data
Md is a directly addressed memory location
* is an unused opcode

Note: The opcode 60 Hex is also the opcode for IFBIT #i,A

Mask Options

The COP888CG mask programmable options are shown below. The options are programmed at the same time as the ROM pattern submission.

```
OPTION 1: CLOCK CONFIGURATION
  = 1     Crystal Oscillator (CKI/10)
              G7 (CKO) is clock generator
              output to crystal/resonator
              CKI is the clock input
  = 2     Single-pin RC controlled
              oscillator (CKI/10)
              G7 is available as a HALT
              restart and/or general purpose
              input
OPTION 2: HALT
  = 1     Enable HALT mode
  = 2     Disable HALT mode
OPTION 3: COP888CG BONDING
  = 1     44-Pin PCC
  = 2     40-Pin DIP
  = 3     28-Pin PCC
  = 4     28-Pin DIP
```

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (if clock option-1 has been selected). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

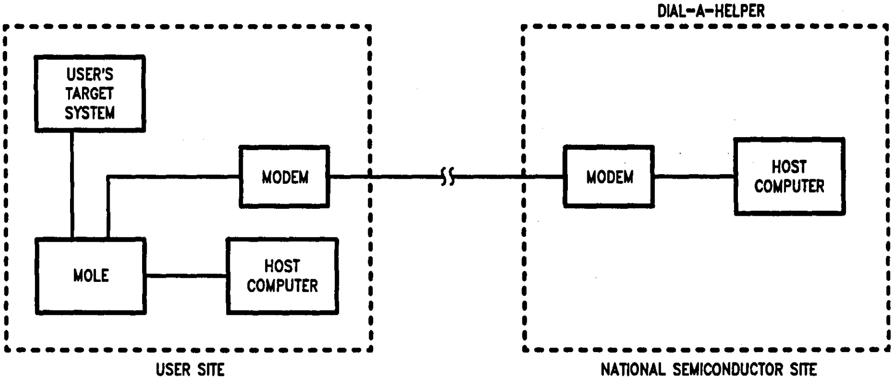
ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
- Dial-A-Helper Users Manual
- Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 Baud	
	Set-up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hrs., 7 Days	



TL/DD/9765-25

National Semiconductor

PRELIMINARY

COP820CP-X/COP840CP-X Piggyback EPROM Microcontrollers

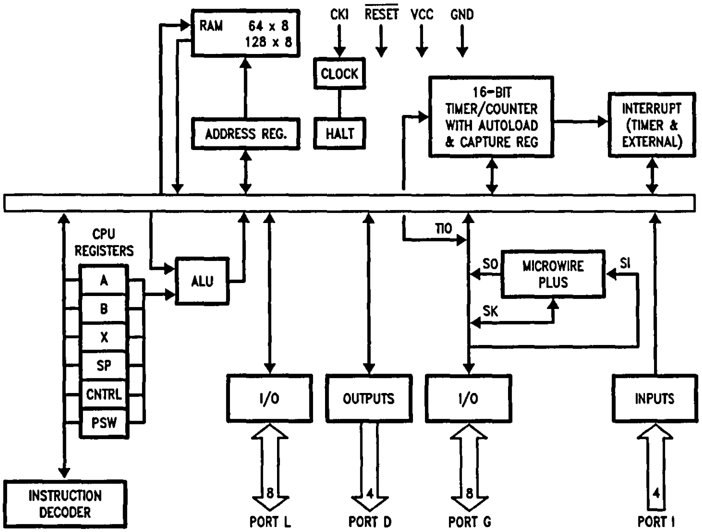
General Description

The COP820CP/COP840CP are piggyback versions of the COP820C/COP840C microcontroller families. They are fully static parts, fabricated using double-metal silicon gate microCMOS technology. These microcontrollers are a complete microcomputer containing all system timing, interrupt logic, RAM, and I/O necessary to implement dedicated control functions in a variety of applications. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, a 16-bit timer/counter with capture register and a multi-sourced interrupt. Each I/O pin has software selectable options to adapt the emulator to the specific application. The part operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a 1 μs per instruction rate. The COP820CP-X/COP840CP-X are totally compatible with the ROM based COP820C/COP840 microcontroller. It serves as an economical low and medium volume emulator devices for the COP820C/COP840C microcontroller family.

Features

- Low cost 8-bit CORE microcontroller
- Fully static CMOS
- 1 μs instruction time (20 MHz clock)
- Low current drain
- Single supply operation: 4.5V to 5.5V
- Up to 32 kbytes of addressable memory
- 64 bytes of RAM (128 bytes for COP840CP)
- 16-bit read/write timer operates in a variety of modes
  - — Timer with 16-bit auto reload register
  - — 16-bit external event counter
  - — Timer with 16-bit capture register (selectable edge)
- Multi-source interrupt
  - — Reset master clear
  - — External interrupt with selectable edge
  - — Timer interrupt or capture interrupt
  - — Software interrupt
- 8-bit stack pointer (stack in RAM)
- Powerful instruction set, most instructions single byte
- BCD arithmetic instruction
- MICROWIRE/PLUS serial I/O
- 28 pin package
- 24 input/output pins (28-pin package)
- Software selectable I/O options (TRI-STATE®, push-pull, weak pull-up)
- Schmitt trigger inputs on Port G
- Fully supported by National's MOLE™ development system

Block Diagram



TL/DD/9683-1

FIGURE 1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
Total Current into V_{CC} Pin (Source)	150 mA
Total Current into GND Pin (Sink)	160 mA
Storage Temperature Range	$-65°C$ to $+140°C$

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak to Peak			$0.1\ V_{CC}$	V
Supply Current (Note 2)					
High Speed Mode, CKI = 20 MHz	$V_{CC} = 5.5V$, $t_C = 1\ \mu s$			95	mA
Normal Mode, CKI = 5 MHz	$V_{CC} = 5.5V$, $t_C = 2\ \mu s$			90	mA
HALT Current (Note 3)	$V_{CC} = 5.5V$, CKI = 0 MHz (Note 4)			80	mA
INPUT LEVELS					
Reset and CKI (Crystal Osc.)					
Logic High		$0.9\ V_{CC}$			V
Logic Low				$0.1\ V_{CC}$	V
All Other Inputs					
Logic High		$0.7\ V_{CC}$			V
Logic Low				$0.2\ V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	-10		$+10$	μA
G and L Port Input Hysteresis	$V_{CC} = 5.5V$		$0.05\ V_{CC}$		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V$, $V_{OH} = 3.8V$	0.4			mA
Sink	$V_{CC} = 4.5V$, $V_{OL} = 1.0V$	10			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V$, $V_{OH} = 3.2V$	10		100	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OH} = 3.8V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage	$V_{CC} = 5.5V$	-2.0		$+2.0$	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
RAM Retention Voltage, Vr	500 ns Rise and Fall Time (Min)		2.0		V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Condition	Min	Typ	Max	Units
Instruction Cycle Time (t_C)					
High Speed Mode (Div-by 20)	V_{CC} ≥ 4.5V	1		DC	μs
Normal Mode (Div-by 10)	V_{CC} ≥ 4.5V	2		DC	μs
R/C Oscillator Mode (Div-by 10)	V_{CC} ≥ 4.5V	3		DC	μs
CKI Clock Duty Cycle (Note 5)	fr = Max	33		66	%
Rise Time (Note 5)	fr = 20 MHz Ext Clock			12	ns
Fall Time (Note 5)	fr = 20 MHz Ext Clock			8	ns
Inputs					
t_{SETUP}	V_{CC} ≥ 4.5V	200			ns
t_{HOLD}	V_{CC} ≥ 4.5V	60			ns
Output Propagation Delay t_{PD1}, t_{PD0} (Note 6)	R_L = 2.2k, C_L = 100 pF				
SO, SK	V_{CC} ≥ 4.5V			0.7	μs
All Others	V_{CC} ≥ 4.5V			1	μs
MICROWIRE™ Setup Time (tUWS)		20			ns
MICROWIRE Hold Time (tUWH)		56			ns
MICROWIRE Output Valid Time (tUPD)				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_C
Interrupt Input Low Time		1			t_C
Timer Input High Time		1			t_C
Timer Input Low Time		1			t_C
Reset Pulse Width		1			μs

Note 1: The rate of voltage change must be less than 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground.

Note 4: This includes the EPROM, and the pull-up resistors on the D and I ports.

Note 5: Parameter sampled but not 100% tested.

Note 6: There is one cycle delay on ports I and D.

EPROM Selection

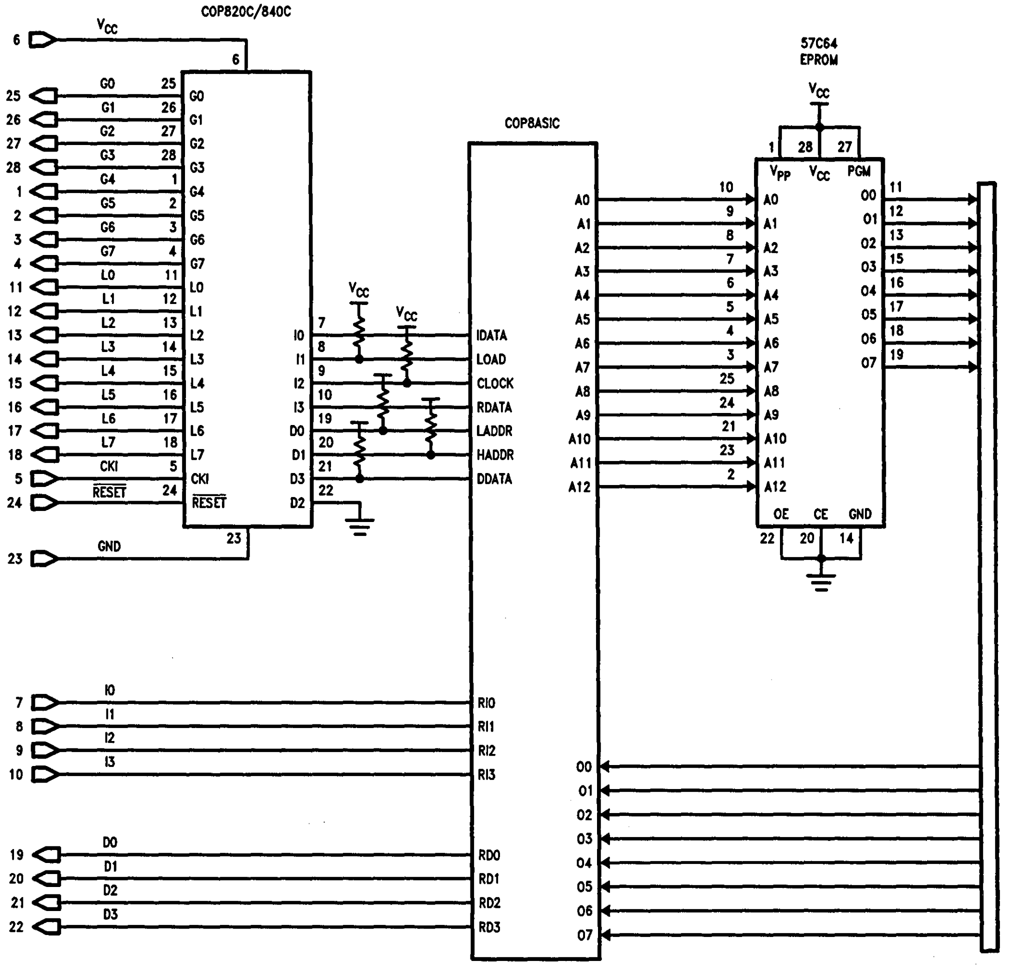
The COP820CP-X/COP840CP-X, (where X=1, 2, 3, 4 or 5, see Table II), are the piggyback versions of the COP820C/COP840C microcontrollers. They are identical to their respective devices except that the program memory has been removed. The device package incorporates the circuitry and the socket on top of the package to allow plugging-in the EPROM 57C64, an 8 kbyte device, or any other comparable EPROM, for high speed operation. With the addition of an EPROM, these devices will perform exactly as their factory masked equivalent.

Table I lists the minimum EPROM access time for a given instruction cycle time of the microcontroller.

TABLE I

EPROM Minimum Access Time	COP8 Instruction Cycle Time
120 ns	1.00 μs
150 ns	1.10 μs
200 ns	1.27 μs
250 ns	1.44 μs
300 ns	1.60 μs
400 ns	1.94 μs

Connection Diagram

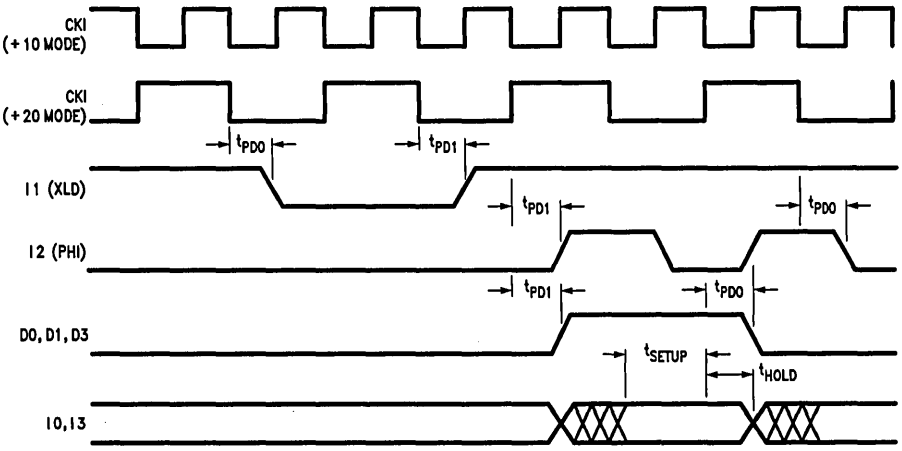


TL/DD/9683-2

All resistors are 330Ω ±20%

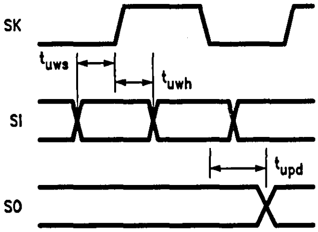
FIGURE 2

AC Timing Diagram



TL/DD/9683-3

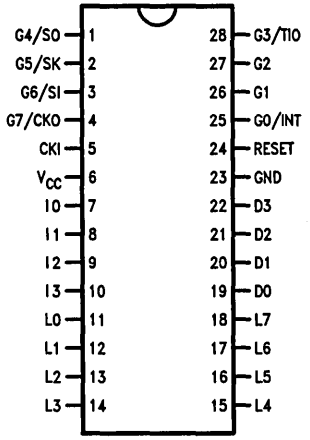
FIGURE 3



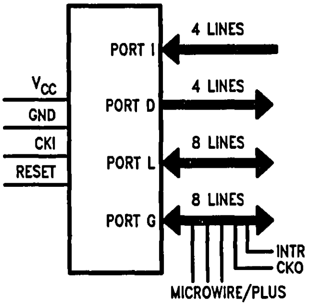
TL/DD/9683-10

FIGURE 3b

COP820CP-X/COP840CP-X Pinout Diagrams



TL/DD/9683-4



TL/DD/9683-5

Order Number COP820CP-X or COP840CP-X

FIGURE 4

Oscillator Circuits

Figure 5 shows the clock oscillator configurations available for the COP820CP-X/COP840CP-X.

A. CRYSTAL OSCILLATOR

The COP820CP-X/COP840CP-X can be driven by a crystal clock. The crystal network is connected between the pins CKI and CKO.

Table IA shows the component values required for various standard crystal values.

B. EXTERNAL OSCILLATOR

CKI can be driven by an external clock signal. CKO is available as a general purpose input and/or HALT restart control.

C. RC OSCILLATOR

CKI is configured as a single pin RC controlled Schmitt trigger oscillator. CKO is available as a general purpose input and/or HALT control.

Table IB shows the variation in the oscillator frequencies as functions of the R and C component values.

TABLE IA. Crystal Oscillator Configuration, T_A = 25°C

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)	Conditions
0	1	30	30–36	20	V_{CC} = 5V
0	1	30	30–36	10	V_{CC} = 5V
0	1	30	30	20 (÷20)	V_{CC} = 5V

TABLE IB. RC Oscillator Configuration, T_A = 25°C

R (kΩ)	C (pF)	CKI Freq. (MHz)	Instr. Cycle (μs)	Conditions
3.3	82	2.8–2.2	3.6 to 4.5	V_{CC} = 5V
5.6	100	1.5–1.1	6.7 to 9	V_{CC} = 5V

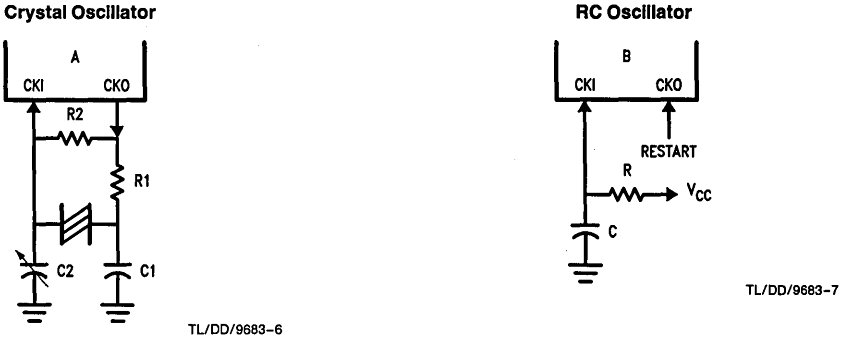
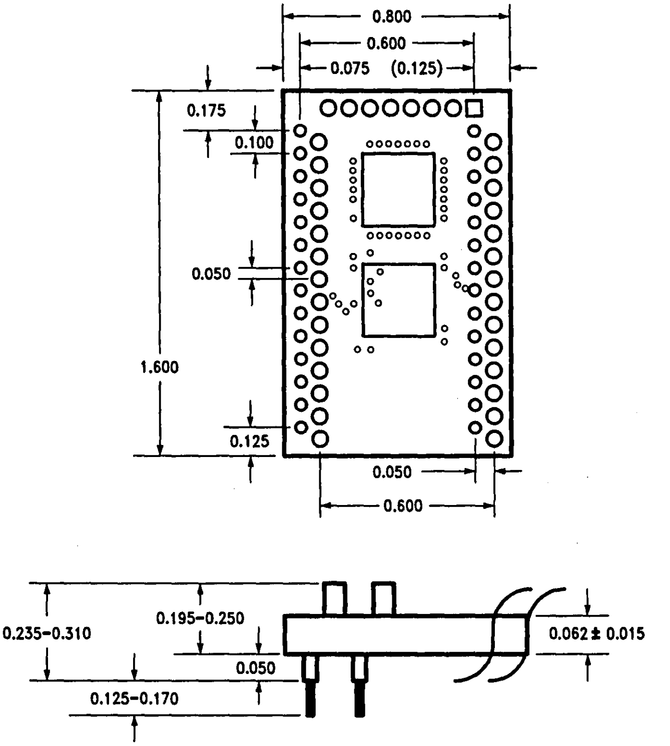


FIGURE 5. Crystal and RC Oscillator Connection Diagrams

TABLE II. Clock Options Per Package

X	Order Part Number	Clock Option
1	COP820CP-1/COP840CP-1	Crystal Oscillator Divide by 10 Option
2	COP820CP-2/COP840CP-2	External Oscillator Divide by 10 Option
3	COP820CP-3/COP840CP-3	RC Oscillator Divide by 10 Option
4	COP820CP-4/COP840CP-4	Crystal Oscillator Divide by 20 Option (High Speed)
5	COP820CP-5/COP840CP-5	External Oscillator Divide by 20 Option

COP820CP/840CP Dimensions Diagram



TL/DD/9683-8

FIGURE 6

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP820C/COP840C	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/COP840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Programmer's Manual		420410703-001

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

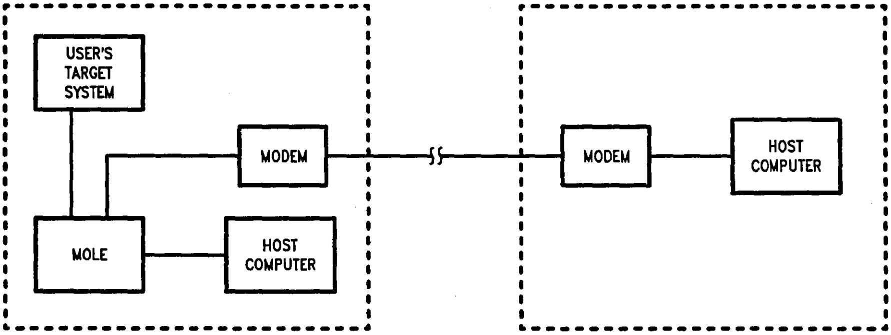
Order P/N: MOLE-DIAL-A-HLP
Information System Package Contains
DIAL-A-HELPER Users Manual P/N
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 baud	
	Set-Up:	Length:	8-bit
		Parity:	None
		Stop bit:	1
	Operation:24 hrs., 7 days		

DIAL-A-HELPER



USER SITE NATIONAL SEMICONDUCTOR SITE

TL/DD/9683-11

National Semiconductor

PRELIMINARY

COP888CLP/COP884CLP Single-Chip microCMOS Microcontroller

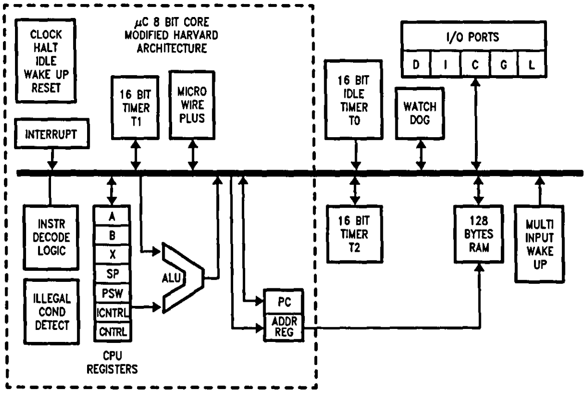
General Description

The COP888CLP and COP884CLP are piggyback versions of the COP888CL and COP884CL. These two devices are identical except that the piggyback device has been placed permanently in ROMless mode so that program memory is only accessed externally. The device package incorporates circuitry and a socket on top of the package to accommodate an EPROM such as an NMC27C64. With the addition of the EPROM, the COP888CLP and COP884CLP perform as their masked equivalent. The COP888CLP/COP884CLP are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS™ serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CLP and COP884CLP operate over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 µs per instruction rate.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS
- 1 µs instruction cycle time
- 128 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- MICROWIRE/PLUS serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
  - External Interrupt
  - Idle Timer T0
  - Two Timers (Each with 2 Interrupts)
  - MICROWIRE/PLUS
  - Multi-Input Wake Up
  - Software Trap
  - Default VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
  - Processor Independent PWM mode
  - External Event counter mode
  - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package:
  - 40 N with 33 I/O pins
  - 28 N with 21 I/O pins
- Software selectable I/O options
  - TRI-STATE® Output
  - Push-Pull Output
  - Weak Pull Up Input
  - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram



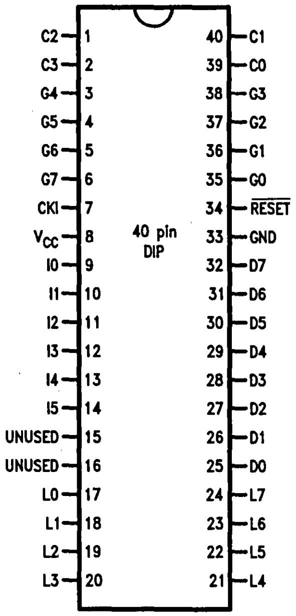
TL/DD/10419-1

FIGURE 1. COP888CLP and COP884CLP Block Diagram

COP888CLP/COP884CLP

Connection Diagrams

Dual-In-Line Package

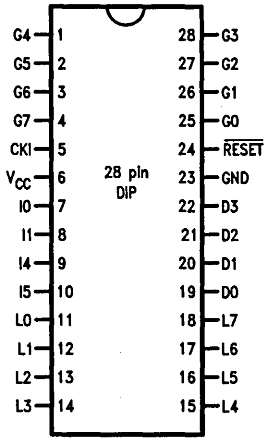


TL/DD/10419-2

Top View

Order Number COP888CLP-XE

Dual-In-Line Package



TL/DD/10419-3

Top View

Order Number COP884CLP-XE

FIGURE 2. COP888CLP and COP884CLP Connection Diagrams

2

COP888CLP and COP884CLP Pinouts for 28- and 40-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.
L0	I/O	MIWU		11	17
L1	I/O	MIWU		12	18
L2	I/O	MIWU		13	19
L3	I/O	MIWU		14	20
L4	I/O	MIWU	T2A	15	21
L5	I/O	MIWU	T2B	16	22
L6	I/O	MIWU		17	23
L7	I/O	MIWU		18	24
G0	I/O	INT		25	35
G1	WDOUT			26	36
G2	I/O	T1B		27	37
G3	I/O	T1A		28	38
G4	I/O	SO		1	3
G5	I/O	SK		2	4
G6	I	SI		3	5
G7	I/CKO	HALT Restart		4	6
D0	O			19	25
D1	O			20	26
D2	O			21	27
D3	O			22	28
I0	I			7	9
I1	I			8	10
I2	I				11
I3	I				12
I4	I			9	13
I5	I			10	14
D4	O				29
D5	O				30
D6	O				31
D7	O				32
C0	I/O				39
C1	I/O				40
C2	I/O				1
C3	I/O				2
UNUSED					16
UNUSED					15
V_{CC}				6	8
GND				23	33
CKI				5	7
$\overline{\text{RESET}}$				24	34

Note: UNUSED pins (15 and 16 on the 40-pin package) must be connected to GND.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	$-65°C$ to $+140°C$

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2)					
CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			100	mA
HALT Current (Note 3)	$V_{CC} = 5.5V$, CKI = 0 MHz			80	mA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			90	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	$V_{CC} = 5.5V$	−2		+2	μA
Input Pullup Current	$V_{CC} = 5.5V$	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	0.4			mA
Sink	$V_{CC} = 4.5V$, $V_{OL} = 1V$	10			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V$, $V_{OH} = 2.7V$	10		100	μA
Source (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage		−2		+2	μA
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current without Latchup	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

Note 4: Human body model, 100 pF through 1500Ω.

AC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator	$4.5V \le V_{CC} \le 5.5V$	1		DC	μs
R/C Oscillator	$4.5V \le V_{CC} \le 5.5V$	3		DC	μs
CKI Clock Duty Cycle (Note 5)	f_r = Max	40		60	%
Rise Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 5)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4.5V \le V_{CC} \le 5.5V$	200			ns
t_{HOLD}	$4.5V \le V_{CC} \le 5.5V$	60			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4.5V \le V_{CC} \le 5.5V$			0.7	μs
All Others	$4.5V \le V_{CC} \le 5.5V$			1	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

Note 5: Parameter sample but not 100% tested.

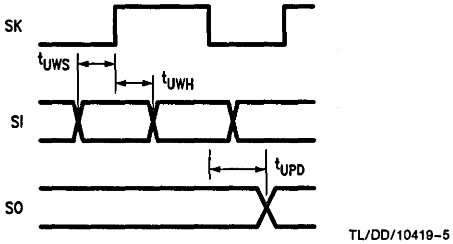
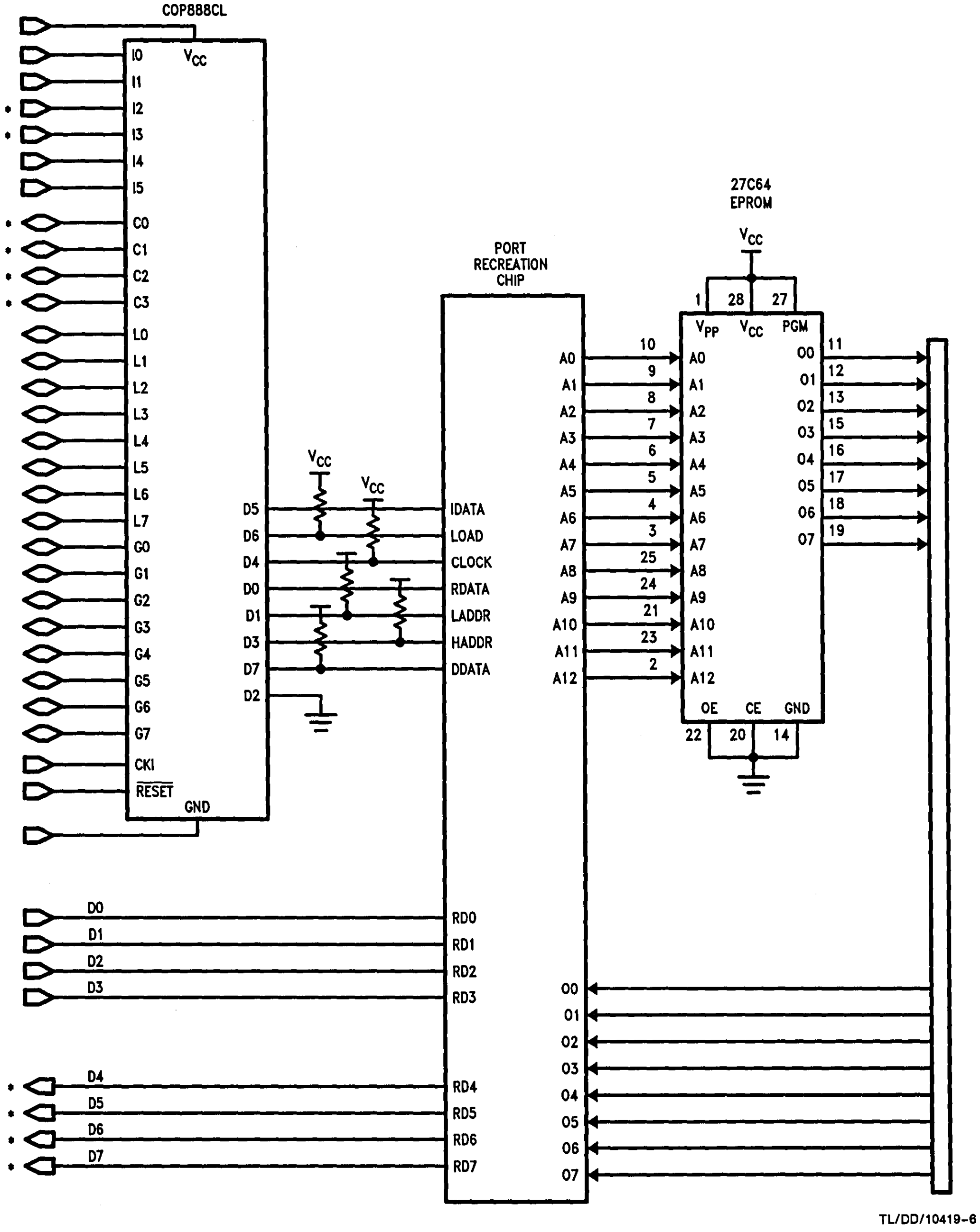


FIGURE 3. MICROWIRE/PLUS Timing

Connection Diagram



All resistors are 330Ω ±20%

(Not needed if the CKI frequency is less than 5 MHz)

*Not available on 28-pin package.

FIGURE 4

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

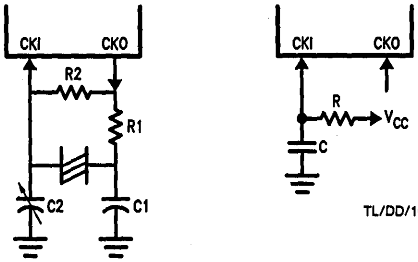
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (Special Order Only)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10419-8

TL/DD/10419-7

FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. R/C Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

EPROM Selection

The COP888CLP and COP884CLP are the piggyback versions of the COP888CL and COP884CL microcontrollers, (see Table IV). With the addition of an EPROM this part is the functional equivalent of the masked version.

Table III lists the minimum access times for a given instruction cycle time of the microcontroller. At high speeds an NMC57C64 (an 8k byte device) or any comparable EPROM must be used.

TABLE III. EPROM Selection

EPROM Minimum Access Time	COP Instruction Cycle Time
120 ns	1.00 μs
150 ns	1.10 μs
200 ns	1.27 μs
250 ns	1.44 μs
300 ns	1.60 μs
400 ns	1.94 μs

TABLE IV. Options

Order Part Number	Options
COP888CLP-XE	Crystal Oscillator Divide by 10 with Halt Enabled. This is identical to the masked COP888CL and COP884CL with Option 1 = 1; Option 2 = 1.

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-001

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information System and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

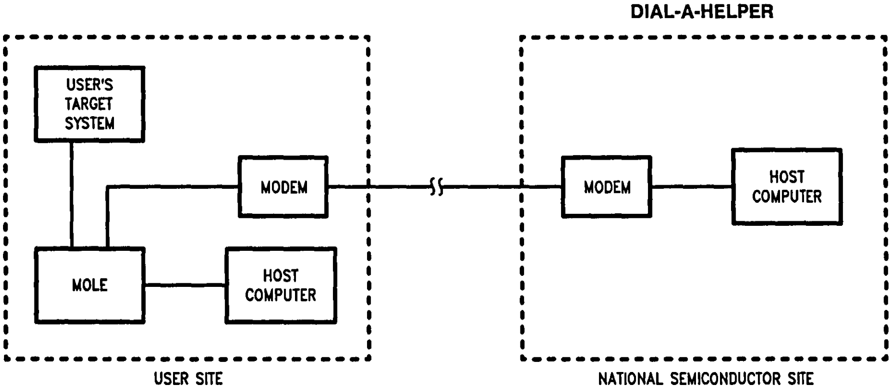
Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains
DIAL-A-HELPER User Manual P/N
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

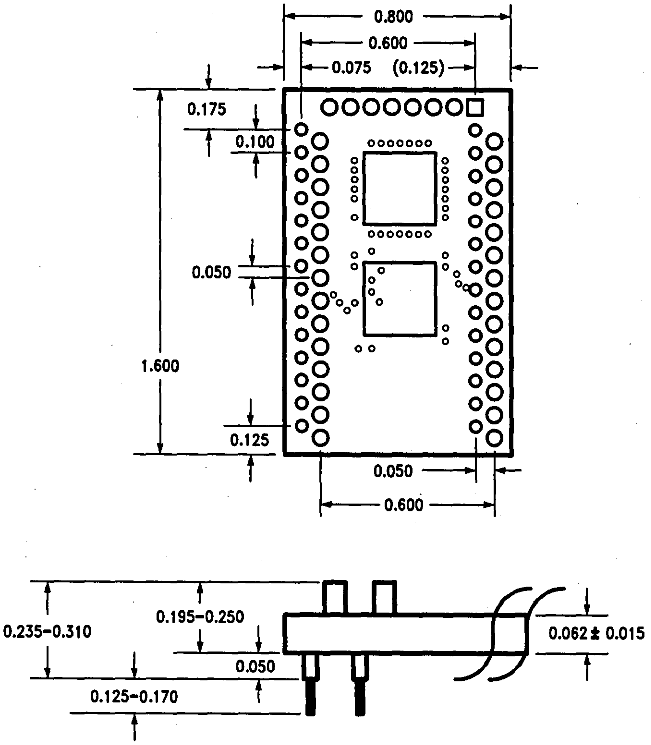
Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 Baud
Set-Up: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hours, 7 Days



TL/DD/10419-9

COP884CLP/COP888CLP Dimensions Diagram

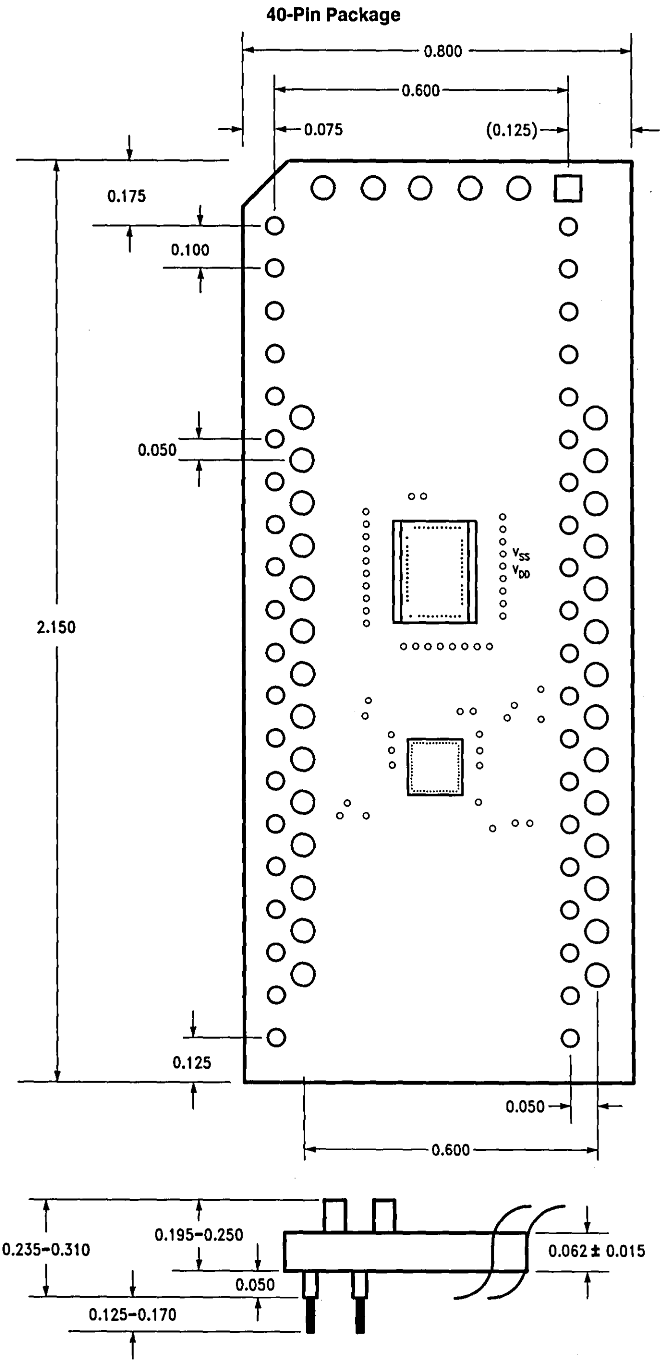
28-Pin Package



TL/DD/10419-10

FIGURE 6

COP888CLP Dimensions Diagram



TL/DD/10419–11

FIGURE 7

National Semiconductor

PRELIMINARY

COP888CFP/COP884CFP Single-Chip microCMOS Microcontroller

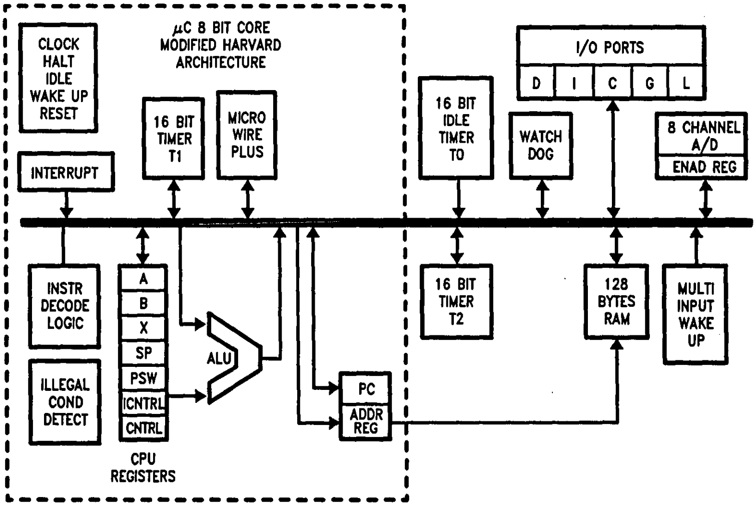
General Description

The COP888CFP and COP884CFP are piggyback versions of the COP888CF and COP884CF. These two devices are identical except that the piggyback device has been placed permanently in ROMless mode so that program memory is only accessed externally. The device package incorporates circuitry and a socket on top of the package to accommodate a piggyback EPROM such as an NMC27C64. With the addition of the EPROM, the COP888CFP/COP884CFP perform like their masked equivalent. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS
- 1 µs instruction cycle time
- 128 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- 8-channel A/D converter with prescaler and both differential and single ended modes
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- Ten multi-source vectored interrupts servicing
  - External Interrupt
  - Idle Timer T0
  - Two Timers (each with 2 Interrupts)
  - MICROWIRE/PLUS
  - Multi-Input Wake Up
  - Software Trap
  - Default VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
  - Processor Independent PWM mode
  - External Event counter mode
  - Input Capture mode
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 40 N or 28 N
  - 40 N with 33 I/O pins
  - 28 N with 21 I/O pins
- Software selectable I/O options
  - TRI-STATE® Output
  - Push-Pull Output
  - Weak Pull Up Input
  - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram



TL/DD/10420–1

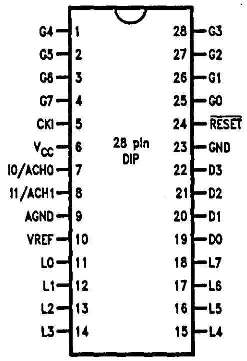
FIGURE 1. COP888CFP Block Diagram

General Description (Continued)

The COP888CFP and COP884CFP are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, two 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), an 8-channel, 8-bit A/D converter with both differential and single ended modes, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CF and COP884CFP operate over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 µs per instruction rate.

Connection Diagrams

Dual-In-Line Package

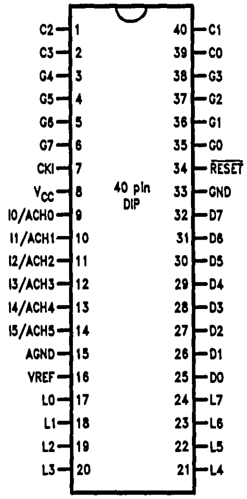


TL/DD/10420-3

Top View

Order Number COP884CPF-XE

Dual-In-Line Package



TL/DD/10420-2

Top View

Order Number COP888CPF-XE

FIGURE 2. COP888CFP Connection Diagrams

COP888CFP Pinouts for 28- and 40-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.
L0	I/O	MIWU		11	17
L1	I/O	MIWU		12	18
L2	I/O	MIWU		13	19
L3	I/O	MIWU		14	20
L4	I/O	MIWU	T2A	15	21
L5	I/O	MIWU	T2B	16	22
L6	I/O	MIWU		17	23
L7	I/O	MIWU		18	24
G0	I/O	INT		25	35
G1	WDOUT			26	36
G2	I/O	T1B		27	37
G3	I/O	T1A		28	38
G4	I/O	SO		1	3
G5	I/O	SK		2	4
G6	I	SI		3	5
G7	I/CKO	HALT Restart		4	6
D0	O			19	25
D1	O			20	26
D2	O			21	27
D3	O			22	28
I0	I	ACH0		7	9
I1	I	ACH1		8	10
I2	I	ACH2			11
I3	I	ACH3			12
I4	I	ACH4			13
I5	I	ACH5			14
D4	O				29
D5	O				30
D6	O				31
D7	O				32
C0	I/O				39
C1	I/O				40
C2	I/O				1
C3	I/O				2
V_{REF}	$+V_{REF}$			10	16
AGND	AGND			9	15
V_{CC}				6	8
GND				23	33
CKI				5	7
$\overline{\text{RESET}}$				24	34

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2) CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			100	mA
HALT Current (Note 3)	V_{CC} = 5.5V, CKI = 0 MHz			80	mA
IDLE Current CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			90	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 5.5V	−2		+2	µA
Input Pullup Current	V_{CC} = 5.5V	40		250	µA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 2.7V	10		100	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
TRI-STATE Leakage		−2		+2	µA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. If the A/D is not being used and minimum standby current is desired, V_{REF} should be tied to AGND (effectively shorting the Reference resistor). The clock monitor is disabled.

Note 4: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All others				3	mA
Maximum Input Current without Latchup (Note 7)	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications $V_{CC} = 5V \pm 10\%$ $(V_{SS} - 0.050V) \leq$ Any Input $\leq (V_{CC} + 0.050V)$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			±1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the Best Straight Line			±½	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			±½	LSB
Input Reference Resistance		1.6		4.8	kΩ
Common Mode Input Range (Note 8)		AGND		V_{REF}	V
DC Common Mode Error				±¼	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
A/D Clock Frequency (Note 6)		0.1		1.67	MHz
Conversion Time (Note 5)			12		A/D Clock Cycles

Note 5: Conversion Time includes sample and hold time.

Note 6: See Prescaler description.

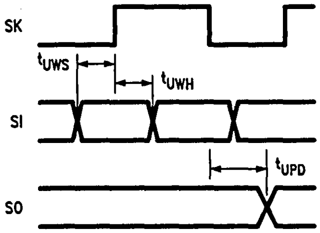
Note 7: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

Note 8: For $V_{IN}(-) \geq V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct for analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

AC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator	$4.5V \leq V_{CC} \leq 5.5V$	1		DC	μs
R/C Oscillator	$4.5V \leq V_{CC} \leq 5.5V$	3		DC	μs
CKI Clock Duty Cycle (Note 10)	f_r = Max	40		60	%
Rise Time (Note 10)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 10)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}	$4.5V \leq V_{CC} \leq 5.5V$	200			ns
t_{HOLD}	$4.5V \leq V_{CC} \leq 5.5V$	60			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK	$4.5V \leq V_{CC} \leq 5.5V$			0.7	μs
All Others	$4.5V \leq V_{CC} \leq 5.5V$			1	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs

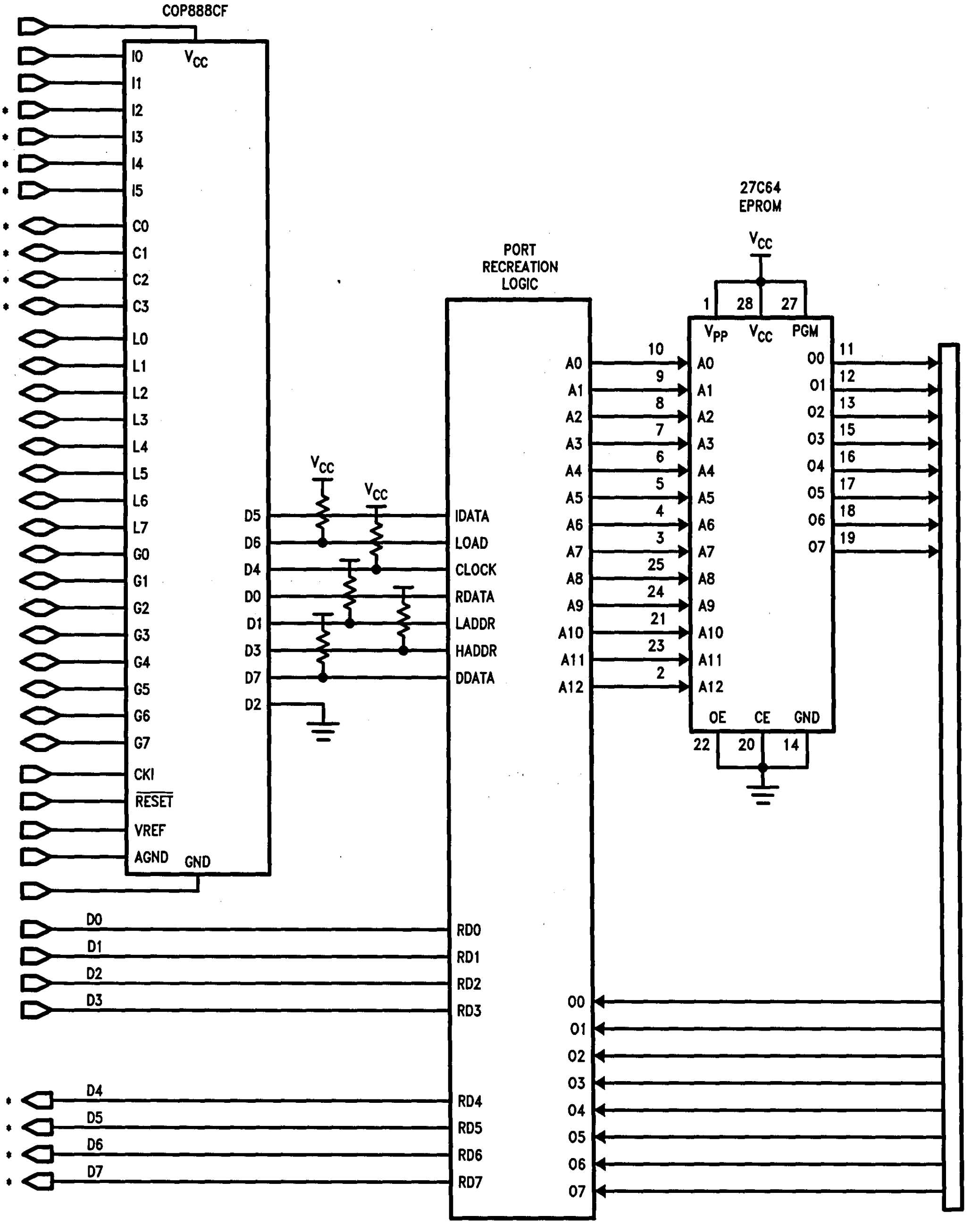
Note 10: Parameter sample but not 100% tested.



TL/DD/10420-5

FIGURE 3. MICROWIRE/PLUS Timing

Connection Diagram



TL/DD/10420-6

All resistors are 330Ω ±20%

(Not needed if the CKI frequency is less than 5 MHz)

*Not available on 28-pin package.

FIGURE 4

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

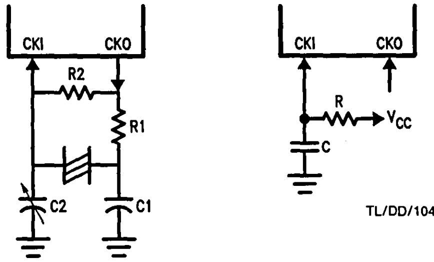
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (SPECIAL ORDER ONLY)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10420-8

TL/DD/10420-7

FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration
$T_A = 25°C, V_{CC} = 5V$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. R/C Oscillator Configuration
$T_A = 25°C, V_{CC} = 5V$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

EPROM Selection

The COP888CFP and COP884CFP are the piggyback versions of the COP888CF and COP884CF microcontrollers, (see Table IV). With the addition of an EPROM this part is the functional equivalent of the masked version.

Table III lists the minimum access times for a given instruction cycle time of the microcontroller. At high speeds an NMC57C64 (an 8k byte device) or any comparable EPROM must be used.

TABLE III. EPROM Selection

EPROM Minimum Access Time	COP Instruction Cycle Time
120 ns	1.00 μs
150 ns	1.10 μs
200 ns	1.27 μs
250 ns	1.44 μs
300 ns	1.60 μs
400 ns	1.94 μs

TABLE IV. Clock Options per Package

Order Part Number	Clock Option
COP888CFP-XE	Crystal Oscillator Divide by 10 with Halt Enabled, this is identical to the masked COP888CF and COP884CF with Option 1 = 1; Option 2 = 1.

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations. It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

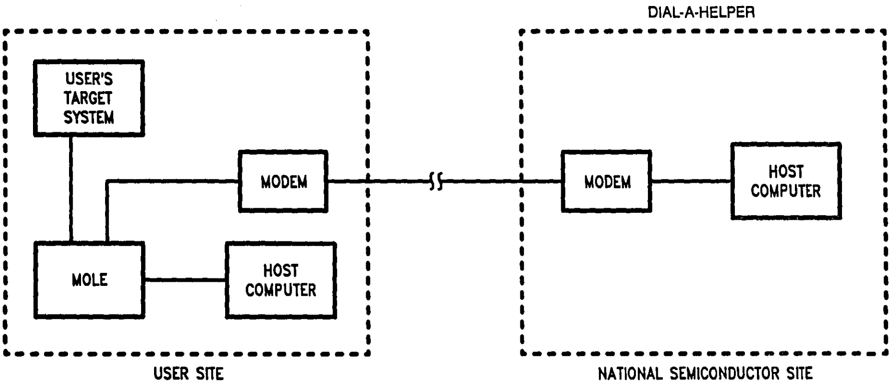
Order P/N: MOLE-DIAL-A-HLP

Information System Package Contents
  Dial-A-Helper User Manual
  Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

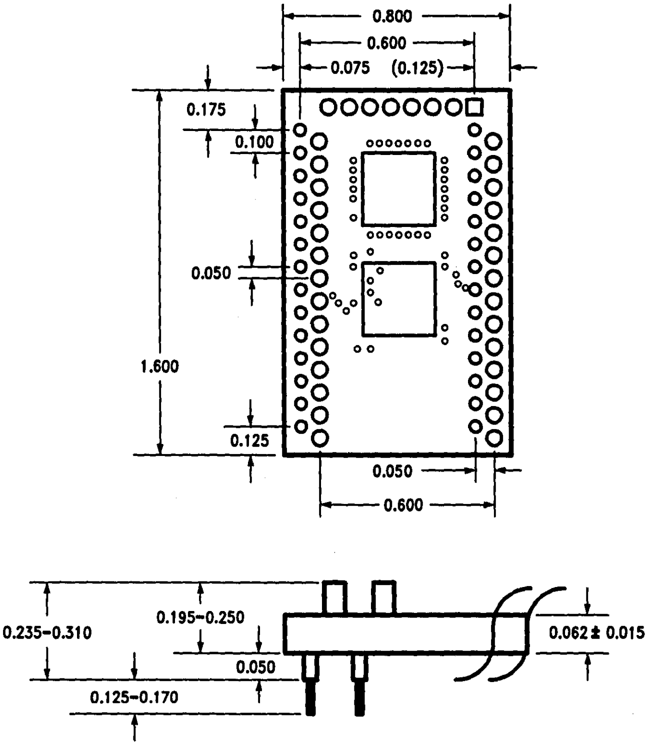
Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 Baud	
	Set-Up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hours, 7 Days	



TL/DD/10420-9

COP888CFP Dimension Diagrams

28-Pin Package

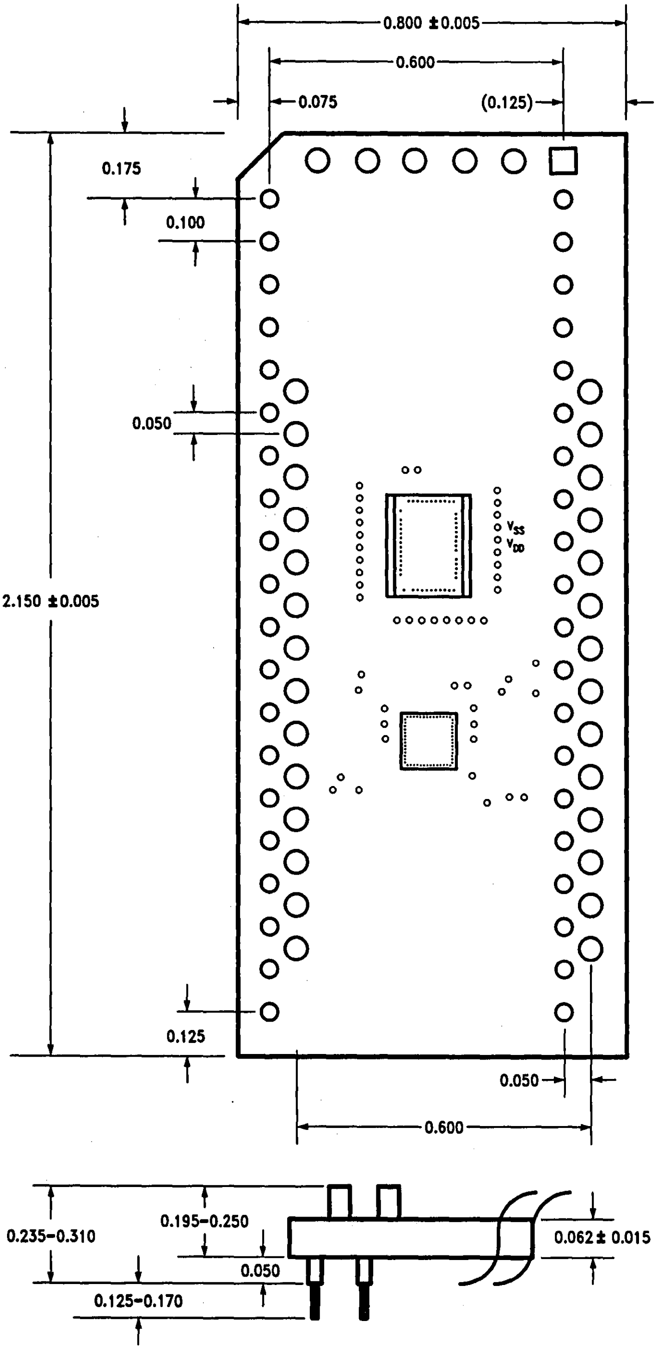


TL/DD/10420–10

FIGURE 6

COP888CFP Dimension Diagrams (Continued)

40-Pin Package



TL/DD/10420–11

FIGURE 7

National Semiconductor

PRELIMINARY

COP888CGP/COP884CGP Single-Chip microCMOS Microcontroller

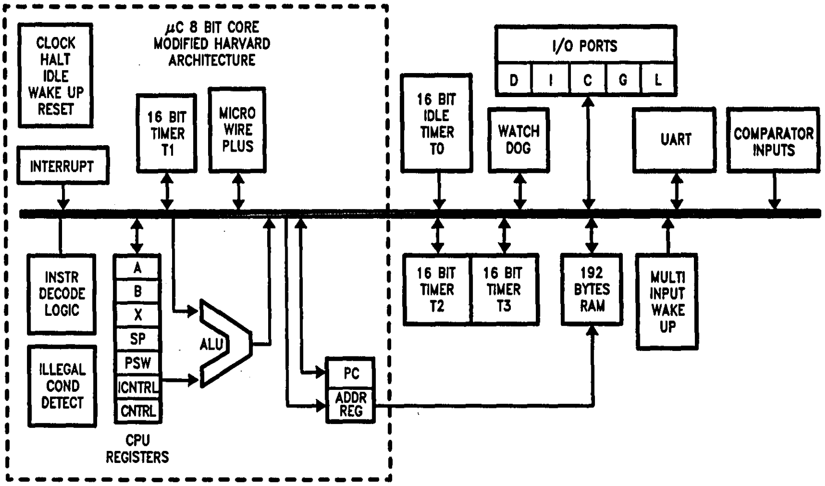
General Description

The COP888CGP and COP884CGP are piggyback versions of the COP888CG and COP884CG. These two devices are identical except that the piggyback device has been placed permanently in ROMless mode so that program memory is only accessed externally. The device package incorporates circuitry and a socket on top of the package to accommodate a piggyback EPROM such as an NMC27C64. With the addition of the EPROM, the COP888CGP and COP884CGP perform like their masked equivalent. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS
- 1 µs instruction cycle time
- 192 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- Full duplex UART
- Two analog comparators
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set
- Fourteen multi-source vectored interrupts servicing
  - External Interrupt
  - Idle Timer T0
  - Three Timers (each with 2 interrupts)
  - MICROWIRE/PLUS
  - Multi-Input Wake Up
  - Software Trap
  - UART (2)
  - Default VIS
- Three 16-bit timers, each with two 16-bit registers supporting:
  - Processor Independent PWM mode
  - External Event counter mode
  - Input Capture mode
- True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 40 N or 28 N
  - 40 N with 35 I/O pins
  - 28 N with 23 I/O pins
- Software selectable I/O options
  - TRI-STATE® Output
  - Push-Pull Output
  - Weak Pull Up Input
  - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram



TL/DD/10421–1

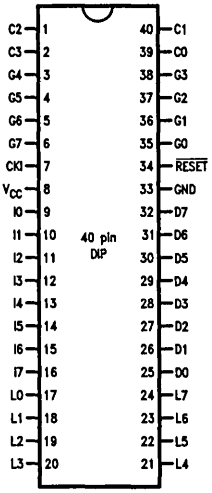
FIGURE 1. COP888CGP and COP884CGP Block Diagram

General Description (Continued)

The COP888CGP and COP884CGP are fully static parts, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CGP and COP884CGP operate over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate.

Connection Diagrams

Dual-In-Line Package

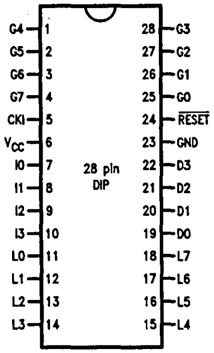


TL/DD/10421-2

Top View

Order Number COP888CGP-E

Dual-In-Line Package



TL/DD/10421-3

Top View

Order Number COP884CGP-E

FIGURE 2. COP888CGP and COP884CGP Connection Diagrams

Connection Diagrams (Continued)

COP888CGP and COP884CGP Pinouts for 28- and 40-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	28-Pin Pack.	40-Pin Pack.
L0	I/O	MIWU		11	17
L1	I/O	MIWU	CKX	12	18
L2	I/O	MIWU	TDX	13	19
L3	I/O	MIWU	RDX	14	20
L4	I/O	MIWU	T2A	15	21
L5	I/O	MIWU	T2B	16	22
L6	I/O	MIWU	T3A	17	23
L7	I/O	MIWU	T3B	18	24
G0	I/O	INT		25	35
G1	WDOUT			26	36
G2	I/O	T1B		27	37
G3	I/O	T1A		28	38
G4	I/O	SO		1	3
G5	I/O	SK		2	4
G6	I	SI		3	5
G7	I/CKO	HALT Restart		4	6
D0	O			19	25
D1	O			20	26
D2	O			21	27
D3	O			22	28
I0	I			7	9
I1	I	COMP1IN−		8	10
I2	I	COMP1IN+		9	11
I3	I	COMP1OUT		10	12
I4	I	COMP2IN−			13
I5	I	COMP2IN+			14
I6	I	COMP2OUT			15
I7	I				16
D4	O				29
D5	O				30
D6	O				31
D7	O				32
C0	I/O				39
C1	I/O				40
C2	I/O				1
C3	I/O				2
V_{CC}				6	8
GND				23	33
CKI				5	7
$\overline{\text{RESET}}$				24	34

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 4)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 1)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 2) CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			100	mA
HALT Current (Note 3)	V_{CC} = 5.5V, CKI = 0 MHz			80	mA
IDLE Current CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			90	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 5.5V	−2		+2	µA
Input Pullup Current	V_{CC} = 5.5V	40		250	µA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 2.7V	10		100	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
TRI-STATE Leakage		−2		+2	µA

Note 1: Rate of voltage change must be less then 0.5 V/ms.

Note 2: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 3: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor and the comparators are disabled.

Note 4: Human body model, 100 pF through 1500Ω.

2

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All others				 15 3	 mA mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

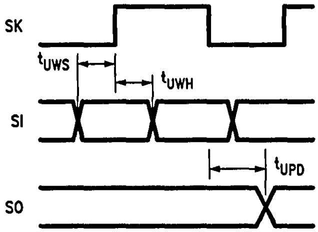
Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator R/C Oscillator	 $4V \le V_{CC} \le 6V$ $4V \le V_{CC} \le 6V$	 1 3		 DC DC	 µs µs
CKI Clock Duty Cycle (Note 5) Rise Time (Note 5) Fall Time (Note 5)	f_r = Max f_r = 10 MHz Ext Clock f_r = 10 MHz Ext Clock	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}	 $4.5V \le V_{CC} \le 5.5V$ $4.5V \le V_{CC} \le 5.5V$	 200 60			 ns ns
Output Propagation Delay t_{PD1}, t_{PD0} SO, SK All Others	$R_L = 2.2k, C_L = 100$ pF $4.5V \le V_{CC} \le 5.5V$ $4.5V \le V_{CC} \le 5.5V$			 0.7 1	 µs µs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		 220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		 1 1 1 1			 t_c t_c t_c t_c
Reset Pulse Width		1			µs

Note 5: Parameter sampled but not 100% tested.

Note 6: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25°C$

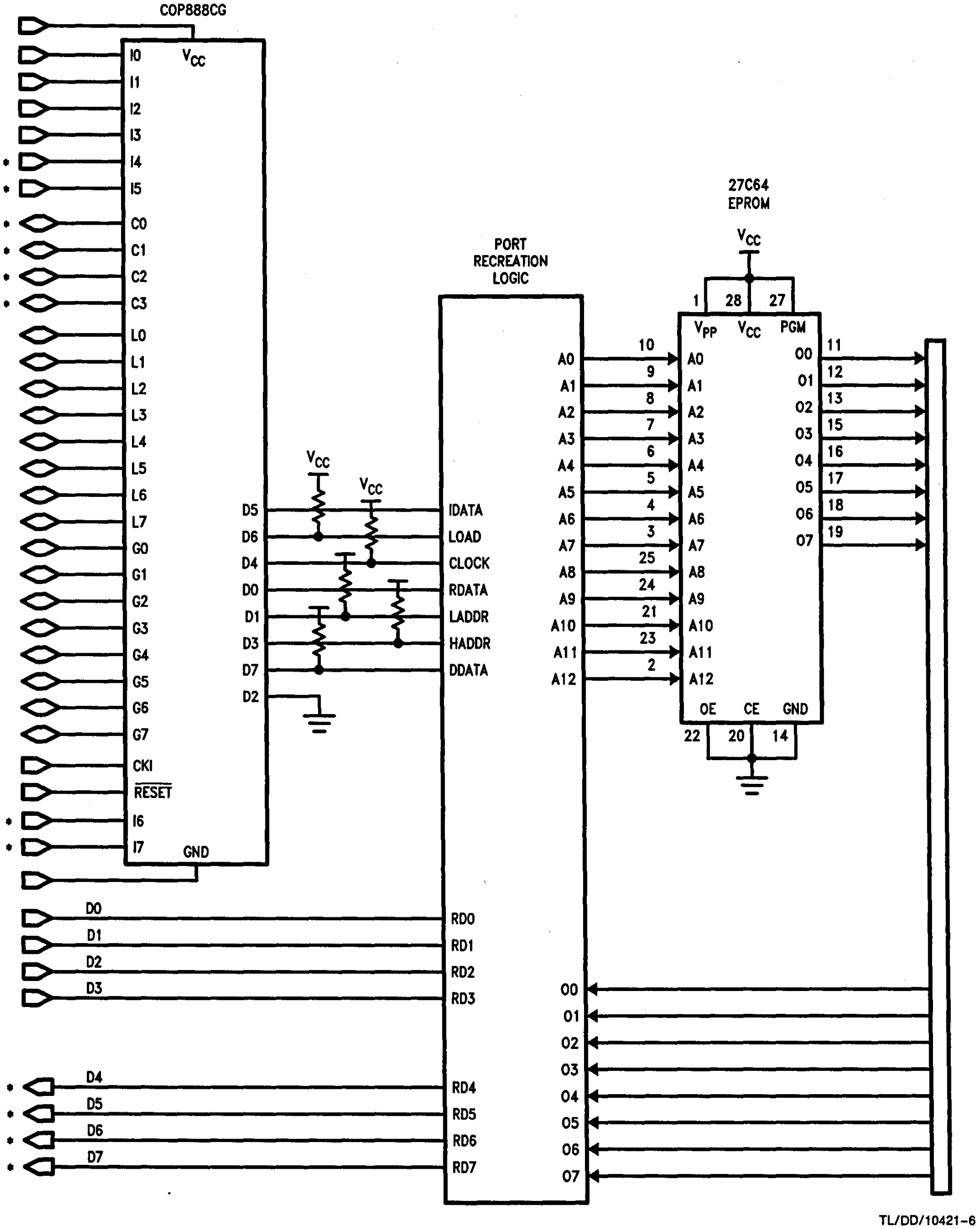
Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		10	25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current Per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



TL/DD/10421-5

FIGURE 3. MICROWIRE/PLUS Timing

Connection Diagram



TL/DD/10421-6

All resistors are 330Ω ±20%

(Not needed if the CKI frequency is less than 5 MHz)

*Not available on 28-pin package.

FIGURE 4

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

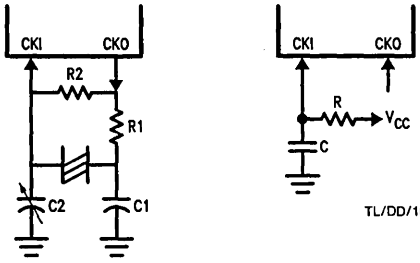
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR (Special Order Only)

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart pin.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10421-8

TL/DD/10421-7

FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration
$T_A = 25°C, V_{CC} = 5V$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. R/C Oscillator Configuration
$T_A = 25°C, V_{CC} = 5V$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

EPROM Selection

The COP888CGP and COP884CGP are the piggyback versions of the COP888CG and COP884CG microcontrollers, (see Table IV). With the addition of an EPROM this part is the functional equivalent of the masked version.

Table III lists the minimum access times for a given instruction cycle time of the microcontroller. At high speeds an NMC57C64 (an 8k byte device) or any comparable EPROM must be used.

TABLE III. EPROM Selection

EPROM Minimum Access Time	COP Instruction Cycle Time
120 ns	1.00 μs
150 ns	1.10 μs
200 ns	1.27 μs
250 ns	1.44 μs
300 ns	1.60 μs
400 ns	1.94 μs

TABLE IV. Options

Order Part Number	Options
COP888CGP-E	Crystal Oscillator Divide by 10 with Halt Enabled. This is identical to the mask COP888CG and COP884CG with Option 1 = 1 and Option 2 = 1.

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs™ microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-001

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

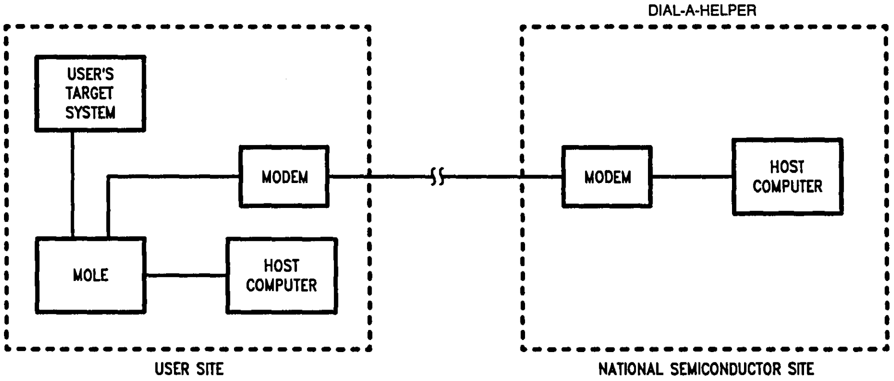
Order P/N: MOLE-DIAL-A-HLP

Information System Package Contents
- Dial-A-Helper User Manual
- Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

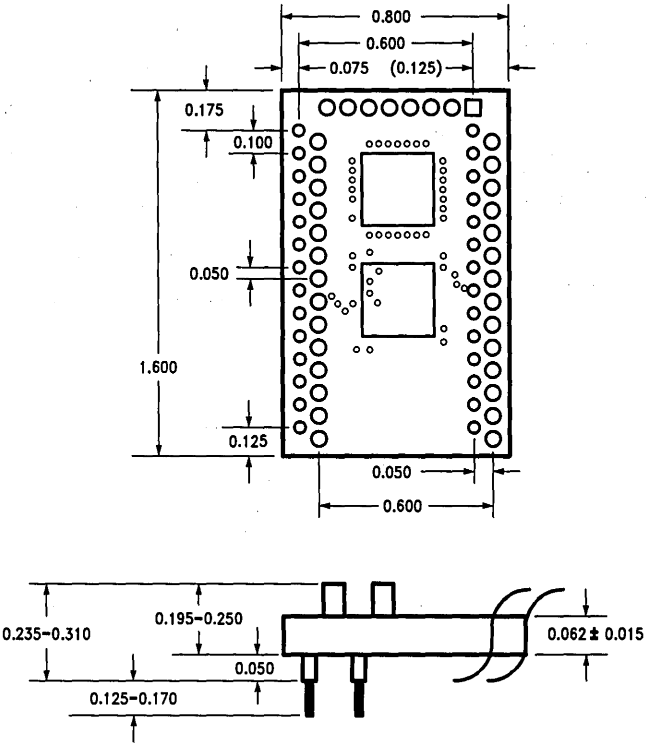
Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 Baud	
	Set-Up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hours, 7 Days	



TL/DD/10421-9

COP888CGP/COP884CGP Dimension Diagrams

28-Pin Package

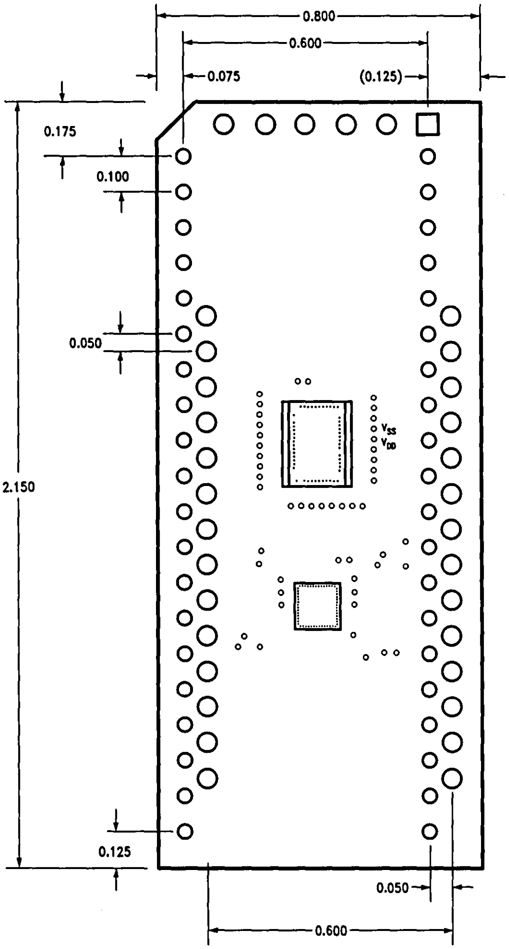


TL/DD/10421–10

FIGURE 6

COP888CGP Dimension Diagrams (Continued)

40-Pin Package



TL/DD/10421-11

FIGURE 7

National Semiconductor

PRELIMINARY

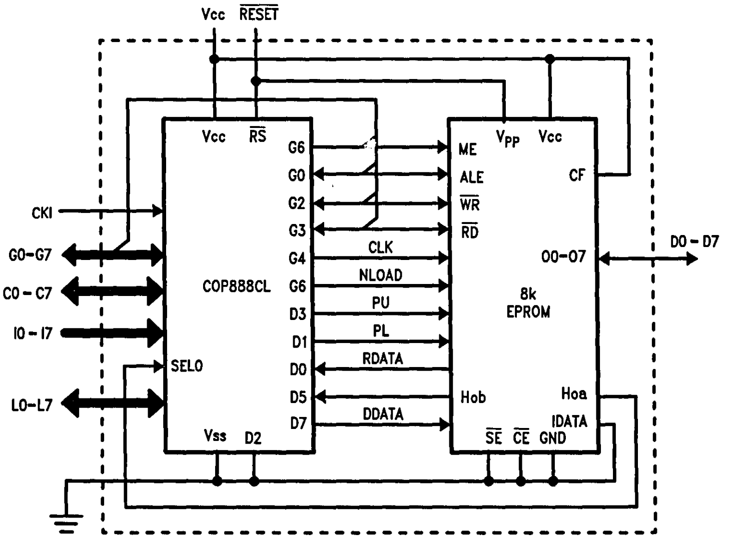
COP888CLMH
Single-Chip microCMOS Microcontroller

General Description

The COP888CLMH hybrid emulator is a member of the COPS™ microcontroller family. It is functionally identical to the COP888CL except that its package contains an 8k EPROM in place of masked program ROM. This 44-pin part contains a transparent window which allows the EPROM to be erased and re-programmed. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 8192 bytes on-board EPROM
- 128 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Two 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- Ten multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Two Timers each with 2 interrupts
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — Default VIS
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set with true bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC with 37 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CG
- Real time emulation and full program debug offered by National's Development Systems



TL/DD/10467–1

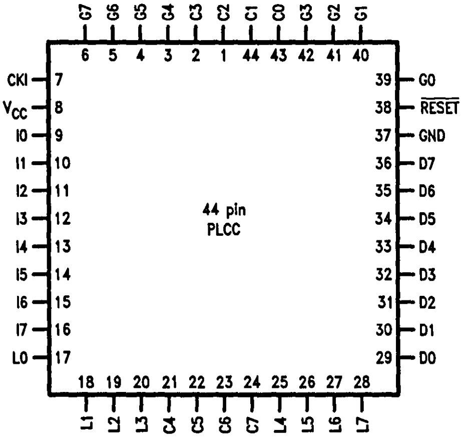
FIGURE 1. COP888CLMH Block Diagram

General Description (Continued)

The COP888CLMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CLMH operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate.

Connection Diagram

Plastic Chip Carrier



TL/DD/10467-2

Top View

FIGURE 2. COP888CLMH Connection Diagram

COP888CLMH Pinouts

Port	Type	Alt. Fun	Alt. Fun	MUX Mode	44-Pin PCC
L0	I/O	MIWU			17
L1	I/O	MIWU			18
L2	I/O	MIWU			19
L3	I/O	MIWU			20
L4	I/O	MIWU	T2A		25
L5	I/O	MIWU	T2B		26
L6	I/O	MIWU			27
L7	I/O	MIWU			28
G0	I/O	INT		ALE	39
G1	WDOUT				40
G2	I/O	T1B		$\overline{WR}$	41
G3	I/O	T1A		$\overline{RD}$	42
G4	I/O	SO			3
G5	I/O	SK			4
G6	I	SI		ME	5
G7	I/CKO	HALT RESTART			6
D0	O			I/O BIT 0	29
D1	O			I/O BIT 1	30
D2	O			I/O BIT 2	31
D3	O			I/O BIT 3	32
I0	I				9
I1	I				10
I2	I				11
I3	I				12
I4	I				13
I5	I				14
I6	I				15
I7	I				16
D4	O			I/O BIT 4	33
D5	O			I/O BIT 5	34
D6	O			I/O BIT 6	35
D7	O			I/O BIT 7	36
C0	I/O				43
C1	I/O				44
C2	I/O				1
C3	I/O				2
C4	I/O				21
C5	I/O				22
C6	I/O				23
C7	I/O				24
V_{CC}					8
GND					37
CKI					7
$\overline{RESET}$				V_{PP}	38

I/O BITS = Address and Data Lines

MUX MODE = Programming Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin (Note 1)	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 5)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics 0°C ≤ T_A ≤ +70°C unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 2)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 3)					
CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 μs			25	mA
HALT Current (Note 4)	V_{CC} = 5.5V, CKI = 0 MHz		200		μA
IDLE Current					
CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 μs			15	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 5.5	−2		+2	μA
Input Pullup Current	V_{CC} = 5.5V	40		250	μA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 2.7V	10		100	μA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
TRI-STATE Leakage	V_{CC} = 4.5V	−2		+2	μA

Note 1: Except pins G6 (ME) and the RESET (V_{PP}) pin during EPROM MUX mode programming at which time the absolute maximum voltage is 14V on these two pins.

Note 2: Rate of voltage change must be less then 0.5 V/ms.

Note 3: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 4: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The clock monitor is disabled.

Note 5: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified (Continued)

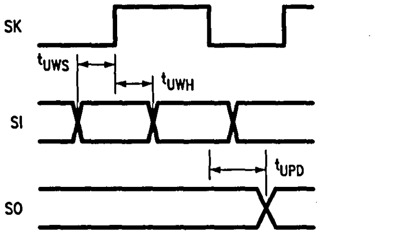
Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin					
D Outputs (Sink)				15	mA
All Others				3	mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator,		1		DC	µs
R/C Oscillator		3		DC	µs
CKI Clock Duty Cycle (Note 7)	f_r = Max	40		60	%
Rise Time (Note 7)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 7)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}		200			ns
t_{HOLD}		60			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK				0.7	µs
All Others				1	µs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			µs

Note 6: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

Note 7: Parameter sampled but not 100% tested.



TL/DD/10467-3

FIGURE 3. MICROWIRE/PLUS Timing

Pin Descriptions

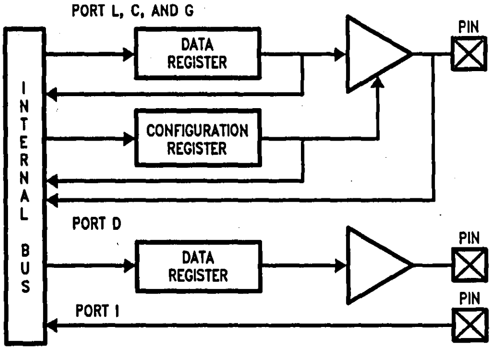
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset Description section.

The COP888CLMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CLMH memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CLMH. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/10467-4

FIGURE 4. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L4 and L5 are used for the timer input functions T2A and T2B.

The Port L has the following alternate features:

L0	MIWU
L1	MIWU
L2	MIWU
L3	MIWU
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU
L7	MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

When programming the COP888CLMH, G0 becomes Address Latch Enable (ALE) and pins G2, G3 and G6 become Write bar ($\overline{\text{WR}}$), Read bar ($\overline{\text{RD}}$) and Mux Enable (ME), respectively.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0	INTR (External Interrupt Input)
G2	T1B (Timer T1 Capture Input)
G3	T1A (Timer T1 I/O)
G4	SO (MICROWIRE Serial Data Output)
G5	SK (MICROWIRE Serial Clock)
G6	SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1	WDOUT WatchDog and/or Clock Monitor dedicated output
G7	CKO Oscillator dedicated output or general purpose input

Pin Descriptions (Continued)

When programming the COP888CLMH, G0 becomes Address Latch Enable (ALE) and pins G2, G3 and G6 become Write ($\overline{\text{WR}}$), Read ($\overline{\text{RD}}$) and Mux Enable (ME), respectively.

Port I is an eight-bit input port. The 28- and 40-pin devices do not have a full complement of Port I pins. The unavailable pins are not terminated i.e., they are floating. A read operation for these unterminated pins will return unpredictable values. The user must ensure that the software takes this into account by either masking or restricting the accesses to bit operations. The unterminated Port I pins will draw power only when addressed.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

I1	COMP1−IN (Comparator 1 Negative Input)
I2	COMP1+IN (Comparator 1 Positive Input)
I3	COMP1OUT (Comparator 1 Output)
I4	COMP2−IN (Comparator 2 Negative Input)
I5	COMP2+IN (Comparator 2 Positive Input)
I6	COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when $\overline{\text{RESET}}$ goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

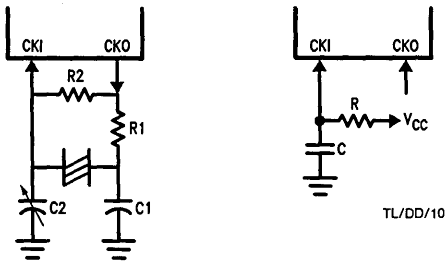
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10467-5

TL/DD/10467-6

FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. RC Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP888CLMH

The COP888CLMH is a hybrid part consisting of a COP888CG die and an 8k EPROM die with port re-creation logic. In order to access the EPROM, the COP888CG die has been placed in ROMless mode by holding its D2 pad at GND. The other D-lines of the COP888CG die are used to communicate with the EPROM. All 8 D-lines are recreated internally and appear on the pins of the COP888CLMH as normal D outputs.

When programming the COP888CLMH, a multiplexed method (MUX MODE) is used. To enter this mode a voltage of 12.2V–13V is applied to pin ME (pin G6). The 8 D-Port pins on the COP888CLMH become address bits with a low to high transition on ALE (pin G0), and data bits with a high to low transition on $\overline{\text{WR}}$ (pin G2). With a low to high transition on $\overline{\text{RD}}$ (pin G3), the data being programmed is verified. On the 28-pin part, address and data bits 4 to 7 are accessed via L4 to L7. The following steps must be followed in order to place the part in programming mode:

1. Apply V_{CC} = 5V.
2. Ground the RESET and CKI pins. This puts the COP outputs in TRI-STATE mode.
3. Apply V_{PP} = 12.2V–13V to pin G6. This places the EPROM in MUX mode. The current requirement is 450 μA maximum (V_{IN} = 13V, V_{CC} = 5.0V, −40°C).
4. Apply 12.2V–13V to the RESET pin. This supplies V_{PP} to the EPROM which requires 30 mA maximum during WRITE pulses. It is permissible for V_{PP} to drop to V_{CC} during the READ pulses as is done on some programmers.
5. Begin programming each EPROM byte interactively. (See *Figures 6* and *7*). The interactive programming algorithm programs a byte with a 0.5 ms pulse, and then does a verify to determine if that byte was fully programmed. If it was not, the program pulse is repeated and verified again. This is done up to a maximum of 20 times, but most bytes will program with a single pulse.

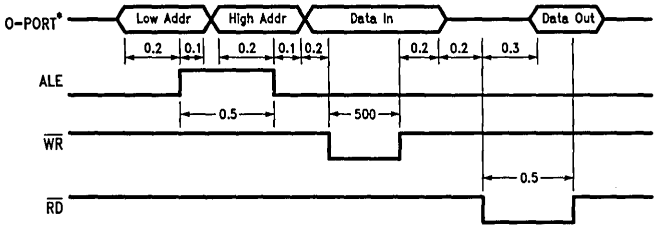
Programming the COP888CLMH (Continued)

Erasure of program memory is achieved by removing the part from its socket and exposing the transparent window to an ultra-violet light source.

Note: The last byte of program memory (EPROM location 01FFF HEX) must contain one of two values: 07F HEX for the HALT enabled mode, or 0FF HEX for the HALT disabled mode. The COP888CGLH will not function properly if any other value resides in this last byte location.

Note: V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}. The EPROM must not be inserted into or removed from a board with voltage applied to V_{PP} or V_{CC}.

The maximum absolute allowable voltage which may be applied to the G6 (ME) and RESET (V_{PP}) pins during programming is 14V. Care must be taken when switching the V_{PP} supply to prevent any overshoot from exceeding this 14V limit. At least a 0.1 µF capacitor is required across V_{PP} to GND and V_{CC} to GND to suppress spurious voltage transients which may damage the device.

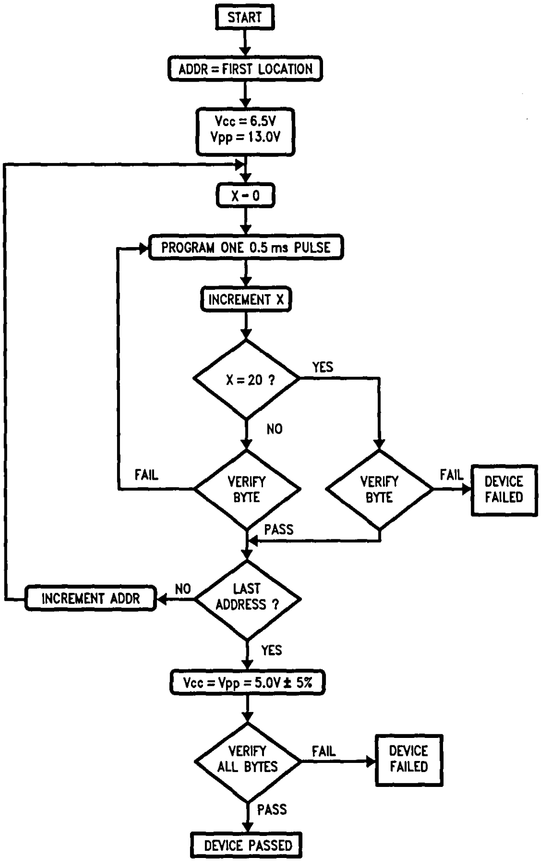


TL/DD/10467-7

Note: All minimum times are in µs.

* O-Port = D0 to D7

FIGURE 6. COP888CLMH MUX Mode Programming Timing Diagram



TL/DD/10467-8

FIGURE 7. COP888CLMH MUX Mode Programming Flow Chart

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

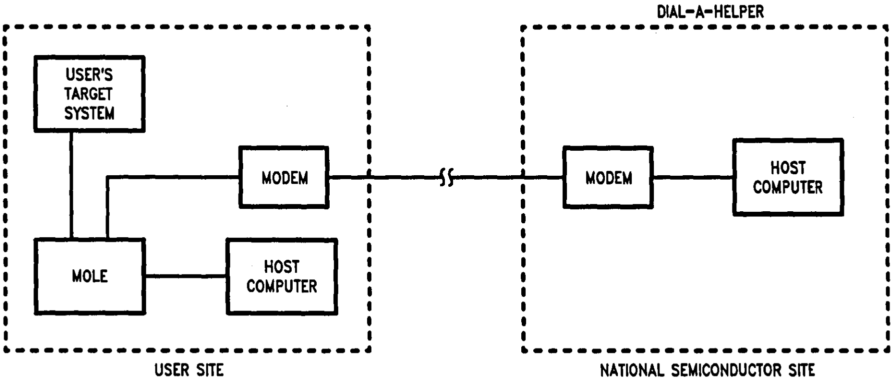
ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 Baud
Set-up: Length: 8-Bit
Parity: None
Stop Bit: 1
Operation: 24 Hrs., 7 Days



TL/DD/10467-9

National Semiconductor

PRELIMINARY

COP888CFMH Single-Chip microCMOS Microcontroller

General Description

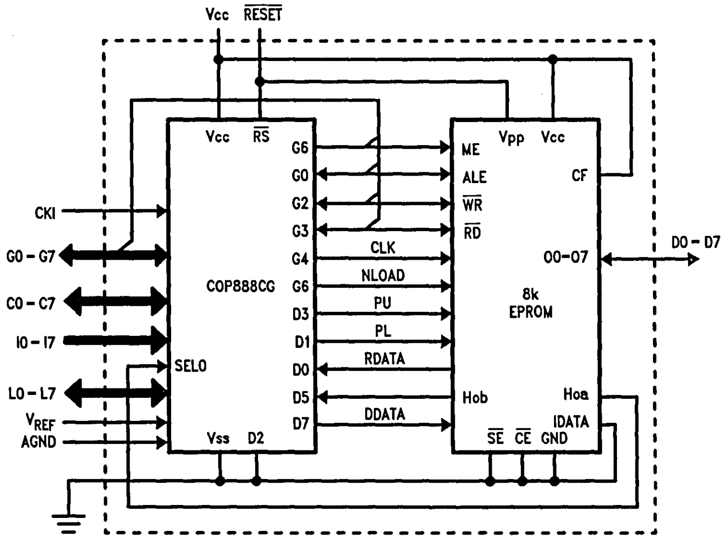
The COP888CFMH hybrid emulator is a member of the COPS™ microcontroller family. It is functionally identical to the COP888CF except that its package contains an 8k EPROM in place of masked program ROM. This 44-pin part contains a transparent window which allows the EPROM to be erased and re-programmed.

The COP88CFMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PULSE serial I/O, two 16-bit timer/counter supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), an 8-channel, 8-bit A/D converter with both differential and single ended modes, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CFMH operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate.

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 μs instruction cycle time
- 8192 bytes on-board EPROM
- 128 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- 8-Channel A/D converter with prescaler and both differential and single ended modes
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Ten multi-source vectored interrupts servicing
  - External Interrupt
  - Idle Timer T0
  - Two Timers each with 2 interrupts
  - MICROWIRE/PULSE
  - Multi-Input Wake Up
  - Software Trap
  - Defalut VIS
- Two 16-bit timers, each with two 16-bit registers supporting:
  - Processor Independent PWM mode
  - External Event counter mode
  - Input Capture mode
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set wih True bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC with 37 I/O pins
- Software selectable I/O options
  - TRI-STATE® Output
  - Push-Pull Output
  - Weak Pull Up Input
  - High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CF
- Real time emulation and full program debug offered by National's Development Systems

Block Diagram

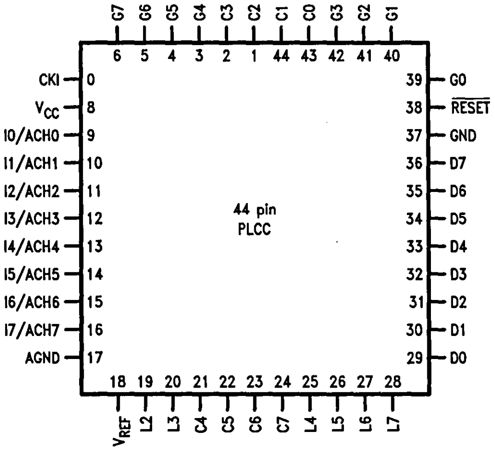


TL/DD/10464-1

FIGURE 1. COP888CFMH Block Diagram

Connection Diagram

Plastic Chip Carrier



TL/DD/10464-2

Top View

FIGURE 2. COP888CFMH Connection Diagram

COP888CFMH Pinouts

Port	Type	Alt. Fun	Alt. Fun	MUX Mode	44-Pin PCC
L0	I/O	MIWU			
L1	I/O	MIWU			
L2	I/O	MIWU			19
L3	I/O	MIWU			20
L4	I/O	MIWU	T2A		25
L5	I/O	MIWU	T2B		26
L6	I/O	MIWU			27
L7	I/O	MIWU			28
G0	I/O	INT		ALE	39
G1	WDOUT				40
G2	I/O	T1B		$\overline{WR}$	41
G3	I/O	T1A		$\overline{RD}$	42
G4	I/O	SO			3
G5	I/O	SK			4
G6	I	SI		ME	5
G7	I/CKO	HALT RESTART			6
D0	O			I/O BIT 0	29
D1	O			I/O BIT 1	30
D2	O			I/O BIT 2	31
D3	O			I/O BIT 3	32
I0	I	ACH0			9
I1	I	ACH1			10
I2	I	ACH2			11
I3	I	ACH3			12
I4	I	ACH4			13
I5	I	ACH5			14
I6	I	ACH7			15
I7	I				16
D4	O			I/O BIT 4	33
D5	O			I/O BIT 5	34
D6	O			I/O BIT 6	35
D7	O			I/O BIT 7	36
C0	I/O				43
C1	I/O				44
C2	I/O				1
C3	I/O				2
C4	I/O				21
C5	I/O				22
C6	I/O				23
C7	I/O				24
V_{REF}	$+V_{REFF}$				18
AGND/GND					17
V_{CC}					8
GND					37
CKI					7
$\overline{RESET}$				V_{PP}	38

I/O BITS = Address and Data Lines

MUX MODE = Programming Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin (Note 1)	$-0.3V$ to $V_{CC} + 0.3V$
ESD Susceptibility (Note 5)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 2)	Peak-to-Peak			$0.1 V_{CC}$	V
Supply Current (Note 3)					
CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			25	mA
HALT Current (Note 4)	$V_{CC} = 5.5V$, CKI = 0 MHz		200		µA
IDLE Current					
CKI = 10 MHz	$V_{CC} = 5.5V$, $t_c = 1\ \mu s$			15	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		$0.8 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
CKI (External and Crystal Osc. Modes)					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
All Other Inputs					
Logic High		$0.7 V_{CC}$			V
Logic Low				$0.2 V_{CC}$	V
Hi-Z Input Leakage	$V_{CC} = 5.5$	−2		+2	µA
Input Pullup Current	$V_{CC} = 5.5V$	40		250	µA
G and L Port Input Hysteresis			$0.05 V_{CC}$		V
Output Current Levels					
D Outputs					
Source	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	0.4			mA
Sink	$V_{CC} = 4.5V$, $V_{OL} = 1V$	10			mA
All Others					
Source (Weak Pull-Up Mode)	$V_{CC} = 4.5V$, $V_{OH} = 2.7V$	10		100	µA
Source (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OH} = 3.3V$	0.4			mA
Sink (Push-Pull Mode)	$V_{CC} = 4.5V$, $V_{OL} = 0.4V$	1.6			mA
TRI-STATE Leakage	$V_{CC} = 4.5V$	−2		+2	µA

Note 1: Except pins G6 (ME) and the RESET (V_{PP}) pin during EPROM MUX mode programming at which time the absolute maximum voltage is 14V on these two pins.

Note 2: Rate of voltage change must be less then 0.5 V/ms.

Note 3: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 4: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. If the A/D is not being used and minimum standby current is desired, V_{REF} should be tied to AGND (effectively shorting the reference resistor). The clock monitor is disabled.

Note 5: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $0°C \leq T_A \leq +70°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All Others				 15 3	 mA mA
Maximum Input Current without Latchup (Note 8)	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

A/D Converter Specifications $V_{CC} = 5V \pm 10\%$ $(V_{SS} - 0.050V) \leq$ Any Input $\leq (V_{CC} + 0.050V)$

Parameter	Conditions	Min	Typ	Max	Units
Resolution				8	Bits
Reference Voltage Input	AGND = 0V	3		V_{CC}	V
Absolute Accuracy	$V_{REF} = V_{CC}$			±1	LSB
Non-Linearity	$V_{REF} = V_{CC}$ Deviation from the best straight line			±½	LSB
Differential Non-Linearity	$V_{REF} = V_{CC}$			±½	LSB
Input Reference Resistance		1.6		4.8	kΩ
Common Mode Input Range (Note 9)		AGND		V_{REF}	V
DC Common Mode Error				±¼	LSB
Off Channel Leakage Current			1		μA
On Channel Leakage Current			1		μA
A/D Clock Frequency (Note 7)		0.1		1.67	MHz
Conversion Time (Note 6)			12		A/D clock Cycles

Note 6: Conversion Time includes sample and hold time.

Note 7: See Prescaler description.

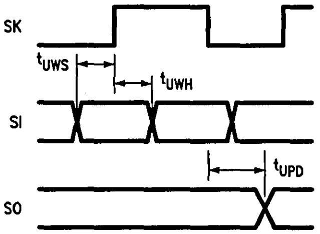
Note 8: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

Note 9: For $V_{IN}(-) \geq V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input. The diodes will forward conduct or analog input voltages below ground or above the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as he analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 10: Parameter sampled but not 100% tested.

AC Electrical Characteristics $0^{\circ}C \leq T_A \leq +70^{\circ}C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c)					
Crystal, Resonator,		1		DC	μs
R/C Oscillator		3		DC	μs
CKI Clock Duty Cycle (Note 10)	f_r = Max	40		60	%
Rise Time (Note 10)	f_r = 10 MHz Ext Clock			5	ns
Fall Time (Note 10)	f_r = 10 MHz Ext Clock			5	ns
Inputs					
t_{SETUP}		200			ns
t_{HOLD}		60			ns
Output Propagation Delay	R_L = 2.2k, C_L = 100 pF				
t_{PD1}, t_{PD0}					
SO, SK				0.7	μs
All Others				1	μs
MICROWIRE™ Setup Time (t_{UWS})		20			ns
MICROWIRE Hold Time (t_{UWH})		56			ns
MICROWIRE Output Propagation Delay (t_{UPD})				220	ns
Input Pulse Width					
Interrupt Input High Time		1			t_c
Interrupt Input Low Time		1			t_c
Timer Input High Time		1			t_c
Timer Input Low Time		1			t_c
Reset Pulse Width		1			μs



TL/DD/10464-3

FIGURE 3. MICROWIRE/PLUS Timing

Pin Descriptions

V_{CC} and GND are the power supply pins.

V_{REF} and AGND are the reference pins for the onboard A/D converter.

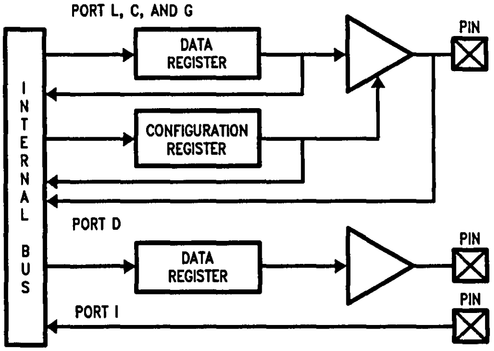
CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset Description section.

The COP888FGMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888FGMH memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888FGMH.

The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/10464-4

FIGURE 4. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

Pin Descriptions (Continued)

The Port L supports Multi-Input Wake Up. L4 and L5 are used for the timer input functions T2A and T2B.

The Port L has the following alternate features:

L2	MIWU
L3	MIWU
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU
L7	MIWU

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

When programming the COP888CFMH, G0 becomes Address Latch Enable (ALE) and pins G2, G3 and G6 become Write ($\overline{WR}$), Read ($\overline{RD}$) and Mux Enable (ME), respectively.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0	INTR (External Interrupt Input)
G2	T1B (Timer T1 Capture Input)
G3	T1A (Timer T1 I/O)
G4	SO (MICROWIRE Serial Data Output)
G5	SK (MICROWIRE Serial Clock)
G6	SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1	WDOUT WatchDog and/or Clock Monitor dedicated output
G7	CKO Oscillator dedicated output or general purpose input

When programming the COP888FGMH, G0 becomes Address Latch Enable (ALE) and pins G2, G3 and G6 become Write ($\overline{WR}$), Read ($\overline{RD}$) and Mux Enable (ME), respectively.

Port I is an eight-bit Hi-Z input port and also provides the analog inputs to the A/D Converter.

Port D is an 8-bit output port that is preset high when $\overline{RESET}$ goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Port C is an 8-bit I/O port.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_C$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.

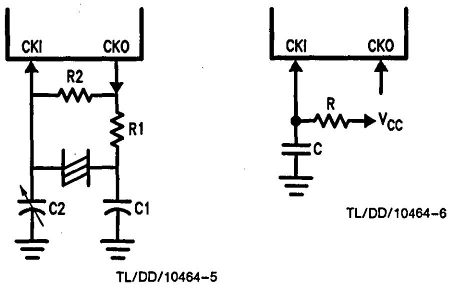


FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. RC Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP888FGMH

The COP888FGMH is a hybrid part consisting of a COP888FG die and an 8k EPROM die with port re-creation logic. In order to access the EPROM, the COP888FG die has been placed in ROMless mode by holding its D2 pad at GND. The other D-lines of the COP888FG die are used to communicate with the EPROM. All 8 D-lines are recreated internally and appear on the pins of the COP888FGMH as normal D outputs.

When programming the COP888FGMH, a multiplexed method (MUX MODE) is used. To enter this mode a voltage of 12.2V–13V is applied to pin ME (pin G6). The 8 D-Port pins on the COP888FGMH become address bits with a low to high transition on ALE (pin G0), and data bits with a high to low transition on $\overline{WR}$ (pin G2). With a low to high transition on $\overline{RD}$ (pin G3), the data being programmed is verified. The following steps must be followed in order to place the part in programming mode:

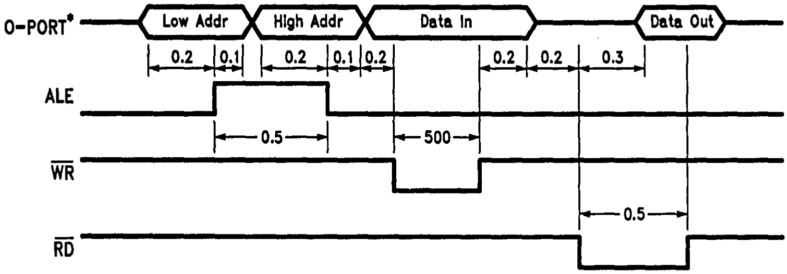
1. Apply V_{CC} = 5V.
2. Ground the RESET and CKI pins. This puts the COP outputs in TRI-STATE mode.
3. Apply V_{PP} = 12.2V–13V to pin G6. This places the EPROM in MUX mode. The current requirement is 450 µA maximum (V_{IN} = 13V, V_{CC} = 6.5V, −40°C).
4. Apply 12.2V–13V to the RESET pin. This supplies V_{PP} to the EPROM which requires 30 mA maximum during WRITE pulses. It is permissible for V_{PP} to drop to V_{CC} during the READ pulses as is done on some programmers.
5. Begin programming each EPROM byte interactively. (See *Figures 6* and *7*). The interactive programming algorithm programs a byte with a 0.5 mS pulse, and then does a verify to determine if that byte was fully programmed. If it was not, the program pulse is repeated and verified again. This is done up to a maximum of 20 times, but most bytes will program with a single pulse.

Erasure of program memory is achieved by removing the part from its socket and exposing the transparent window to an ultra-violet light source.

Notes: The last byte of program memory (EPROM location 01FFF Hex) must contain one of two values: 07F Hex for the HALT enabled mode, or 0FF Hex for the HALT disabled mode. The COP888CFMH will not function properly if any other value resides in this last byte location.

V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}. The EPROM must not be inserted into or removed from a board with voltage applied to V_{PP} or V_{CC}.

The maximum absolute allowable voltage which may be applied to the G6 (ME) and RESET (V_{PP}) pins during programming is 14V. Care must be taken when switching the V_{PP} supply to prevent any overshoot from exceeding this 14V limit. At least a 0.1 µF capacitor is required across V_{PP} to GND and V_{CC} to GND to suppress spurious voltage transients which may damage the device.

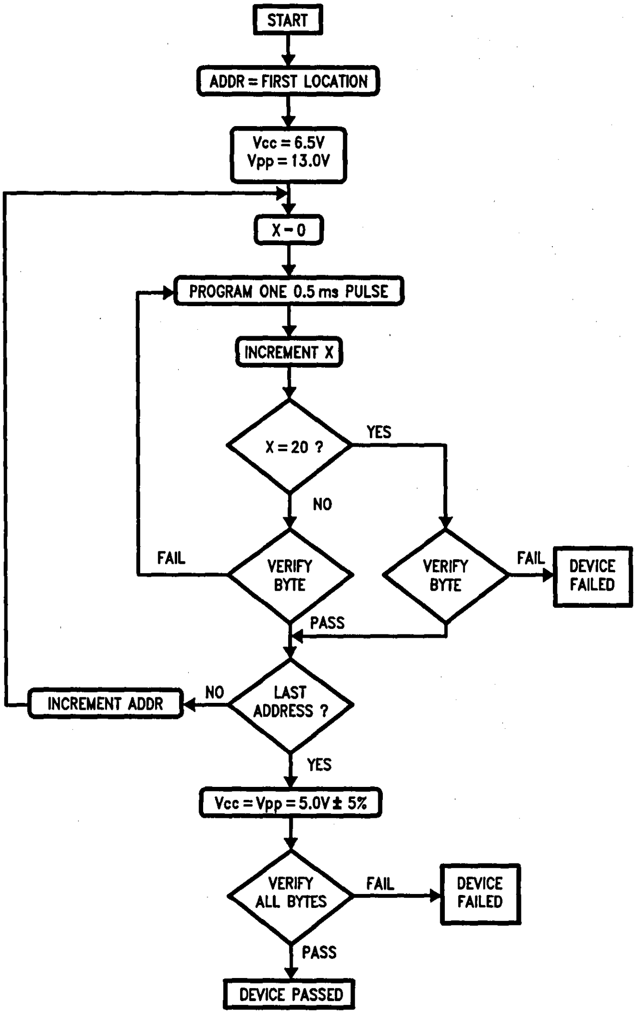


TL/DD/10464-7

Note: All minimum times are in µs.

* O-Port = D0 to D7

FIGURE 6. COP888FGMH MUX Mode Programming Timing Diagram



TL/DD/10464-08

FIGURE 7. COP888FGMH MUX Mode Programming Flow Chart

Development Support

MOLE DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

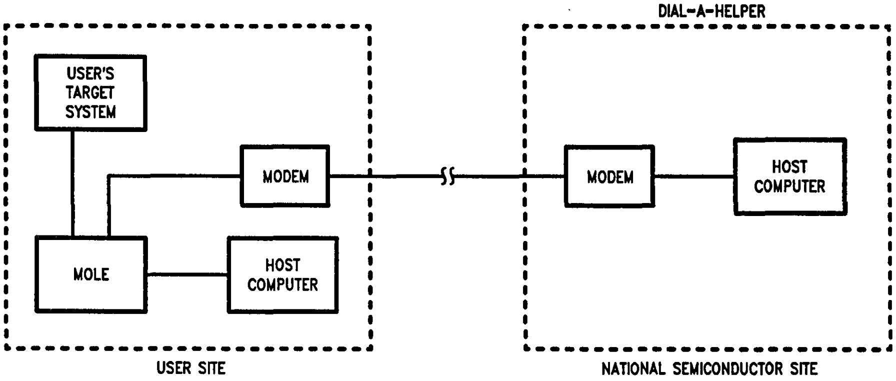
ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 Baud	
	Set-up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hrs., 7 Days	



TL/DD/10464-09

National Semiconductor

PRELIMINARY

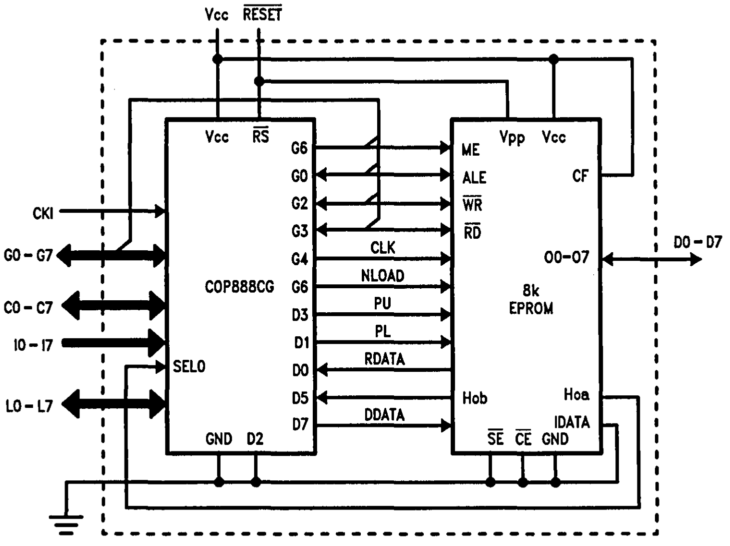
COP888CGMH Single-Chip microCMOS Microcontroller

General Description

The COP888CGMH hybrid emulator is a member of the COPS™ microcontroller family. It is functionally identical to the COP888CG except that its package contains an 8k EPROM in place of masked program ROM. This 44-pin part contains a transparent window which allows the EPROM to be erased and re-programmed. (Continued)

Features

- Low cost 8-bit microcontroller
- Fully static CMOS, with low current drain
- Two power saving modes: HALT and IDLE
- 1 µs instruction cycle time
- 8192 bytes on-board EPROM
- 192 bytes on-board RAM
- Single supply operation: 4.5V–5.5V
- Full duplex UART
- Two analog comparators
- MICROWIRE/PLUS™ serial I/O
- WatchDog and Clock Monitor logic
- Idle Timer
- Multi-Input Wakeup (MIWU) with optional interrupts (8)
- Three 16-bit timers, each with two 16-bit registers supporting:
  - — Processor Independent PWM mode
  - — External Event counter mode
  - — Input Capture mode
- Fourteen multi-source vectored interrupts servicing
  - — External Interrupt
  - — Idle Timer T0
  - — Two Timers each with 2 interrupts
  - — MICROWIRE/PLUS
  - — Multi-Input Wake Up
  - — Software Trap
  - — UART (2)
  - — Default VIS
- 8-bit Stack Pointer SP (stack in RAM)
- Two 8-bit Register Indirect Data Memory Pointers (B and X)
- Versatile instruction set with true bit manipulation
- Memory mapped I/O
- BCD arithmetic instructions
- Package: 44 PCC with 39 I/O pins
- Software selectable I/O options
  - — TRI-STATE® Output
  - — Push-Pull Output
  - — Weak Pull Up Input
  - — High Impedance Input
- Schmitt trigger inputs on ports G and L
- Form fit and function emulation device for the COP888CG
- Real time emulation and full program debug offered by National's Development Systems



TL/DD/10425-1

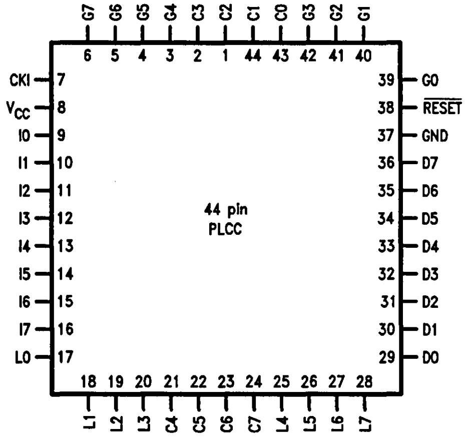
FIGURE 1. COP888CGMH Block Diagram

General Description (Continued)

The COP888CGMH is a fully static part, fabricated using double-metal silicon gate microCMOS technology. Features include an 8-bit memory mapped architecture, MICROWIRE/PLUS serial I/O, three 16-bit timer/counters supporting three modes (Processor Independent PWM generation, External Event counter, and Input Capture mode capabilities), full duplex UART, two comparators, and two power savings modes (HALT and IDLE), both with a multi-sourced wakeup/interrupt capability. This multi-sourced interrupt capability may also be used independent of the HALT or IDLE modes. Each I/O pin has software selectable configurations. The COP888CGMH operates over a voltage range of 4.5V to 5.5V. High throughput is achieved with an efficient, regular instruction set operating at a maximum of 1 μs per instruction rate.

Connection Diagram

Plastic Chip Carrier



TL/DD/10425-2

Top View

FIGURE 2. COP888CGMH Connection Diagram

COP888CGMH Pinouts for 28-, 40- and 44-Pin Packages

Port	Type	Alt. Fun	Alt. Fun	MUX Mode	44-Pin PCC
L0	I/O	MIWU			17
L1	I/O	MIWU	CKX		18
L2	I/O	MIWU	TDX		19
L3	I/O	MIWU	RDX		20
L4	I/O	MIWU	T2A		25
L5	I/O	MIWU	T2B		26
L6	I/O	MIWU	T3A		27
L7	I/O	MIWU	T3B		28
G0	I/O	INT		ALE	39
G1	WDOUT				40
G2	I/O	T1B		$\overline{WR}$	41
G3	I/O	T1A		$\overline{RD}$	42
G4	I/O	SO			3
G5	I/O	SK			4
G6	I	SI		ME	5
G7	I/CKO	HALT RESTART			6
D0	O			I/O BIT 0	29
D1	O			I/O BIT 1	30
D2	O			I/O BIT 2	31
D3	O			I/O BIT 3	32
I0	I				9
I1	I	COMP1IN−			10
I2	I	COMP1IN+			11
I3	I	COMP1OUT			12
I4	I	COMP2IN−			13
I5	I	COMP2IN+			14
I6	I	COMP2OUT			15
I7	I				16
D4	O			I/O BIT 4	33
D5	O			I/O BIT 5	34
D6	O			I/O BIT 6	35
D7	O			I/O BIT 7	36
C0	I/O				43
C1	I/O				44
C2	I/O				1
C3	I/O				2
C4	I/O				21
C5	I/O				22
C6	I/O				23
C7	I/O				24
V_{CC}					8
GND					37
CKI					7
$\overline{RESET}$				V_{PP}	38

I/O BITS = Address and Data Lines

MUX MODE = Programming Mode

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	6V
Voltage at Any Pin (Note 1)	−0.3V to V_{CC} + 0.3V
ESD Susceptibility (Note 5)	2000V
Total Current into V_{CC} Pin (Source)	100 mA
Total Current out of GND Pin (Sink)	110 mA
Storage Temperature Range	−65°C to +140°C

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

Parameter	Conditions	Min	Typ	Max	Units
Operating Voltage		4.5		5.5	V
Power Supply Ripple (Note 2)	Peak-to-Peak			0.1 V_{CC}	V
Supply Current (Note 3)					
CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			25	mA
HALT Current (Note 4)	V_{CC} = 5.5V, CKI = 0 MHz		200		µA
IDLE Current					
CKI = 10 MHz	V_{CC} = 5.5V, t_c = 1 µs			15	mA
Input Levels					
$\overline{\text{RESET}}$					
Logic High		0.8 V_{CC}			V
Logic Low				0.2 V_{CC}	V
CKI (External and Crystal Osc. Modes)					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
All Other Inputs					
Logic High		0.7 V_{CC}			V
Logic Low				0.2 V_{CC}	V
Hi-Z Input Leakage	V_{CC} = 5.5	−2		+2	µA
Input Pullup Current	V_{CC} = 5.5V	40		250	µA
G and L Port Input Hysteresis			0.05 V_{CC}		V
Output Current Levels					
D Outputs					
Source	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink	V_{CC} = 4.5V, V_{OL} = 1V	10			mA
All Others					
Source (Weak Pull-Up Mode)	V_{CC} = 4.5V, V_{OH} = 2.7V	10		100	µA
Source (Push-Pull Mode)	V_{CC} = 4.5V, V_{OH} = 3.3V	0.4			mA
Sink (Push-Pull Mode)	V_{CC} = 4.5V, V_{OL} = 0.4V	1.6			mA
TRI-STATE Leakage	V_{CC} = 4.5V	−2		+2	µA

Note 1: Except pins G6 (ME) and the RESET (V_{PP}) pin during EPROM MUX mode programming at which time the absolute maximum voltage is 14V on these two pins. (Refer to section 6.0, Programming the COP888CGMH).

Note 2: Rate of voltage change must be less then 0.5 V/ms.

Note 3: Supply current is measured after running 2000 cycles with a square wave CKI input, CKO open, inputs at rails and outputs open.

Note 4: The HALT mode will stop CKI from oscillating in the RC and the Crystal configurations. Test conditions: All inputs tied to V_{CC}, L and G ports in the TRI-STATE mode and tied to ground, all outputs low and tied to ground. The comparators and Clock Monitor are disabled.

Note 5: Human body model, 100 pF through 1500Ω.

DC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified (Continued)

Parameter	Conditions	Min	Typ	Max	Units
Allowable Sink/Source Current per Pin D Outputs (Sink) All Others				 15 3	 mA mA
Maximum Input Current without Latchup (Note 6)	$T_A = 25°C$			±100	mA
RAM Retention Voltage, V_r	500 ns Rise and Fall Time (Min)	2			V
Input Capacitance				7	pF
Load Capacitance on D2				1000	pF

AC Electrical Characteristics $0°C \le T_A \le +70°C$ unless otherwise specified

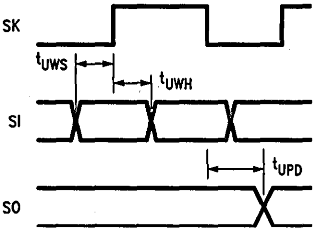
Parameter	Conditions	Min	Typ	Max	Units
Instruction Cycle Time (t_c) Crystal, Resonator, R/C Oscillator		 1 3		 DC DC	 μs μs
CKI Clock Duty Cycle (Note 7) Rise Time (Note 7) Fall Time (Note 7)	f_r = Max f_r = 10 MHz Ext Clock f_r = 10 MHz Ext Clock	40		60 5 5	% ns ns
Inputs t_{SETUP} t_{HOLD}		 200 60			 ns ns
Output Propagation Delay t_{PD1}, t_{PD0} SO, SK All Others	R_L = 2.2k, C_L = 100 pF			 0.7 1	 μs μs
MICROWIRE™ Setup Time (t_{UWS}) MICROWIRE Hold Time (t_{UWH}) MICROWIRE Output Propagation Delay (t_{UPD})		20 56		 220	ns ns ns
Input Pulse Width Interrupt Input High Time Interrupt Input Low Time Timer Input High Time Timer Input Low Time		 1 1 1 1			 t_c t_c t_c t_c
Reset Pulse Width		1			μs

Note 6: Except pin G7: −60 mA to +100 mA (sampled but not 100% tested).

Note 7: Parameter sampled but not 100% tested.

Comparators AC and DC Characteristics $V_{CC} = 5V, T_A = 25°C$

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$0.4V \leq V_{IN} \leq V_{CC} - 1.5V$		±10	±25	mV
Input Common Mode Voltage Range		0.4		$V_{CC} - 1.5$	V
Low Level Output Current	$V_{OL} = 0.4V$	1.6			mA
High Level Output Current	$V_{OH} = 4.6V$	1.6			mA
DC Supply Current per Comparator (When Enabled)				250	μA
Response Time	TBD mV Step, TBD mV Overdrive, 100 pF Load		1		μs



TL/DD/10425-5

FIGURE 3. MICROWIRE/PLUS Timing

Pin Descriptions

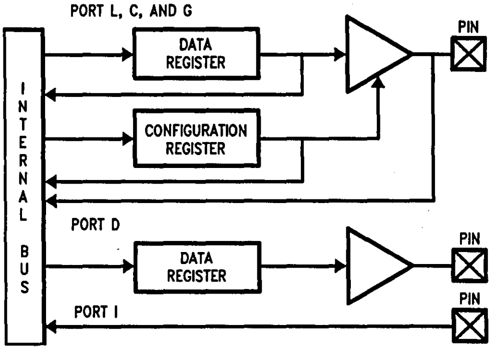
V_{CC} and GND are the power supply pins.

CKI is the clock input. This can come from an R/C generated oscillator, or a crystal oscillator (in conjunction with CKO). See Oscillator Description section.

$\overline{\text{RESET}}$ is the master reset input. See Reset Description section.

The COP888CGMH contains three bidirectional 8-bit I/O ports (C, G and L), where each individual bit may be independently configured as an input, output or TRI-STATE under program control. Three data memory address locations are allocated for each of these I/O ports. Each I/O port has two associated 8-bit memory mapped registers, the CONFIGURATION register and the output DATA register. A memory mapped address is also reserved for the input pins of each I/O port. (See the COP888CGMH memory map for the various addresses associated with the I/O ports.) *Figure 3* shows the I/O port configurations for the COP888CGMH. The DATA and CONFIGURATION registers allow for each port bit to be individually configured under software control as shown below:

CONFIGURATION Register	DATA Register	Port Set-Up
0	0	Hi-Z Input (TRI-STATE Output)
0	1	Input with Weak Pull-Up
1	0	Push-Pull Zero Output
1	1	Push-Pull One Output



TL/DD/10425-6

FIGURE 4. I/O Port Configurations

PORT L is an 8-bit I/O port. All L-pins have Schmitt triggers on the inputs.

The Port L supports Multi-Input Wake Up on all eight pins. L1 is used for the UART external clock. L2 and L3 are used for the UART transmit and receive. L4 and L5 are used for the timer input functions T2A and T2B. L6 and L7 are used for the timer input functions T3A and T3B.

The Port L has the following alternate features:

L0	MIWU
L1	MIWU or CKX
L2	MIWU or TDX
L3	MIWU or RDX
L4	MIWU or T2A
L5	MIWU or T2B
L6	MIWU or T3A
L7	MIWU or T3B

Port G is an 8-bit port with 5 I/O pins (G0, G2–G5), an input pin (G6), and two dedicated output pins (G1 and G7). Pins G0 and G2–G6 all have Schmitt Triggers on their inputs. Pin G1 serves as the dedicated WDOUT WatchDog output, while pin G7 is either input or output depending on the oscillator mask option selected. With the crystal oscillator option selected, G7 serves as the dedicated output pin for the CKO

Pin Descriptions (Continued)

clock output. With the single-pin R/C oscillator mask option selected, G7 serves as a general purpose input pin but is also used to bring the device out of HALT mode with a low to high transition. There are two registers associated with the G Port, a data register and a configuration register. Therefore, each of the 5 I/O bits (G0, G2–G5) can be individually configured under software control.

When programming the COP888CGMH, G0 becomes Address Latch Enable (ALE) and pins G2, G3 and G6 become Write bar ($\overline{\text{WR}}$), Read bar ($\overline{\text{RD}}$) and Mux Enable (ME), respectively.

Since G6 is an input only pin and G7 is the dedicated CKO clock output pin (crystal clock option) or general purpose input (R/C clock option), the associated bits in the data and configuration registers for G6 and G7 are used for special purpose functions as outlined below. Reading the G6 and G7 data bits will return zeros.

Note that the chip will be placed in the HALT mode by writing a "1" to bit 7 of the Port G Data Register. Similarly the chip will be placed in the IDLE mode by writing a "1" to bit 6 of the Port G Data Register.

Writing a "1" to bit 6 of the Port G Configuration Register enables the MICROWIRE/PLUS to operate with the alternate phase of the SK clock. The G7 configuration bit, if set high, enables the clock start up delay after HALT when the R/C clock configuration is used.

	Config Reg.	Data Reg.
G7	CLKDLY	HALT
G6	Alternate SK	IDLE

Port G has the following alternate features:

G0 INTR (External Interrupt Input)

G2 T1B (Timer T1 Capture Input)

G3 T1A (Timer T1 I/O)

G4 SO (MICROWIRE Serial Data Output)

G5 SK (MICROWIRE Serial Clock)

G6 SI (MICROWIRE Serial Data Input)

Port G has the following dedicated functions:

G1 WDOUT WatchDog and/or Clock Monitor dedicated output

G7 CKO Oscillator dedicated output or general purpose input

When programming the COP888CGMH, G0 becomes Address Latch Enable (ALE) and pins G2, G3 and G6 become Write ($\overline{\text{WR}}$), Read ($\overline{\text{RD}}$) and Mux Enable (ME), respectively.

Port I1–I3 are used for Comparator 1. Port I4–I6 are used for Comparator 2.

The Port I has the following alternate features.

I1 COMP1−IN (Comparator 1 Negative Input)

I2 COMP1+IN (Comparator 1 Positive Input)

I3 COMP1OUT (Comparator 1 Output)

I4 COMP2−IN (Comparator 2 Negative Input)

I5 COMP2+IN (Comparator 2 Positive Input)

I6 COMP2OUT (Comparator 2 Output)

Port D is an 8-bit output port that is preset high when $\overline{\text{RESET}}$ goes low. The user can tie two or more D port outputs together in order to get a higher drive.

Port C is an 8-bit I/O port.

Oscillator Circuits

The chip can be driven by a clock input on the CKI input pin which can be between DC and 10 MHz. The CKO output clock is on pin G7 (crystal configuration). The CKI input frequency is divided down by 10 to produce the instruction cycle clock ($1/t_c$).

Figure 5 shows the Crystal and R/C diagrams.

CRYSTAL OSCILLATOR

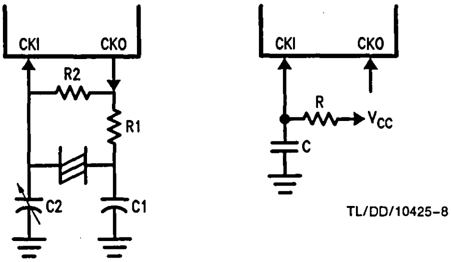
CKI and CKO can be connected to make a closed loop crystal (or resonator) controlled oscillator.

Table I shows the component values required for various standard crystal values.

R/C OSCILLATOR

By selecting CKI as a single pin oscillator input, a single pin R/C oscillator circuit can be connected to it. CKO is available as a general purpose input, and/or HALT restart input.

Table II shows the variation in the oscillator frequencies as functions of the component (R and C) values.



TL/DD/10425-8

TL/DD/10425-7

FIGURE 5. Crystal and R/C Oscillator Diagrams

TABLE I. Crystal Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R1 (kΩ)	R2 (MΩ)	C1 (pF)	C2 (pF)	CKI Freq (MHz)
0	1	30	30–36	10
0	1	30	30–36	4
0	1	200	100–150	0.455

TABLE II. RC Oscillator Configuration, $T_A = 25°C, V_{CC} = 5V$

R (kΩ)	C (pF)	CKI Freq (MHz)	Instr. Cycle (μs)
3.3	82	2.8 to 2.2	3.6 to 4.5
5.6	100	1.5 to 1.1	6.7 to 9
6.8	100	1.1 to 0.8	9 to 12.5

Programming the COP888CGMH

The COP888CGMH is a hybrid part consisting of a COP888CG die and an 8k EPROM die with port re-creation logic. In order to access the EPROM, the COP888CG die has been placed in ROMless mode by holding its D2 pad at GND. The other D-lines of the COP888CG die are used to communicate with the EPROM. All 8 D-lines are recreated internally and appear on the pins of the COP888CGMH as normal D outputs.

When programming the COP888CGMH, a multiplexed method (MUX MODE) is used. To enter this mode a voltage of 12.2V–13V is applied to ME (pin G6). The 8 D-Port pins on the COP888CGMH become address bits with a low to high transition on ALE (pin G0), and data bits with a high to low transition on $\overline{\text{WR}}$ (pin G2). With a low to high transition on $\overline{\text{RD}}$ (pin G3), the data being programmed is verified. The following steps must be followed in order to place the part in programming mode:

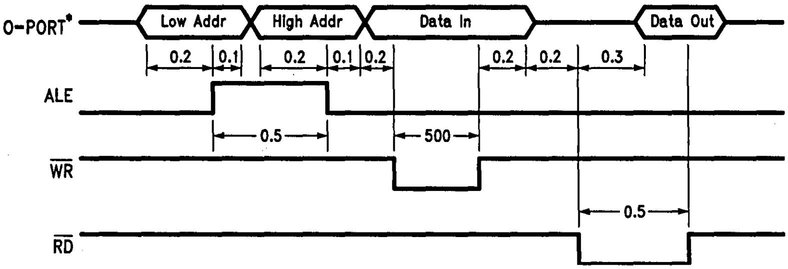
1. Apply V_{CC} = 5V.
2. Ground the RESET and CKI pins. This puts the COP outputs in TRI-STATE mode.
3. Apply 12.2V–13V to pin G6. This places the EPROM in MUX mode. The current requirement is 450 µA maximum (V_{IN} = 13V, V_{CC} = 5.0V, −40°C).
4. Apply 12.2V–13V to the RESET pin. This supplies V_{PP} to the EPROM which requires 30 mA maximum during WRITE pulses. It is permissible for V_{PP} to drop to V_{CC} during the READ pulses as is done on some programmers.
5. Begin programming each EPROM byte interactively. (See *Figures 6* and *7*). The interactive programming algorithm programs a byte with a 0.5 mS pulse, and then does a verify to determine if that byte was fully programmed. If it was not, the program pulse is repeated and verified again. This is done up to a maximum of 20 times, but most bytes will program with a single pulse.

Erasure of program memory is achieved by removing the part from its socket and exposing the transparent window to an ultra-violet light source.

NOTE: The last byte of program memory (EPROM location 01FFF HEX) must contain one of two values: 07F HEX for the HALT enabled mode, or 0FF HEX for the HALT disabled mode. The COP888CGMH will not function properly if any other value resides in this last byte location.

V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}. The EPROM must not be inserted into or removed from a board with voltage applied to V_{PP} or V_{CC}.

The maximum absolute allowable voltage which may be applied to the G6 (ME) and RESET (V_{PP}) pins during programming is 14V. Care must be taken when switching the V_{PP} supply to prevent any overshoot from exceeding this 14V limit. At least a 0.1 µF capacitor is required across V_{PP} to GND and V_{CC} to GND to suppress spurious voltage transients which may damage the device.

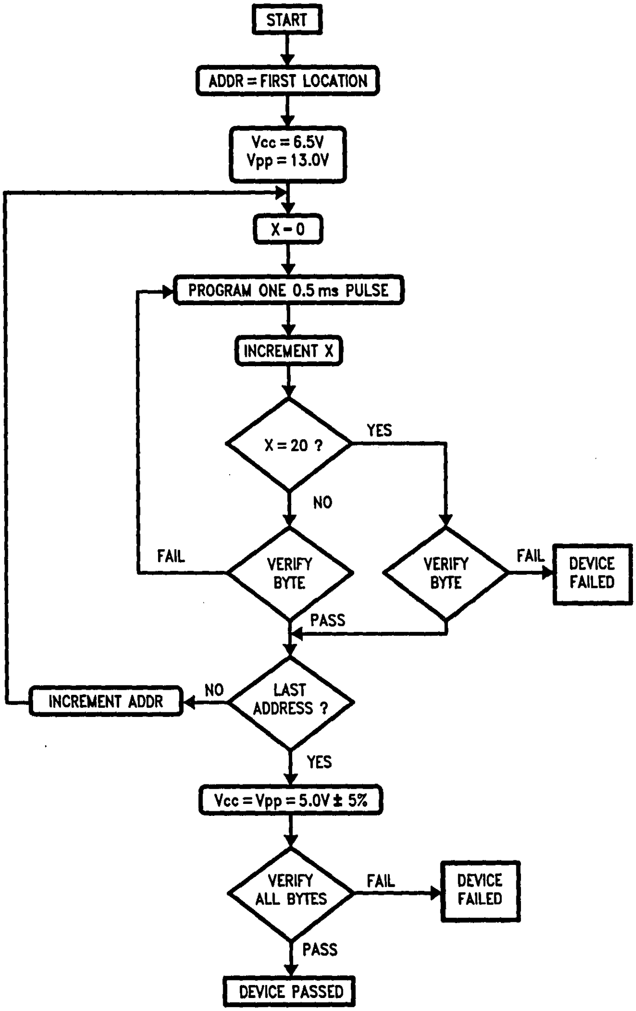


TL/DD/10425-9

Note: All minimum times are in µs.

* O-Port = D0 to D7

FIGURE 6. COP888CGMH MUX Mode Programming Timing Diagram



TL/DD/10425-10

FIGURE 7. COP888CGMH MUX Mode Programming Flow Chart

Development Support

MOLE™ DEVELOPMENT SYSTEM

The MOLE (Microcomputer On Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs microcontrollers and the HPC family of products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other MOLEs in a multi-MOLE environment.

MOLE can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Programmer's Manual		420411060-01

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. The Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities could be found. The minimum requirement for accessing the Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

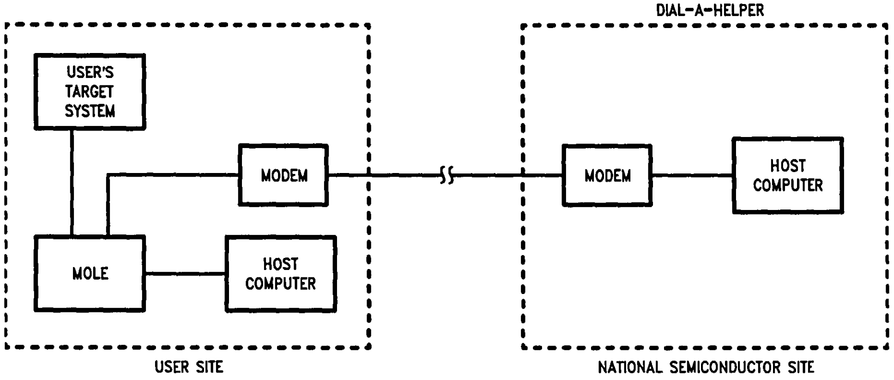
ORDER P/N: MOLE-DIAL-A-HLP

Information System Package contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factor applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 Baud	
	Set-up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hrs., 7 Days	



TL/DD/10425-11

Section 3
COPS Applications

Section 3 Contents

Easy Logarithms for COP400

National Semiconductor
COP Brief 2

Logarithms have long been a convenient tool for the simplification of multiplication, division, and root extraction. Many assembly language programmers avoid the use of logarithms because of supposed complexity in their application to binary computers. Logarithms conjure up visions of time consuming iterations during the solution of a long series. The problem is far simpler than imagined and its solution yields, for the applications programmer, the classical benefits of logarithms:

1) Multiplication can be performed by a single addition.

2) Division can be performed by a single subtraction.

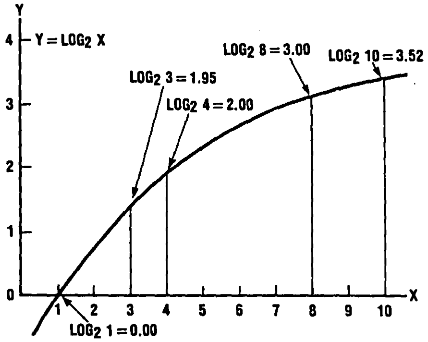
3) Raising a number to a power involves a single multiply.

4) Extracting a root involves a single divide.

When applied to binary computer operation logarithms yield two further important advantages. First, a broad range of values can be handled without resorting to floating point techniques (other than implied by the characteristic). Second, it is possible to establish the significance of an answer during the body of a calculation, again, without resorting to floating point techniques.

Implementation of base$_{10}$ logarithms in a binary system is cumbersome and unnecessary since logarithmic functions can be implemented in a number system of any base. The techniques presented here deal only with logarithms to the base$_2$.

A logarithm consists of two parts: an integer characteristic and a fractional mantissa.



TL/DD/6942-1

	CHARACTERISTIC	MANTISSA
LOG$_2$ 3 =	1	0.95
LOG$_2$ 4 =	2	0.00
LOG$_2$ 8 =	3	0.00
LOG$_2$ 10 =	3	0.52

FIGURE 1. The Logarithmic Function and Some Example Values

In *Figure 1* some points on the logarithmic curve are identified and evaluated to the base$_2$. Notice that the characteristic in each case represents the highest even power of 2 contained in the value of X. This is readily seen when binary notation is used.

X_{10}	X_2					Log$_2$ X	Log$_2$ X Where X =
	2^4	2^3	2^2	2^1	2^0	Characteristic	Even Power of 2
3	0	0	0	↑	1	1	
4	0	0	↑	0	0	2	010.0000
8	0	↑	0	0	0	3	011.0000
10	0	↑	0	1	0	3	

FIGURE 2. Identification of the Characteristic

In *Figure 2* each point evaluated in *Figure 1* has been repeated using binary notation. An arrow subscript indicates the highest even power of 2 appearing in each value of X. Notice that in X = 3 the highest even power of 2 is 2^1. Thus the characteristic of the log$_2$ 3 is 1. Where X = 10 the characteristic of the log$_2$ 10 is 3.

To find the log$_2$ X is very easy where X is an even power of 2. We simply shift the value of X left until a carry bit emerges from the high order position of the register. This procedure is illustrated in *Figure 3.* This characteristic is found by counting the number of shifts required and subtracting the result from the number of bits in the register. In practice it is easier to being with the number of bits and count down once prior to each shift.

Counter for Characteristic	Value of X in Binary		
1000	0000	1000	Initial
0111	0001	0000	First Shift
0110	0010	0000	Second Shift
0101	0100	0000	Third Shift
0100	1000	0000	Fourth Shift
0011	0000	0000	Fifth Shift
Characteristic	**Mantissa**		**Final**
011.0000	0000		Log$_2$ X = 3.00

FIGURE 3. Conversion to Base$_2$ Logarithm by Base Shift

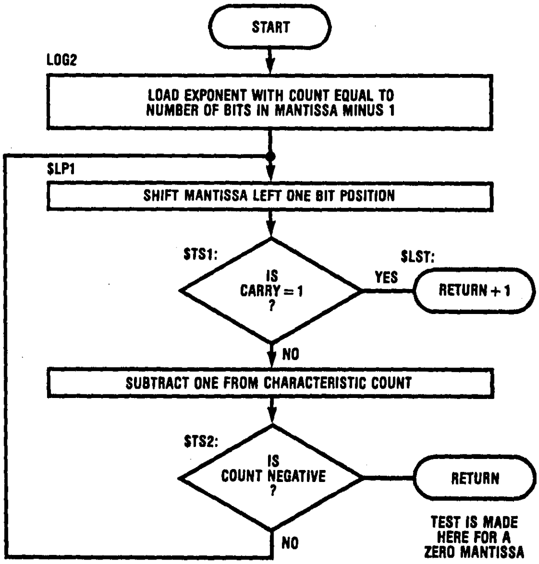
Examination of the final value obtained in *Figure 3* reveals no bits in the mantissa. The value 3 in the characteristic, however, indicates that a bit did exist in the 2^3 position of the original number and would have to be restored in order to reconstruct the original value (antilog).

The log of any even power of 2 can be found in this way:

Decimal	Binary	Log_2
128	10000000	0111.00000000
64	01000000	0110.00000000
32	00100000	0101.00000000
4	00000100	0010.00000000
2	00000010	0001.00000000
1	00000001	0000.00000000

FIGURE 4. Base_2 Logarithms of Even Powers of 2

A simple flow chart, and program, can be devised for generating the values found in the table and, as will be apparent, a straight line approximation for values that are not even powers of 2. The method, as already illustrated in *Figure 3,* involves only shifting a binary number left until the most significant bit moves into the carry position. The characteristic is formed by counting. Since a carry on each successive shift will yield a decreasing power of 2, we must start the characteristic count with the number of bits in the binary value (x) and count down one each shift.


START
LOG2
LOAD EXPONENT WITH COUNT EQUAL TO NUMBER OF BITS IN MANTISSA MINUS 1
$LP1
SHIFT MANTISSA LEFT ONE BIT POSITION
$TS1:
IS CARRY = 1 ?
YES
$LST:
RETURN + 1
NO
SUBTRACT ONE FROM CHARACTERISTIC COUNT
$TS2:
IS COUNT NEGATIVE ?
RETURN
TEST IS MADE HERE FOR A ZERO MANTISSA
NO

TL/DD/6942-2

FIGURE 5. Log Flowchart

```
COP CROSS ASSEMBLER     PAGE:  1
LOGS

                       ; TITLE LOGS                  ; BINARY LOGARITHMS

          01A4                     . CHIP 420

                                   ; ------> CONVERT TO LOGARITHM <------;
                       ;
                       ;                             RAM ASSIGNMENT
                       ;
                       ; DIGIT:    15  14  13  12  11  10  09  08  07  06  05  04  03  02  01  00
                       ;          +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                       ; REG 0    |   |   |   |   |   |   |   |   |CH |HM |LM |   |   |   | TEMP  |
                       ;          +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                       ; REG 1    |   |   |   |CH |HM |LM |   |   |   |   |   |   |   |   | TEMP  |
                       ;          +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                       ; REG 2    |   |   |   |   | TEMP  |   |   |   |   |   |   |   |CH |HM |LM |
                       ;          +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                       ; REG 3    |   |   |   |   |   |   |   |   |   |   | TEMP  |   |CH |HM |LM |
                       ;          +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                       ;
                                   . LOCAL
                       ;
                       ; CH, HM, LM REPRESENT ANY THREE SEQUENTIAL MEMORY DIGITS. THEY
                       ; MAY BE DEFINED IN ANY REGISTER. THE SYMBOLIC NOTATION CH, HM,
                       ; AND LM ARE USED FOR ADDRESSING TO ALLOW USER FLEXIBILITY.
                       ; UPON ENTRY TO THE ROUTINE HM AND LM CONTAIN THE HI AND LO
                       ; OF SOME VALUE X. THE MEMORY POINTER MUST CONTAIN THE ADDRESS
                       ; OF THE CHARACTERISTIC (CH). THE CONTENTS OF THIS LOCATION ARE
                       ; IGNORED AND ARE LOST DURING EXECUTION.
                       ;
                       ; UPON EXIT CH, HM, LM CONTAIN A STRAIGHT LINE APPROXIMATION OF
                       ; THE LOG BASE 2 OF X. CH = CHARACTERISTIC  HM = HI ORDER MANTISSA
                       ; LM = LO ORDER MANTISSA. AN 8 BIT MEMORY AREA (TEMP) IS USED IN
                       ; THE REGISTER OPPOSITE DURING THE CORRECTION OF A STRAIGHT
                       LINE APPROXIMATION OF A LOG OR AN ANTILOG.
                       ;           A TEST IS MADE FOR X = 0. IF THE VALUE OF X
                       ; IS NOT ZERO AN INSTRUCTION IS SKIPPED UPON RETURN
                       ; TO THE CALLING ROUTINE.
                       ;
                       ;                                            — EXAMPLE —
                       ;
                       ; SUBROUTINE CALL                                JSR  LOG2
                       ; RETURN HERE IF X = 0 --->                      JP   ZERO
                       ; RETURN HERE IF X>0 --->                         CONTINUE
                       ;
                       ;
                       ;
                       ;
                       ;
                       ;
     000  00           LOG2:       CLRA                             ; SET CHARACTERISTIC.
     001  57                       AISC       07                    ; TO REG LENGTH -1.
     002  06                       X                                ; STORE IN MEMORY.

COP CROSS ASSEMBLER     PAGE:  2
LOGS

     003  A4           $LP1:       JSRP       SDB2                  ; SET ADDRESS POINTER
                                                                    ; BACK 2 DIGITS.
     004  A9                       JSRP       SHLR                  ; RESET CARRY AND SHIFT
                                                                    ; REG LEFT ONE BIT.
     005  20           $TS1:       SKC                              ; IS CARRY = 1 YET?
     006  C8                       JP         $NO                   ; NO — KEEP GOING.
     007  49           $LST:       RETSK                            ; YES — FINISHED!!
     008  05           $NO:        LD                               ; NO — LOAD COUNT IN ACC.
     009  5F                       AISC       -1                    ; SUBTRACT ONE.
     00A  48           $TS2:       RET                              ; MANTISSA IS A 0! RETURN
     00B  06                       X                                ; STORE CHARACTERISTIC.
     00C  C3                       JP         $LP1                  ; DO IT AGAIN!





                       ; 2 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
                       ; PROGRAM: SDB2, SHLR.
```

TL/DD/6942-5

FIGURE 6

The program shown develops the $\log_2$ of any even power of 2 by shifting and testing as previously described. Examine what happens to a value of X that is not an even power of 2. In *Figure 7,* the number 25 is converted to a base 2 log.

$$25_{10} = 0\,0\,0\,1\,1\,0\,0\,2_2$$

Shift left until carry = 1

```
Characteristic Carry   Mantissa          Log2
   0100          1   100100000100.10010000
```

Figure 7. Straight Line Approximation of Base_2 Log

The resulting number when viewed as an integer characteristic and a fractional mantissa is 4.5625_{10}. The fraction 0.5625 is a straight line approximation of the logarithmic curve between the correct values for the base_2 logs of 2^4 and 2^5. The accuracy of this approximation is sufficient for many applications. The error can be corrected, as will be seen later in this discussion, but for now let's look at the problem of exponents or the conversion to an antilog.

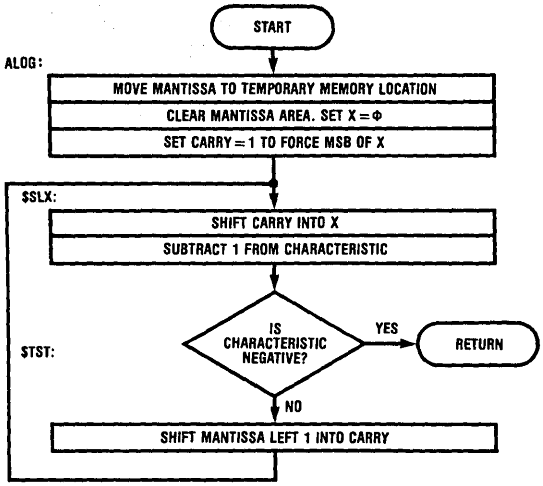
To reconstruct the original value of X, find the antilog, requires only restoration of the most significant bit and then its alignment with the power of 2 position indicated by the characteristic. In the example, approximation ($\log_2 25$ = 0100.1001) restoration of MSB can be accomplished by shifting the mantissa (only) one position to the right. In the process a one is shifted into the MSB position.

Approximation of Log_2 X	**Restoration of MSB**
Char. Mantissa	Char. Mantissa
0100.10010000	0100.11001000

The value of the characteristic is 4 so the mantissa must be shifted to the right until MSB is aligned with the 2^4 position.

$$2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

The completion of this operation restores the value of X (X = 25) and is the procedure used to find an antilog. *Figure 8* is a flow chart for finding an antilog using this procedure. Ths implementation in source code is shown in *Figure 9.*


START
ALOG:
MOVE MANTISSA TO TEMPORARY MEMORY LOCATION
CLEAR MANTISSA AREA. SET X = Φ
SET CARRY = 1 TO FORCE MSB OF X
$SLX:
SHIFT CARRY INTO X
SUBTRACT 1 FROM CHARACTERISTIC
$TST:
IS CHARACTERISTIC NEGATIVE?
YES
RETURN
NO
SHIFT MANTISSA LEFT 1 INTO CARRY

TL/DD/6942-3

FIGURE 8. Flow Chart for Conversion to Antilog

```
COP CROSS ASSEMBLER PAGE 3
LOGS

                    . FORM        ; -----→ CONVERT TO ANTILOG ←----- ;


                    ; THE FOLLOWING SUBROUTINE CONVERTS THE STRAIGHT LINE
                    ; THE APPROXIMATION OF A BASE 2 LOGARITHM TO ITS CORRESPONDING
                    ; ANTILOG. UPON EXIT FROM THE ROUTINE THE CONTENTS OF CH
                    ; WILL BE EQUAL TO THE HEXADECIMAL VALUE OF 'ΦF'.

                                  . LOCAL


00D     A4          ALOG:         JSRP        SDB2             ; SET ACC TO 0.
00E     00          CLRA                                       ; CLEAR MANTISSA AREA.
00F     36                        X           03               ; AND MOVE MANTISSA TO
010     34                        XIS         03               ; TEMPORARY STORAGE.
011     00                        CLRA                         ; LEAVE POINTER AT LO
012     36                        X           03               ; ORDER OF MANTISSA.
013     37                        XDS         03
014     22                        SC                           ; RESTORE MSB OF X.
015     D8                        JP          $SLX
01      A9          $SLM:         JSRP        SHLR             ; SHIFT REMAINDER
                                                               ; LEFT INTO CARRY.
017     A3                        JSRP        SDR2             ; MOVE BACK 2 DIGITS.
018     AA          $SLX:         JSRP        SHLC             ; SHIFT X LEFT 1.
019     05                        LD                           ; LOAD CHARACTERISTIC.
01A     5F          $TST:         AISC        -1               ; CHARACTERISTIC -1.
01B     48          $LST:         RET                          ; IF NO CARRY — FINIS.
01C     36                        X           03               ; STORE REMAINDER AND MOVE
                                                               ; DOWN ONE REGISTER.
01D     A4                        JSRP        SDB2             ; MOVE BACK 2 DIGITS.
01E     D6                        JP          $SLM             ; DO IT AGAIN.


                    ; 4 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
                    ; PROGRAM: SDB2, SDR2, SHLR, SHLC.


```

TL/DD/6942-6

FIGURE 9

Using the linear approximation technique just described, some error will result when converting any value of X that is not an even power of 2.

Figure 10 contains a table of correct base 2 logarithms for values of X from 1 through 32 along with the error incurred for each when using linear approximation. Notice that no error results for values of X that are even powers of 2. Also notice that the error incurred for multiples of even powers of 2 of any given value of X is always the same.

Value of X	Error
5	0.12
2 × 5 = 10	0.12
4 × 5 = 20	0.12
3	0.15
2 × 3 = 6	0.15
4 × 3 = 12	0.15
8 × 3 = 24	0.15

X	Hexadecimal Log Base	Linear Approximation of Log Base 2	Error Hexadecimal	$E_M - 1 + \frac{EM - EM - 1}{2}$
1	0.00	0.00	0.00	
2	1.00	1.00	0.00	
3	1.95	1.80	0.15	
4	2.00	2.00	0.00	
5	2.52	2.40	0.12	
6	2.95	2.80	0.15	
7	2.CE	2.C0	0.0E	
8	3.00	3.00	0.00	
9	3.2B	3.20	0.0B	
10	3.52	3.40	0.12	
11	3.75	3.60	0.15	
12	3.95	3.80	0.15	
13	3.B3	3.A0	0.13	
14	3.CE	3.C0	0.0E	
15	3.E8	3.E0	0.08	
16	4.00	4.00	0.00	
17	4.16	4.10	0.06	0.03
18	4.2B	4.20	0.0B	0.09
19	4.3F	4.30	0.0F	0.0D
20	4.52	4.40	0.12	0.11
21	4.67	4.50	0.17	0.15
22	4.75	4.60	0.15	0.16
23	4.87	4.70	0.17	0.16
24	4.95	4.80	0.15	0.16
25	4.A4	4.90	0.14	0.15
26	4.B3	4.IA0	0.13	0.14
27	4.C1	4.B0	0.11	0.12
28	4.CE;	4.C0	0.0E	0.10
29	4.DB	4.D0	0.0B	0.0D
30	4.E8	4.E0	0.08	0.0A
31	4.F4	4.F0	0.04	0.06
32	5.00	5.00	0.00	0.02
33		5.1-		

FIGURE 10. Error Incurred by Linear Approximation of Base 2 Logs

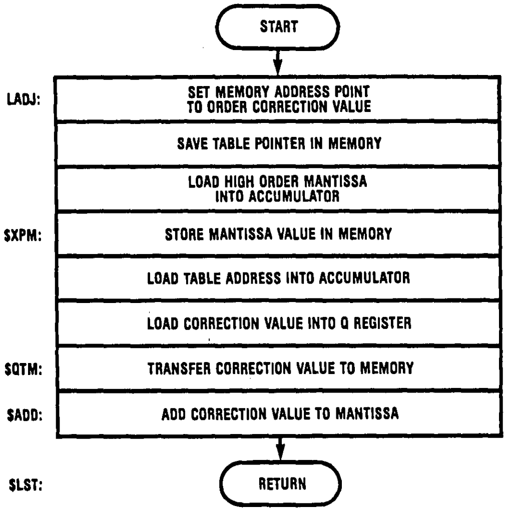
An error that repeats in this way is easily corrected using a look-up table. The greatest absolute error will occur for the least value of X not an even power of 2, X = 3, is about 8%. A 4 point correction table will eliminate this error but will move the greatest uncompensated error to X = 9 where it will be about 4%. This process continues until at 16 correction points the maximum error for the absolute value of the logarithm is less than 1 percent. This can be reduced to 0.3 percent by distributing the error. Interpolated error values are listed in *Figure 10* and are repeated in *Figure 11* as a binary table.

High Order 4 Mantissa Bits	Binary Correction Value	Hexadecimal Correction Value
0000	0000 0000	0 0
0001	0000 1001	0 9
0010	0000 1101	0 3
0011	0001 0001	1 1
0100	0001 0101	1 5
0101	0001 0110	1 6
0110	0001 0110	1 6
0111	0001 0110	1 6
1000	0001 0101	1 5
1001	0001 0100	1 4
1010	0001 0010	1 2
1011	0001 0000	1 0
1100	0000 1101	0 D
1101	0000 1010	0 A
1110	0000 0110	0 6
1111	0000 0010	0 2

FIGURE 11. Correction Table for L_2 X Linear Approximations

Notice in *Figure 10* that left justification of the mantissa causes its high order four bits to form a binary sequence that always corresponds to the proper correction value. This works to advantage when combined with the COP400 LQID instruction. LQID implements a table look-up function using the contents of a memory location as the address pointer. Thus we can perform the required table look-up without disturbing the mantissa.

Figure 12 is the flow chart for correction of a logarithm found by linear approximation. *Figure 13* is its implementation in COP400 assembly language. Notice that there are two entry points into the program. One is for correction of logs (LADJ:), the other is for correction of a value prior to its conversion to an antilog (AADJ:).


START
LADJ:
SET MEMORY ADDRESS POINT TO ORDER CORRECTION VALUE
SAVE TABLE POINTER IN MEMORY
LOAD HIGH ORDER MANTISSA INTO ACCUMULATOR
$XPM:
STORE MANTISSA VALUE IN MEMORY
LOAD TABLE ADDRESS INTO ACCUMULATOR
LOAD CORRECTION VALUE INTO Q REGISTER
$QTM:
TRANSFER CORRECTION VALUE TO MEMORY
$ADD:
ADD CORRECTION VALUE TO MANTISSA
$LST:
RETURN

TL/DD/6942-4

FIGURE 12. Flow Chart for Correction of a Value Found by Straight Line Approximation

```
COP CROSS ASSEMBLER     PAGE:  4
LOGS

                     . FORM      ;-----> ADJUST VALUE OF LOGARITHM <----- ;

                                 . LOCAL



                     ; THE FOLLOWING TABLE IS USED DURING THE CORRECTION OF VALUES
                     ; FOUND BY STRAIGHT LINE APPROXIMATION. IT IS PLACED HERE IN
                     ; ORDER TO ALIGN ITS BEGINNING ELEMENT WITH A ZERO ADDRESS AS
                     REQUIRED BY THE LQID INSTRUCTION.

  01F     44                     NOP                                    ; REGISTER WITH ZERO ADDRESS.
  020     03         TPLS:       . WORD       03,09,0D,011
  021     09
  022     0D
  023     11
  024     15         . WORD      015,016,016,016
  025     16
  026     16
  027     16
  028     15                     . WORD       015,014,012,010
  029     14
  02A     12
  02B     10
  02C     0D                     . WORD       0D,0A,06,02
  02D     0A
  02E     06
  02F     02

                     ; THE FOLLOWING SUBROUTINE ADJUSTS THE VALUE OF A BASE 2
                     ; LOGARITHM FOUND BY STRAIGHT LINE APPROXIMATION. THE
                     ; CORRECTION TERMS ARE TAKEN FROM THE TABLE ABOVE. THE
                     ; SUBROUTINE HAS 2 ENTRY POINTS:
                     ;
                     ;           LADJ: — ADJUSTS A VALUE DURING CONVERSION TO A LOG
                     ;
                     ;           AADJ: — ADJUSTS A VALUE DURING CONVERSION TO ANTILOG
                     ;
                     ; THE CARRY FLAG IS SET UPON ENTRY TO DISTINGUISH BETWEEN LOG
                     ; (C=1) AND ANTILOG (C=0) CONVERSIONS. DURING A LOGARITHM
                     ; CONVERSION THE VALUE FOUND IN THE ABOVE TABLE IS ADDED TO
                     ; THE MANTISSA. DURING AN ANTILOG CONVERSION THE VALUE FOUND
                     ; IN THE ABOVE TABLE IS SUBTRACTED FROM THE MANTISSA.


  030     32         AADJ:       RC                                     ; C=0 FOR ANTILOG
  031     F3                     JP           $LD                       ; CONVERSION.
  032     22         LADJ:       SC                                     ; C=  FOR LOG2 ADJ.
  033     05         $LD         LD                                     ; MOVE ADDRESS POINTER BACK
  034     07                     XDS                                    ; ONE LOCATION.
  035     05                     LD                                     ; LOAD CONTENTS OF HI MANTISSA
  036     37                     XDS          03                        ; AND STORE IT IN THE LO ORDER
  037     06                     X                                      ; OF THE TEMP MEMORY LOCATION.
  038     00                     CLRA                                   ; SET TABLE POINTER
  039     52                     AISC         TBL                       ; (ACC) TO TABLE ADDRESS.

COP CROSS ASSEMBLER     PAGE:  5
LOGS

  03A     BF                     LQID                                   ; LOAD CORRECTION VALUE TO Q.
  03B     332C       $GTM:       CQMA                                   ; TRANSFER Q REGISTER
  03D     04                     XIS                                    ; CONTENTS TO MEMORY.
  03F     07                     XDS
  03F     20                     SKC                                    ; ANTILOG?

  040     80                     JSRP         COMP                      ; YES — COMPLIMENT.
  041     98         $ADD:       JSRP         ADRO                      ; ADD CORRECTION VALUE
                                                                        ; TO MANTISSA.
  042     35                     LD           03                        ; SET POINTER TO
  043     48         $LST:       RET          ; CHARACTERISTIC AND
                                                                        ; RETURN.

                     ; 2 ROUTINES ARE CALLED FROM THE SUBROUTINE PAGE BY THIS
                     ; PROGRAM: COMP, ADRO

          0020                   V1=TPLS&OFF
          0002                   TBL=V1/16



```

TL/DD/6942-7

FIGURE 13

Subroutines Used by the Log and Antilog Programs

```
COP CROSS ASSEMBLER     PAGE:  6
LOGS

                                        . FORM

               0080        . PAGE 02                     ; ----→ SUBROUTINES ←---- ;

                           ; THE FOLLOWING ROUTINES RESIDE ON THE SUBROUTINE PAGE. THEY
                           ; ARE CALLED BY THE LOGS PROGRAM BUT ARE GENERAL PURPOSE IN
                           ; NATURE AND FUNCTION AS UTILITY ROUTINES.



                                        ; ----→ COMPLEMENT 8 BITS ←---- ;

                                        . LOCAL

                           ; THIS ROUTINE FORMS IN MEMORY THE 2'S COMPLEMENT OF THE TWO
                           ; ADJACENT DIGITS IDENTIFIED BY THE ADDRESS POINTER. THE
                           ; CONTENTS OF THE ADDRESS POINTER ARE NOT ALTERED.

                           ; THERE ARE TWO ENTRY POINTS:
                           ;
                           ; COP: COMPLEMENT 8 BITS.
                           ;
                           ; CMPE: EXTEND THE COMPLEMENT TO AN ADDITIONAL 8 BITS
                           ;

080     22                 COMP:        SC
081     00                 CMPE:        CLRA                                ; SET MINUEND=0
082     06                              X                                   ; AND STORE IN MEMORY.
083     10                              CASC
084     44                              NOP                                 ;
085     04                              XIS                                 ;
086     00                              CLRA                                ; SET MINUEND=0
087     06                              X                                   ; AND STORE IN MEMORY.
083     10                              CASC                                ;
089     44                              NOP
08A     04                              XIS                                 ;
08B     44                              NOP                                 ; AVOID SKIP IF DIGIT 15.
08C     A4                              JP          SDB2                    ; RETURN THRU SDB2
                                                                            ; TO RESTORE POINTER.




                           ; ----→ ADD 8 BITS IN ADJACENT REGISTERS ←---- ;

                                        . LOCAL



                           ; THIS ROUTINE ADDS TWO BINARY DIGITS (8 BITS) FROM ANY REGISTER
                           ; TO THE CORRESPONDING TWO BINARY DIGITS IN EITHER REGISTER
                           ; IMMEDIATELY ADJACENT. THERE ARE THREE ENTRY POINTS:
                           ;
                           ;            LADR: — RESET CARRY AND ADD 2 DIGIT PAIRS
```

TL/DD/6942-8

```
COP CROSS ASSEMBLER     PAGE:  7
LOGS

                     ;                 LADD: — ADD 2 DIGIT PAIRS WITH UNMODIFIED CARRY
                     ;                 ADD1: — ADD 2 SINGLE DIGITS WITH UNMODIFIED CARRY




08D     32           LADR:             RC                                   ; RESET CARRY PRIOR TO ADD.
08E     15           LADD:             :D             01                    ; LD ADDEND AND MOVE TO ADJ REG
08F     30                             ASC                                  ; ADD AUGEND.
090     44                             NOP                                  ; AVOID CARRY!
091     14                             XIS            01                    ; STORE SUM AND MOVE TO ADDEND
092     15           ADD1:             LD             01                    ; REPEAT PROCESS
093     30                             ASC                                  ; FOR
094     44                             NOP                                  ; HIGH ORDER
095     14                             XIS            01                    ; DIGIT.
096     44                             NOP                                  ; AVOID SKIP IF DIGIT 15.
097     48           $LST:             RET                                  ; FINISHED — RETURN!!!!




                                       ; ----→ ADD 8 BITS IN OPPOSITE REGISTERS ←---- ;

                                       . LOCAL



                     ; THIS ROUTINE ADDS TWO BINARY DIGITS (8BITS) FROM ANY REGISTER
                     ; TO THE CORRESPONDING TWO BINARY DIGITS IN EITHER REGISTER
                     ; DIRECTLY OPPOSITE. THERE ARE THREE ENTRY POINTS:
                     ;
                     ;                 ADR0: — RESET CARRY AND ADD 2 DIGIT PAIRS
                     ;                 ADD0: — ADD 2 DIGIT PAIRS WITH UNMODIFIED CARRY
                     ;                 AD01: — ADD 2 SINGLE DIGITS WITH UNMODIFIED CARRY




098     32           ADR0:             RC                                   ; RESET CARRY PRIOR TO ADD.
099     35           ADD0:             LD             03                    ; LD ADDEND AND MOVE TO OPP REG
09A     30                             ASC                                  ; ADD AUGEND.
09B     44                             NOP                                  ; AVOID CARRY!
09C     34                             XIS            03                    ; STORE SUM AND MOVE TO ADDEND.
09D     15           AD01:             LD             01                    ; REPEAT PROCESS
09E     30                             ASC                                  ; FOR
09F     44                             NOP                                  ; HIGH ORDER
0A0     34                             XIS            03                    ; DIGIT.
0A1     44                             NOP                                  ; AVOID SKIP IF DIGIT 15.
0A2     48           $LST:             RET                                  ; FINISHED — RETURN!!!!



                                       ; ----→ SET DIGIT ADDRESS BACK TWO ←---- ;

```

TL/DD/6942-9

```
COP CROSS ASSEMBLER     PAGE:  8
LOGS

                                  . LOCAL

                         ; THIS ROUTINE SUBTRACTS 2 FROM THE CONTENTS OF THE
                         ; DIGIT POINTER (B REGISTER). THE CONTENTS OF THE
                         ; ACCUMULATOR ARE LOST IN THE PROCESS. THE USE OF
                         ; SDB2 ALLOWS ADDRESSING WITHIN THE LOGS SUB
                         ; ROUTINE TO BE RELATIVE TO THE CONTENTS OF THE
                         ; ADDRESS POINTER (B REGISTER) UPON ENTRY.
                         ; SDB2 IS COMMONLY USED IN BYTE OPERATIONS TO RESTORE THE
                         ; DIGIT POINTER TO THE LOW ORDER POSITION.
                         ; THERE ARE TWO ENTRY POINTS:
                         ;
                         ; SDR2:      SET DIGIT ADDRESS BACK 2 AND MOVE TO OPPOSITE REGISTER.
                         ;
                         ; SDB2: SET DIGIT ADDRESS BACK 2 RETAINING PRESENT REGISTER.



     0A3     35          SDR2:      LD        03                 ; MOVE TO OPPOSITE REGISTER.
     0A4     4E          SDB2:      CBA                          ; PLACE DIGIT COUNT IN ACC.
     0A5     5E                     AISC      -2                 ; SUBTRACT 2.
     0A6     44                     NOP                          ; SHOULD ALWAYS SKIP.
     0A7     50                     CAB                          ; PUT DIGIT COUNT BACK.
     0A8     48                     RET                          ; FINISHED — RETURN!!


                                  ; ----→ SHIFT LEFT ←---- ;

                                  . LOCAL

                         ; THIS ROUTINE SHIFTS LEFT THE CONTENTS OF TWO MEMORY
                         ; LOCATIONS ONE BIT. THERE ARE THREE ENTRY POINTS:

                         ;                    SHLR: RESETS THE CARRY BEFORE SHIFTING
                         ;                          IN ORDER TO FILL THE LOW ORDER
                         ;                          BIT POSITION WITH A 0.
                         ;
                         ;                    SHLC: SHIFTS THE STATE OF THE CARRY INTO
                         ;                          THE LOW ORDER BIT POSITION.
                         ;
                         ;                    SHL1: SHIFTS LEFT THE CONTENTS OF ONLY
                         ;                          ONE MEMORY LOCATION. THE STATE
                         ;                          OF THE CARRY IS SHIFTED INTO THE
                         ;                          LOW ORDER POSITION OF MEMORY.



     0A9     32          SHLR:      RC                           ; CLEAR CARRY PRIOR TO SHIFT.
     0AA     05          SHLC:      LD                           ; LOAD FIRST MEM DIGIT.
     0AB     30                     ASC                          ; DOUBLE IT.
     0AC     44                     NOP                          ; AVOID SKIP.
     0AD     04                     XIS                          ; STORE SHIFTED DIGIT.
     0AE     05          SHL1:      LD                           ; LOAD NEXT MEM DIGIT.
     0AF     30                     ASC                          ; DOUBLE IT TOO.

COP CROSS ASSEMBLER     PAGE:  9
LOGS

     0B0     44                     NOP                          ; AVOID SKIP, IF ANY
     0B1     04                     XIS                          ; STORE SHIFTED DIGIT.
     0B2     48          $LST:      RET                          ; FINISHED — RETURN!



                                    . END
```

TL/DD/6942-10

RAM Keep-Alive

National Semiconductor
COP Brief 6

A COPS™ application is a small scale computer system and the design of a power shut-down is not trivial. During the time that power is available, but out of the designed operating range, the system must be prevented from doing anything to harm protected data. This will typically involve some type of external protection of timing circuit.

There is an option on the COP420, 420L, and 410L parts called "RAM Keep-Alive" that provides a separate power supply to the RAM area of the chip via the CKO pin. The application of power to the RAM while the remainder of the chip has been powered down via V_{CC} will keep the RAM "alive".

However, the integrity of data in the RAM is not only a function of power but is also influenced by transient conditions as power is removed and reapplied. During power-on, the Power On Reset (POR) circuit will keep transients from causing changes in the RAM states. The condition of power loss will have some probability of data change if external control is not used.

At some point below the minimum operating voltage certain gates will no longer respond properly while others may still be functional until a much lower voltage. During this transition time any false signal could cause a false write to one or more cells. Another effect could be to turn on multiple address select lines causing data destruction.

Testing the rate of data change is very difficult because it must be done on a statistical basis with many turn/on-turn/off cycles. Two factors have a major bearing on the numbers derived by testing. One is to call any change in a related data block a failure, even though more than one bit in that block may have changed (this latter case may well be due to the "address select mode"). The second factor is that without massive instrumentation it is impossible to examine the data after each power cycle. Indeed, to do so might have caused errors!

By running the power cycle for a period of time and then looking for changes, one could overlook multiple changes thus reducing the error rate. This has been minimized by more frequent checking which indicates that the errors are spread out randomly over time.

With a power supply that drops from 4.5 to 2V in approximately 100 ms, the drop-out rate is 1 in 5k to 6k power cycles. Reducing the voltage fall time will cause an improvement in the number of cycles per drop-out. This will reach a limit condition of a very high number (1 per 1 million?) when the power falls within one instruction cycle (4–10 μs for the 420, 15–40 μs for the "L" parts). Attaining very rapid fall time may cause problems due to the lack of decoupling/bypass capacitance. By inserting an electronic switch between the regulator and V_{CC} of the COP chip one might be able to meet this type of fall time. By implication some type of sensing is required to cause the switching.

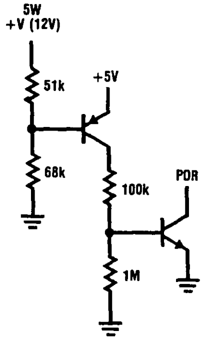
The desirable approach is to force the COP reset input to zero before the voltage falls below 4.5V. This provides a drop out rate of approximately 1 in 50k for the "L" parts and 1 in 100k for the 420. By also stopping the clock of the "L" parts they can achieve a drop-out rate similar to the 420. While not perfect, the number of cycles between data error should be considered with respect to the needs of the application.

The external circuitry to control the chip during the power transition has several implementations each one being a function of the application. The simplest hardware is found in a battery powered (automotive) application. The circuit must sense that the switched 12V is falling (e.g., at some value much below 12V and still greater than 5V). This can be done by using the unswitched 12V as a reference for a divider to a nominal voltage of 8V. As the switched 12V drops below the reference a detector will turn on a clamp transistor to a series switch, the POR, and/or the clock circuit *(Figure 1)*. It should be noted that this draws current during the absence of the switched 12V circuit.

In non-automotive usage a similar circuit can be used where there is a stable reference voltage available to use with the comparator/clamp. Thus a 3.6V rechargable Ni-Cad battery could be used as the reference voltage and V_{RAM} if the appropriate divider is used to level shift to this operating range.

In AC line-powered applications, a similar method could be used with the raw DC being sensed for drop. Another method would be to sense that the line had missed 2–3 cycles either by means of a charge pump or peak detection technique. This will provide the signal to turn on the clamp. One must make this faster than the time to discharge the output capacitance of the power supply, thus assuring that the clamp has performed its function before the supply falls below spec value.

In conclusion, to protect the data stored in RAM during power-off cycle, the POR should go low before the V_{CC} power drops below spec and come up after V_{CC} is within spec. The first item must be handled with an external circuit like *Figure 1* and the latter by an RC per the data sheet.



TL/DD/6946-1

FIGURE 1

Analog to Digital Conversion Techniques With COPS™ Family Microcontrollers

National Semiconductor
COP Note 1
Leonard A. Distaso

TABLE OF CONTENTS

1.0 Introduction

A variety of techniques for performing analog to digital conversion are presented. The COP420 microcontroller is used as the control element in all cases. However, any of the COPS family of microcontrollers could be used with only minor changes in some component values to allow for different instruction cycle times.

Indirect analog to digital converters are composed of three basic building blocks:

- D/A Converter
- Comparator
- Control logic

In a software driven system the D/A converter and comparator are present but the control logic is replaced by instruction sequences. There are a variety of software/hardware techniques for implementing A/D converters. They differ primarily in their approach to the included D/A. There are two primary approaches to the digital to analog conversion which can in turn be divided into a number of sub-categories:

- D/A as a function of weight closures
  - — R/2R ladder
  - — Binary weighted ladder
- D/A as function of time
  - — RC exponential charge
  - — Linear charge/discharge (dual slope)
  - — Pulse width modulation

These techniques should be generally familiar to persons skilled in the electronic art. The objective here is to illustrate the application of these established methods to a low cost system with a COPS microcontroller as the intelligent control element. Circuit configurations are provided as well as the appropriate flow charts and code to implement the function.

Some mathematical and theoretical analysis is presented as an aid to understanding the various techniques and their limits. However, it is not the purpose here to provide a definitive theoretical analysis of the analog to digital conversion process or of the various techniques described.

2.0 Simple Capacitor Charge Time Measurement

2.1 BASIC APPROACH

General

Perhaps the simplest means to perform an analog to digital conversion is to charge a capacitor until the capacitor voltage is equal to the unknown voltage. The capacitor voltage and the unknown are compared by means of a standard analog comparator. The unknown is determined simply by counting, in the microcontroller, the amount of time it takes for the charge on the capacitor to reach a value equal to the unknown voltage. The capacitor voltage is given by the standard capacitor charge equation:

$$V_C = V0 + [V1 - V0][1 - e^{**}(-t/RC)]$$

where: V_C = capacitor voltage

V0 = "dischage voltage" — low level voltage

V1 = high level voltage

The most obvious problem with this method, from the standpoint of software implementation, is the nonlinearity of the relationship. This can be circumvented in several ways. First of all, a routine to calculate the exponential can be implemented. This, however, usually requires too much code if the exponential routine is not otherwise required in the program. Alternatively, the range of input voltages can be restricted so that only a portion of the capacitor charge curve — which can be approximated with a linear relationship or with some minor straight time curve fitting — is used. Finally, a look up table can be used which will effectively convert the measured time to the appropriate voltage. The look up table has the advantage that all the math can be built into the table, thereby simplifying matters significantly. If arithmetic routines are going to be used, it is clear that the relationship is simplified if V0 is 0V because it then drops out the equation.

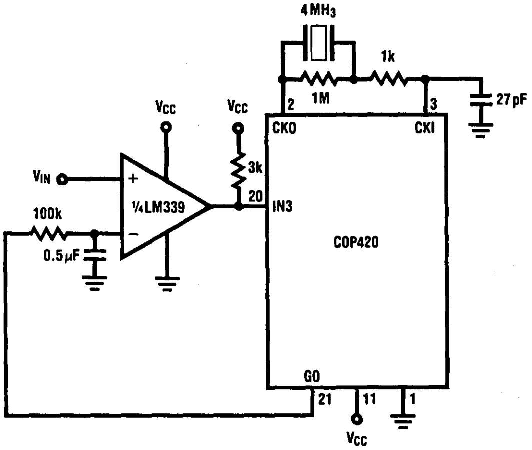
BASIC CIRCUIT IMPLEMENTATION

The circuit in *Figure 1* is the basic implementation of the capacitor charge method of A/D conversion. The selection of input and output used is arbitrary and is dictated by general system considerations. V0 is the "0" level of the G output and V1 is the "1" level of the output. The technique is basically to discharge the capacitor to V0 (which is ideally ground) and then to apply V1 and increment an internal counter until the comparator changes state. The flow chart and code for this implementation are shown in *Figure 2*.

ACCURACY CONSIDERATIONS

The levels reached by the microcontroller output constitute one of the more significant problems with this basic implementation. The levels of V1 and V0 are not V_{CC} and ground as would be desired. The level is defined by the load on the output, the value of V_{CC}, and the device itself. Furthermore, these levels are likely to change from device to device and over temperature. To be sure, the output values will be at least those given in the data sheet, but it must be remembered that those values are minimum high voltages and maximum low voltages. Typically, the high value will be greater than the spec minimum and the low value will be lower than the spec maximum. In fact, with a light load the values will be close to V_{CC} and ground. Therefore, in order to obtain any accurate result for a voltage measurement the exact values of V1 and V0 need to be measured and somehow stored in the microcontroller. Typical values of these voltages can be measured experimentally and an average could be used for final implementation.

The other problem associated with the levels is that the capacitive load on the output line is substantial and far in excess of the values used when specifying the characteristics of the various COP420 outputs. The significant effect of this is that it will take longer than "normal" for the output to reach its maximum value. In addition, it is likely that there will be dips in the output as it rises to its maximum value since the capacitor will start to draw charging current from the output. All of this will be fast relative to the other system times. Still it will affect the result since the level to which the capacitor is attempting to charge is not being applied uniformly and "instantaneously". It can be viewed as though the voltage V1 is bouncing before it stabilizes.



TL/DD/6935-01

Crystal oscillator values chosen to give 4 μs cycle time with divide by 16 option selected on COP 420 CKO/CKI Pins

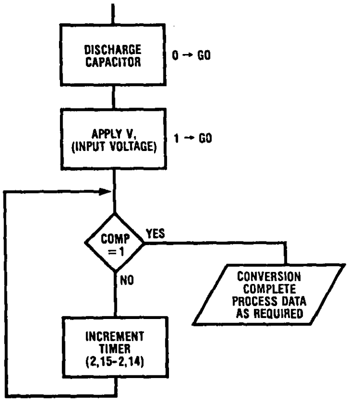
$V_{CC} = +5V$

FIGURE 1. Basic Capacitor Charge Technique

```
          OGI      0        ;TURN OFF G TO DISCHARGE CAPACITOR
          ; INSERT SOME DELAY TO MAKE SURE CAPACITOR DISCHARGED
          ;USING 12 BIT COUNTER, BUT ONLY UPPER 8 USED IN TABLE
          ;LOOK UP DUE TO ACCURACY OF RC CHARGE METHOD.  THE OTHER
          ;BITS COULD BE USED BUT THE COMPLICATIONS ARE NOT WORTH
          ;THE EFFORT FOR THIS PARTICULAR TECHNIQUE.  ALSO, HERE THE
          ;INPUT RANGE IS RESTRICTED SO THAT THE TOP 3 BITS ARE ZERO

RCAD:     OGI      1        ;TURN ON THE G LINE
INCR:     LBI      2,13     ;BINARY INCREMENT OF 12 BIT COUNTER
BINPL8:   SC                ;LOWER FOUR BITS WILL BE DISCARDED
BINPL1:   CLRA              ;ONLY TOP BITS USED IN TABLE LOOK UP
          ASC               ;SPEED WOULD BE IMPROVED IF THE ADD WERE
          NOP               ;STRAIGHT LINE CODED-BUT COSTS MORE CODE
          XIS
          JP       BINPL1
          ININ              ;READ IN3 TO SEE IF COMPARATOR CHANGED
          AISC     8
          JP       END
          CLRA
          JP       INCR
END:      OGI      0        ;TURN OFF THE G LINE AND DISCHARGE C
          ;DO ARITHMETIC HERE OR LOOK UP TABLE OR WHATEVER IS
          ;REQUIRED--SAMPLE LOOK UP TABLE CONTROL INDICATED BELOW
          ;SAMPLE TABLE WRITTEN CORRECTING FOR THE EXPONENTIAL
          ;RELATIONSHIP.  THE TABLE ALSO INCORPORATES A CONVERSION
          ;TO BCD.  THE VALUE IN THE TABLE IS THE RATIO OF
          ;THE CAPACITOR VOLTAGE V TO THE MAXIMUM VOLTAGE VMAX.
          ;THE NUMBER IS A TWO DIGIT BCD FRACTION.  WE ARE USING
          ;A 5 BIT COUNT IN THIS EXAMPLE.  ADDRESSING ARBITRARILY
          ;SET UP ASSUMING THAT CONTROL CODE IS IN PAGE 0 (OTHER
          ;THAN AT ADDRESS 0) AND THAT THE TABLE THEREFORE IS IN
          ;PAGE 1 (STARTING AT HEX ADDRESS 040).
          ;
          LBI      2,15     ;POINT TO TOP 4 BITS
          XDS               ;TOP 4 IN A,POINTING TO LOWER 4 IN 2,14
          AISC     4        ;THIS MERELY ADJUSTING FOR ADDRESS--NO
                            ;OTHER FUNCTION
          LQID              ;DO THE LOOK UP
          CQMA              ;FETCH THE ADJUSTED VALUE FROM Q
           ; THE ADJUSTED VALUE IS NOW IN A AND M.  FROM THIS POINT MAY
           ;USE THE VALUE IN OTHER CALCULATIONS OR OUTPUT THE INFORMATION,
           ;OR WHATEVER MAY BE REQUIRED BY THE APPLICATION.
          LBI      2,13     ;CLEAR THE COUNTER
          STII     0
          STII     0
          STII     0
          JP       RCAD:    ;JUMP BACK AND REPEAT

          .=X'040           ;SET UP TABLE ADDRESS
          .WORD 000,003,006,008  ;SET UP THE TABLE VALUES
          .WORD 011,014,016,019  ;HERE, COMPENSATED FOR EXPONENTIAL
          .WORD 021,023,026,028  ;AND CONVERTED TO BCD FRACTION
          .WORD 030,032,034,036  ;TABLE VALUE IS RATIO V/VMAX
          .WORD 038,039,041,043
          .WORD 045,046,048,049
          .WORD 051,052,053,055
          .WORD 056,057,059,060
```

TL/DD/6935-55

FIGURE 2A. Typical RC Charge A/D Code


DISCHARGE CAPACITOR
0 → GO
APPLY V, (INPUT VOLTAGE)
1 → GO
COMP =1
YES
NO
CONVERSION COMPLETE PROCESS DATA AS REQUIRED
INCREMENT TIMER (2,15-2,14)

TL/DD/6935-2

FIGURE 2B. Charge Flow Chart

A more general problem is that of the tolerance of RC time constant. The value of the voltage with respect to time is obviously related to the RC value. Therefore, a change in that value will result in a change in the voltage for a given time period t. The graph in *Figure 3* illustrates the effect of a ±10% variation in the RC value upon the voltage measured for a given time t. If one cares to work out the math, it comes out that the error is an exponential relationship in much the same manner as the capacitor voltage itself. The maximum error induced for ±10% RC variation is ±3.9%.

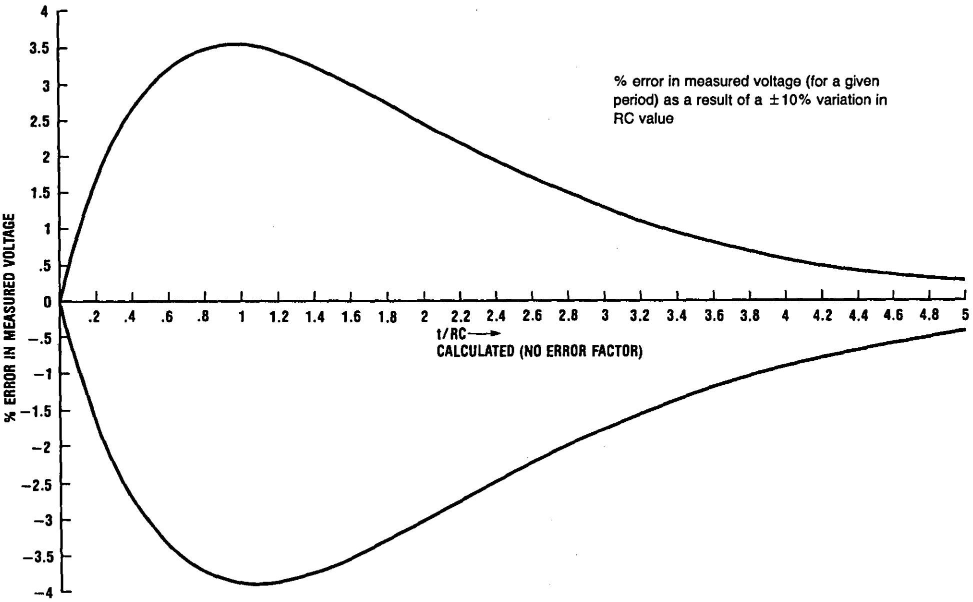
Remember also that we are measuring time. Therefore variation in the RC value will have a direct, linear effect on the time required to measure a given voltage. It is also necessary that the time base for the COP420 be accurate. A variation in the accuracy in the operating frequency of the COP420 will have a direct impact on the accuracy of the result.

Given the errors mentioned so far and assuming that no changes are made in the hardware, the accuracy of the technique then is determined by the resolution of the time measurement. This is improved in two ways: increase the RC time constant so that there is a smaller change in capacitor voltage for a given time period or try to minimize the loop time required to increment the counter. Lengthening the RC time constant is easier but the cost is increased conversion time. The minimum time to increment a 5 to 8 bit binary counter and test an input is 13 cycle times. For a 9 to 12 bit binary counter this minimum time is 17 cycle times. Note also that the minimum time to perform the function does not necessarily correspond to the minimum number of code words required to implement the function. At a cycle time of 4 μs, the 13 cycle times correspond to 52 μs.

2.2 ACCURACY IMPROVEMENTS

Several options are available if it is desired to improve the accuracy of this method. Three such improvements are shown in *Figure 4*. *Figure 4A* is the smallest change. Here a pullup resistor has been added to the G output line and the G line is run open drain internally, i.e., the internal pullup is removed. This improves the "bounce" problem mentioned earlier. The G line will go to the high state and remain there with this setup. However, the addition of the resistor does little more than eliminate the bounce. The degree of improvement is not great, but it is an easy way to eliminate a minor source of error.

Figure 4B is the next step. A 74C04 is used as a buffer. The 74C04 was chosen because of its symmetric output characteristics. Any CMOS gate with such characteristics could be used. The software can easily be adjusted to provide the proper polarity. The COP420 output drives a CMOS gate which in turn drives the RC network. This change does make significant improvements in accuracy. With a light



TL/DD/6935-3

FIGURE 3

load the CMOS gate will typically swing from ground to V_{CC} and its output level is not as likely to be affected by the capacitor discharge.

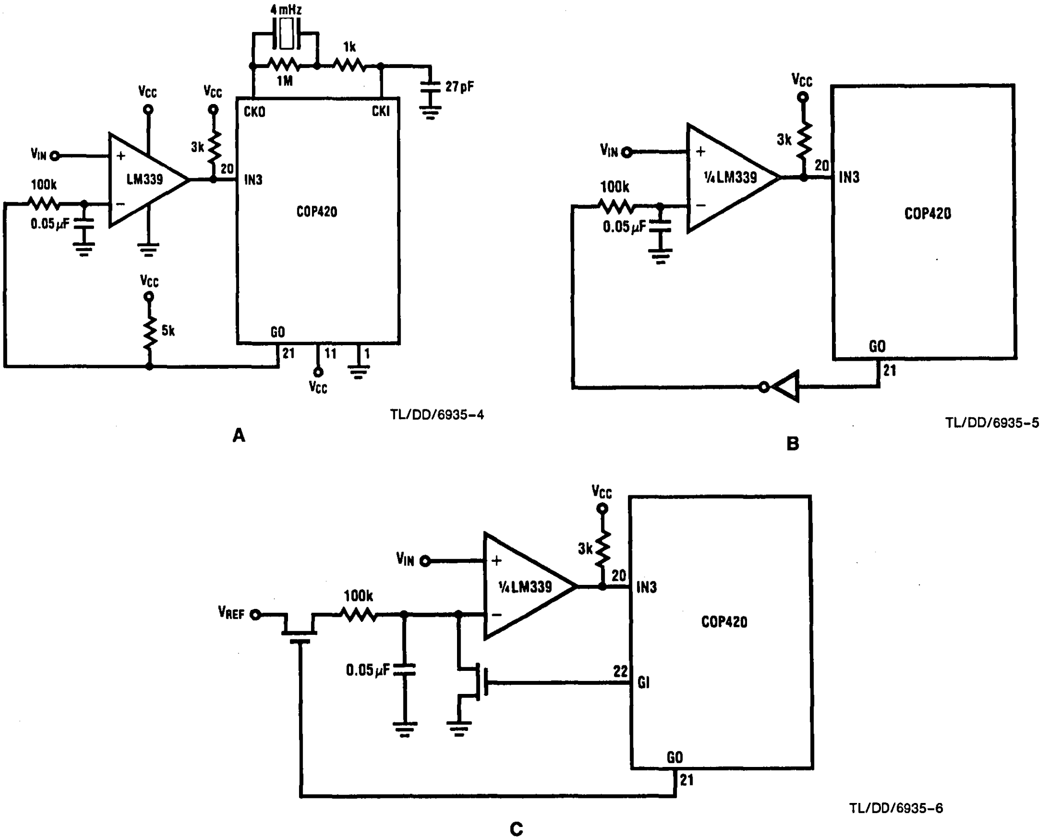
Figure 4C is the best approach, but it involves the greatest component cost. Here two G outputs are controlling analog switches. Ground is connected to the RC network to discharge the capacitor, and a positive reference is used to charge the capacitor. This reference can be any suitable voltage source: zener diodes, V_{CC}, etc. The controlling voltage tolerance is now clearly the tolerance of the reference. Precise voltage references are readily obtainable. *Figure 4C* also shows an analog switch connected directly across the capacitor to speed up the capacitor discharge time. When using this version of the basic scheme, remember to include the 'on' resistance of the analog switch connected to V_{REF} in the RC calculation. Failure to do so will introduce error into the result.

Note that the LM339 is a quad comparator. If these comparators are not otherwise needed in the system, they can be used in much the same manner as the CMOS gate mentioned above. They can be used to buffer the output of the COPS device and to reset the capacitor, or whatever other function is required. This has the advantage of fully utilizing the components in the system and eliminates the need to add another package to the system.

2.3 CONCLUSIONS

This approach is an inexpensive way to perform an A/D conversion. However, it is not that accurate. With a 10% V_{CC} supply and a 10% tolerance in the RC value and 10% variation in the oscillator frequency the best that can be hoped for is about 25% accuracy. If a 1% reference voltage is used, this accuracy becomes about 15%.

Under laboratory conditions—holding all variables constant and using precise measured values in the calculations—the configuration of *Figure 2* yielded 5 bit accuracy over an input range of 0 to 3.5V. Over the same range and under the same conditions, the circuit of *Figure 4B* yield 7 to 8 bit accuracy. It must be emphasized that these accuracies were obtained under controlled conditions. All variables were held constant and actual measured values were used in all calculations. It is unlikely that the general situation will yield these accuracies unless adjustments are provided and a calibration procedure is used. This could defeat the low cost objective.



TL/DD/6935-4

A

TL/DD/6935-5

B

TL/DD/6935-6

C

FIGURE 4

3.0 Pulse Width Modulation (Duty Cycle) Technique

3.1 MATHEMATICAL ANALYSIS

The pulse width modulation, or duty cycle, conversion technique is based on the fact that if a repetitive pulse waveform is applied to an RC network, the capacitor will charge to the average voltage of the waveform provided that the RC time constant is sufficiently large relative to the pulse period. See *Figure 5.*

In this technique, the capacitor voltage V_C is compared to the voltage to be measured by means of an analog comparator. The duty cycle is then adjusted to cause V_C to approach the input voltage. The COPS device reads the comparator output and then drives one of its outputs high or low depending on the result, i.e., if V_C is lower than the input voltage, a positive voltage (V1) is applied to charge the capacitor; if V_C is higher than the input voltage, a lower voltage (V0) is applied to discharge the capacitor. Thus the capacitor voltage will seek a point where it varies above and below the input voltage by a small amount. *Figure 6* illustrates the capacitor voltage and the comparator output.

Some mathematical analysis here will be useful to help clarify the technique and to point out its restrictions. Referring to *Figure 6*, we have the following:

$$V_A = V0 + [V_B - V0][e**(-t1/RC)]$$

$$V_B = V_A + [V1 - V_A][1 - e**(-t2/RC)]$$

$$= V1 + [V_A - V1][e**(-t2/RC)]$$

solving for t1 and t2 we have:

$$t1 = -RC\ \ln[(V_A - V0)/(V_B - V0)]$$

$$t2 = -RC\ \ln[(V_B - V1)/(V_A - V1)]$$

let:

$$V_A = V_{IN} - d1$$

$$V_B = V_{IN} - d2$$

substituting the above, the equations for t1 and t2 become:

$$t1 = -RC\ \ln\{[1 - (d1/(V_{IN} - V0))]/[1 + d2/(V_{IN} - V0))]\}$$

$$t2 = -RC\ \ln\{[1 - (d2/(V_{IN} - V1))]/[1 - d1/(V_{IN} - V1))]\}$$

the equations reduce by means of the following assumptions:

1. $d1 = d2 = d$
2. $|V_{IN} - V0| \gg d$
   $|V_{IN} - V1| \gg d$

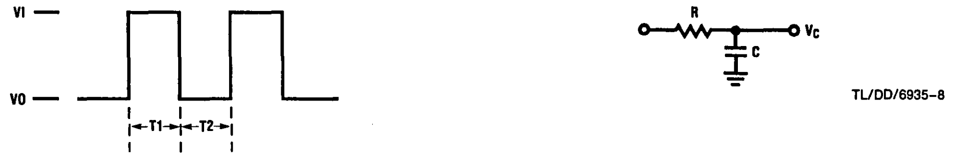
applying these assumptions, we get the following:

$$t1 = -RC\ \ln[(1 + x)/(1 - x)] \text{ where } x = -d/(V_{IN} - V0)$$

$$t2 = -\ RC\ \ln[(1 + x)/(1 - y) \text{ where } y = d/(V_{IN} - V1)$$

because of the assumptions above, the x and y terms in the preceding equations are less than 1, therefore the following expansion can be used:

$$\ln[(1 + z)/(1 - z)] = 2[z + (z**3)/3 + (z**5)/5 + \ldots]$$



$$V_C = \frac{(V1 - V0) \times T1}{T1 + T2}$$

FIGURE 5

Capacitor Voltage

Comparator Output

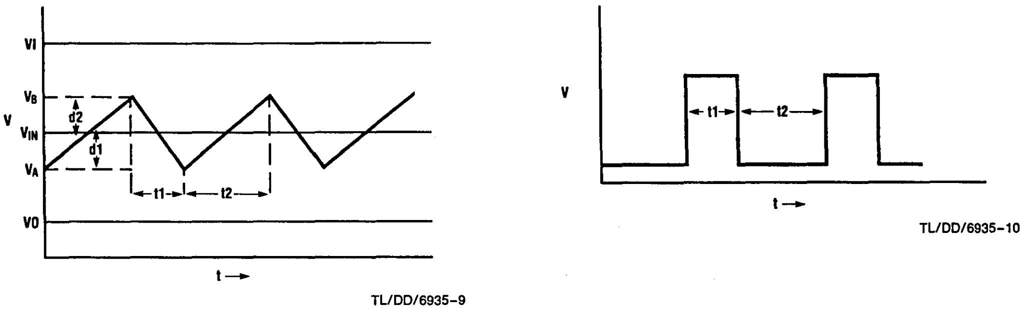


FIGURE 6

substituting we have:

$t1 = -2RC[x + (x**3)/3 + \ldots]$

$t2 = -2RC[y + (y**3)/3 + \ldots]$

under assumption 2 above, the linear term completely swamps the exponential terms yielding the following result (after substituting back into the equation):

$t1 = 2dRC/V_{IN} - V0) \qquad t2 = -2dRC/(V_{IN} - V1)$

therefore:

$$t1/(t1 + t2) = (V1 - V_{IN})/(V1 - V0)$$

$$t2/(t1 + t2) = (V_{IN} - V0)/(V1 - V0)$$

solving for V_{IN}:

$V_{IN} = [t2/(t1 + t2)][V1 - V0] + V0$

or $V_{IN} = V1 - [t1/(t1 + t2)][V1 - V0]$

It follows from the above results that by measuring the times t1 and t2, the input voltage can be accurately determined. As will be seen the restrictions based upon the assumptions above do not cause any serious difficulty.

General Accuracy Considerations

In the preceding calculations it was assumed that the differential output above and below the input voltage was the same. If the comparator output is checked at absolutely regular intervals, and if the intervals are kept as small as possible this assumption can be fairly easily guaranteed—at least to within the comparator offset which is only a few millivolts. As we shall see, this aspect of the technique presents few, if any, difficulties. In addition, there is an RC network at the input of the comparator. The time constant of this network must be long relative to the time between checks of the comparator output. This will insure that the capacitor voltage does not change very much between checks and thereby help to insure that the differences above and below the input voltage are the same.

The next major approximation has to do with the difference between the input voltage and either V1 or V0. We have relied on this difference being much greater than the amount the capacitor voltage changes above and below the input voltage. This approximation allows the nonlinear terms in the logarithmic expansion to be discarded. In practicality, the approximation means that the input voltage must not be "close" to either V1 or V0. Therefore, it becomes necessary to determine how closely the input voltage can approach V1 or V0. It is obvious that the smaller the difference d can be made, the closer the input voltage can approach either reference. The following calculations illustrate the method for determining that difference d. Note, using either V1 or V0 produces the same result. Thus $V = V1 = V0$.

For at least 1% accuracy

$$x + (x**3)/3 < 1.01x$$

$$\text{therefore } x < 0.173$$

$$\text{since } x = d/|(V_{IN} - V)| \text{ we have } d < 0.173|(V_{IN} - V)|.$$

Using the same analysis for 0.1% accuracy in the approximation we get $d < 0.0548|(V_{IN} - V)|$. By applying this relationship, the RC time constant can be adjusted so that, within the time interval, the capacitor voltage does not change by more than d V. The user may then select, within reason, how close to the references he can allow the input voltage to go.

The next consideration is really just one of simplification. It is clear that if V0 is zero, it drops out of the first equation and the relationship is simplified. Therefore, it is desirable to use zero volts as the V0 value. The equation then becomes:

$$V_{IN} = V1t2/(t1 + t2).$$

It is obvious by now that the heart of the technique lies in accurately measuring the times t1 and t2. Clearly this requires that the time base of the COP420 be accurate. Short term variations in the COP420 time base will clearly impact the accuracy of the result. In addition to that there is a serious problem in being able to check the comparator output often enough to get any accuracy and resolution out of simply measuring the times t1 and t2. This problem is circumvented by measuring many periods of the waveform. Doing this gives a large average, which improves the accuracy and tends to eliminate any spurious changes. Of course, the trade off is increased time to do the conversion. However if the time is available, the technique becomes restricted only by the accuracy of the external components. Those of the comparator and the reference voltage are most critical.

It is clear from the equation above that the accuracy of the result is directly dependent upon the accuracy of the reference voltage V1. In other words, it is not possible to be more accurate than the reference voltage. If, however, all that is required is a ratio between the input voltage and the reference voltage, the accuracy of the reference will not be a controlling factor provided that the input voltage tracks the reference. This requires that the input voltage be generated from the reference voltage in some form, e.g., a voltage divider with V_{IN} coming off a variable resistance.

Finally, we have noted that the difference d must be small. If the capacitor had to charge or discharge a long way toward V_{IN}, the nonlinearity of the capacitor charge curve would be significant. This therefore requires that the conversion begin with the capacitor voltage close to the input voltage.

Note that the RC value is not part of the equation. Therefore the accuracy of the time constant has no effect on the result as long as the time constant is long relative to the time between checks of the comparator output.

The final point is that the reference voltages, whatever they may be, must be hard sources. Should these voltages vary or drift at all, they will directly affect the result. In those configurations where the references are being switched in and out, the voltage should not change when it is switched into the circuit.

3.2 BASIC IMPLEMENTATION

General

The objective, then, is to measure the times t1 and t2. This is accomplished in the software by means of two counters. One of the two counters counts the t2 time; the other counter counts the total time t1 + t2.

It is necessary to check the comparator output at regular intervals. Thus the software must insure that path lengths through the test and increment loops are equal in time. Further it is desirable to keep the time required to increment the counters as short as possible. A trade off usually comes into play here. The shortest loop in terms of code required to implement the function is rarely the shortest loop in terms of time required to execute the function. The user has to decide which implementation is best for him. The choice will frequently be governed by factors other than the A/D conversion limits.

It must be remembered that we are now dealing with analog signals. If significant accuracy is required, we are handling very small analog signals. This requires the user to take precautions that are normally required when working with linear circuits, e.g., power supply decoupling and bypassing, lead length restrictions, crosstalk, op amp and comparator stabilization and compensation, desired and undesired feedback, etc. As greater accuracy is sought these factors are more and more significant. It is suggested that the reader refer to the National Semiconductor Linear Applications Handbook and to the data sheets for the various components involved to see what specific precautions should be taken both in general and for a specific device.

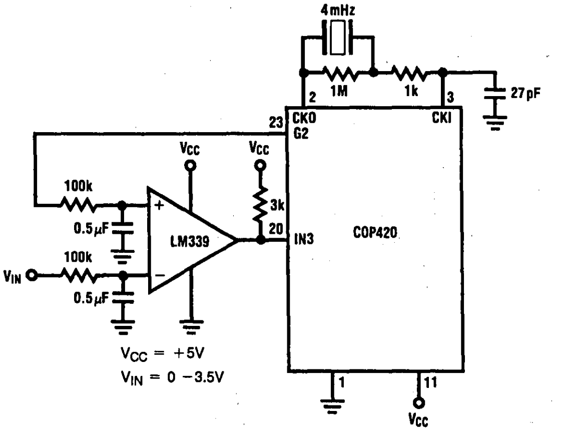
The Base Circuit

Figure 7 shows the diagram for the basic circuit required to implement the duty cycle conversion scheme. The flow chart and code required to implement the function are shown in *Figure 8*. Note that the flow chart and code do not change—except for possible polarity change on output to allow for an inverting buffer—for any of the improvements in accuracy discussed later. The only exception to this is the technique illustrated in *Figure 10* and the variations there are minor.

The code and flow chart in *Figure 8* implement the technique as described above. The large averaging technique is used as it would be too difficult to measure the times t1 and t2 in a single period. The total time, t1 + t2, is the viewing window under complete control of the software. This window is a time equal to the total number of counts, determined by desired accuracy, multiplied by the loop time for a single count. A second counter is counting the t2 time. Special care is taken to insure that all paths through the code take the same length of time since the integrity of the time count is the essence of the technique. The full conversion scheme would use the subroutine in *Figure 8*. Normally the subroutine would be called first just to get the capacitor charged close to the input voltage. The result obtained here would be discarded. Then the routine would be called a second time and the result used as required.

In the configuration in *Figure 7*, there is an RC network in both input legs of the comparator. This is to balance the inputs of the device. For this reason, R1 = R2. C1 is the capacitor whose voltage is being varied by the pulse waveform. C2 is in the circuit only for stabilization and symmetry and is not significant in the result. The comparator tends to oscillate when the + and − inputs are nearly equal without capacitor C2 in the circuit.

As would be expected, the basic circuit has some difficulties. By far the most serious of these difficulties is the output level of the G line. To be sure of the high and low level of this output the levels should be measured. The "1" level will be between the spec minimum of 2.4V and V_{CC} (here assumed to be 5V). The "0" level will be between the 0.4V spec maximum and ground. With light loads, these levels are likely to vary from device to device. Furthermore, we have the same "1" level problem that was mentioned in the simplest technique: the capacitive load is large and the capacitor is charging while the output is trying to go to the high level.

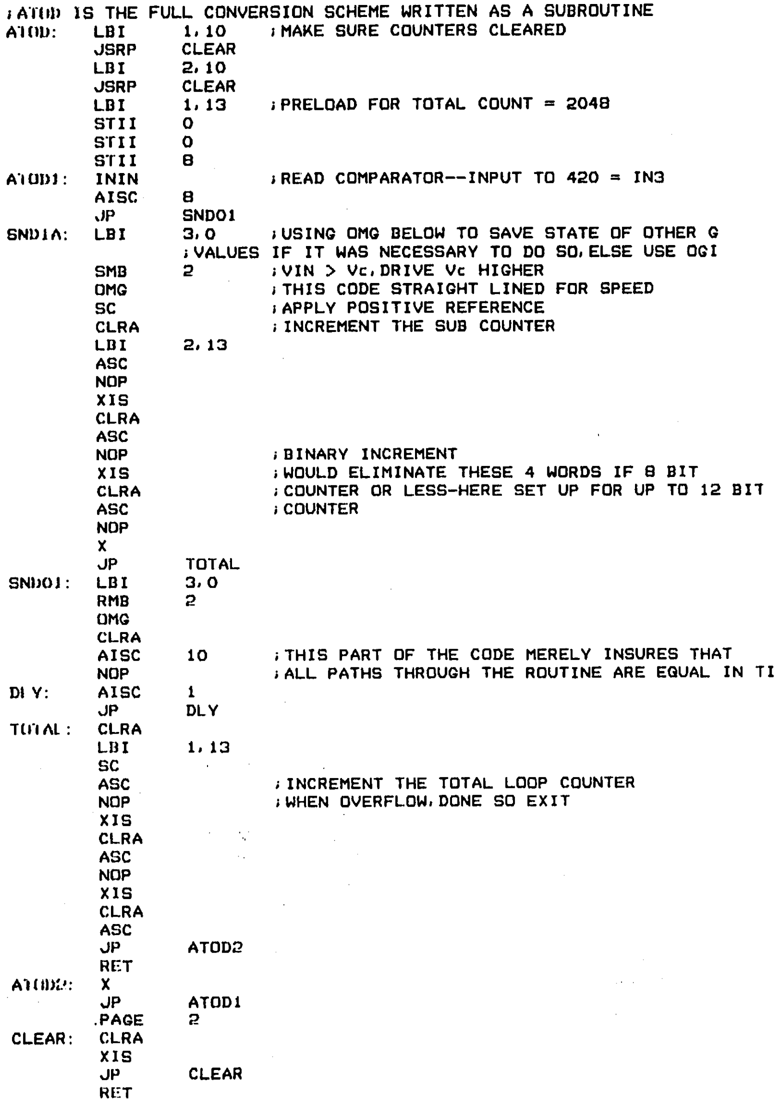


TL/DD/6935-11

FIGURE 7. Basic Duty Cycle A/D

There is also a problem with the low level. When the output goes low, the capacitor begins to discharge through the output device of the COP420. This discharge current has the effect of raising the "0" level and thereby introducing error. Note that we are not talking about large changes in the voltages, especially the low level. Typically, the change will only be a few millivolts but that can translate into a loss of accuracy of several bits.

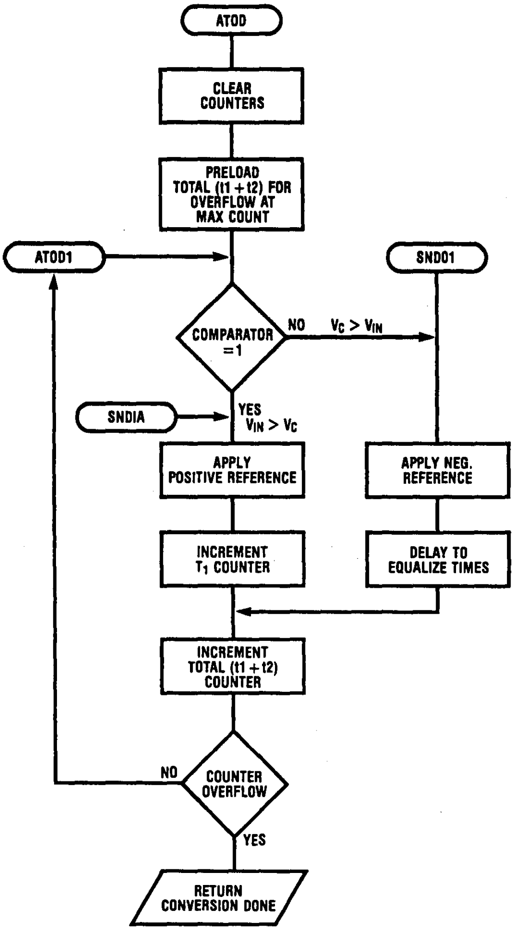
Under laboratory conditions—holding all variables constant and using precise measured values in the calculations—the circuit of *Figure 7* yielded 5 bit ± 1 bit accuracy over the range of V0 (here measured to be 0.028V) to 3.5V (the maximum specified input voltage for the comparator with V_S = 5V). Increasing the number of total counts had very little effect on the result. In the general case, the basic scheme should not be relied upon for more than 4 bits of accuracy, especially if one assumes that V1 = V_{CC} and V0 = 0. As shall be seen, it is not difficult to improve this accuracy considerably.



```
;ATOD IS THE FULL CONVERSION SCHEME WRITTEN AS A SUBROUTINE
ATOD:   LBI     1,10    ;MAKE SURE COUNTERS CLEARED
        JSRP    CLEAR
        LBI     2,10
        JSRP    CLEAR
        LBI     1,13    ;PRELOAD FOR TOTAL COUNT = 2048
        STII    0
        STII    0
        STII    8
ATOD1:  ININ            ;READ COMPARATOR--INPUT TO 420 = IN3
        AISC    8
        JP      SND01
SND1A:  LBI     3,0     ;USING OMG BELOW TO SAVE STATE OF OTHER G
                ;VALUES IF IT WAS NECESSARY TO DO SO,ELSE USE OGI
        SMB     2       ;VIN > Vc,DRIVE Vc HIGHER
        OMG             ;THIS CODE STRAIGHT LINED FOR SPEED
        SC              ;APPLY POSITIVE REFERENCE
        CLRA            ;INCREMENT THE SUB COUNTER
        LBI     2,13
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP             ;BINARY INCREMENT
        XIS             ;WOULD ELIMINATE THESE 4 WORDS IF 8 BIT
        CLRA            ;COUNTER OR LESS-HERE SET UP FOR UP TO 12 BIT
        ASC             ;COUNTER
        NOP
        X
        JP      TOTAL
SND01:  LBI     3,0
        RMB     2
        OMG
        CLRA
        AISC    10      ;THIS PART OF THE CODE MERELY INSURES THAT
        NOP             ;ALL PATHS THROUGH THE ROUTINE ARE EQUAL IN TI
DLY:    AISC    1
        JP      DLY
TOTAL:  CLRA
        LBI     1,13
        SC
        ASC             ;INCREMENT THE TOTAL LOOP COUNTER
        NOP             ;WHEN OVERFLOW,DONE SO EXIT
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        JP      ATOD2
        RET
ATOD2:  X
        JP      ATOD1
        .PAGE   2
CLEAR:  CLRA
        XIS
        JP      CLEAR
        RET
```

TL/DD/6935-45

FIGURE 8A. Duty Cycle A/D Code



TL/DD/6935-12

FIGURE 8B. Duty Cycle A/D Flow Chart

3.3 ACCURACY IMPROVEMENTS

General Improvements

Figure 9 illustrates circuit changes that will make significant improvements in the accuracy of the technique. In *Figure 9A* a CMOS buffer is used to drive the RC network. The output of the COP420 drives the CMOS gate, which here is a 74C04 because of its output characteristics. The main thing that this technique does is to reduce the difficulties with the output levels. Typically, V0 is 0V and V1 is V_{CC}. We also have a "harder" source for the voltages — the levels don't change while the capacitor is charging or discharging. Now, even more clearly than before, the accuracy of V_{CC} is the controlling voltage tolerance. The accuracy of the result will be no better than the accuracy of V_{CC} (for a system requiring absolute accuracy).

Under laboratory conditions, the circuit of *Figure 9A* yielded the accuracies as indicated below for various total counts. The accuracy increased with the total count until the count exceeded 2048. There was no significant increase in accuracy with this circuit for counts in excess of 2048. (Remember that these results were obtained under controlled conditions). We may then view the results obtained with 2048 counts as the upper limit of accuracy with the circuits of *Figure 9A*. The results were as follows:

Total Count	Resultant Accuracy
512	8 ± 1/2 bits
1024	9 ± 1bits
2048	9 ± 1/2 bits
4096	9 ± 1/2 bits

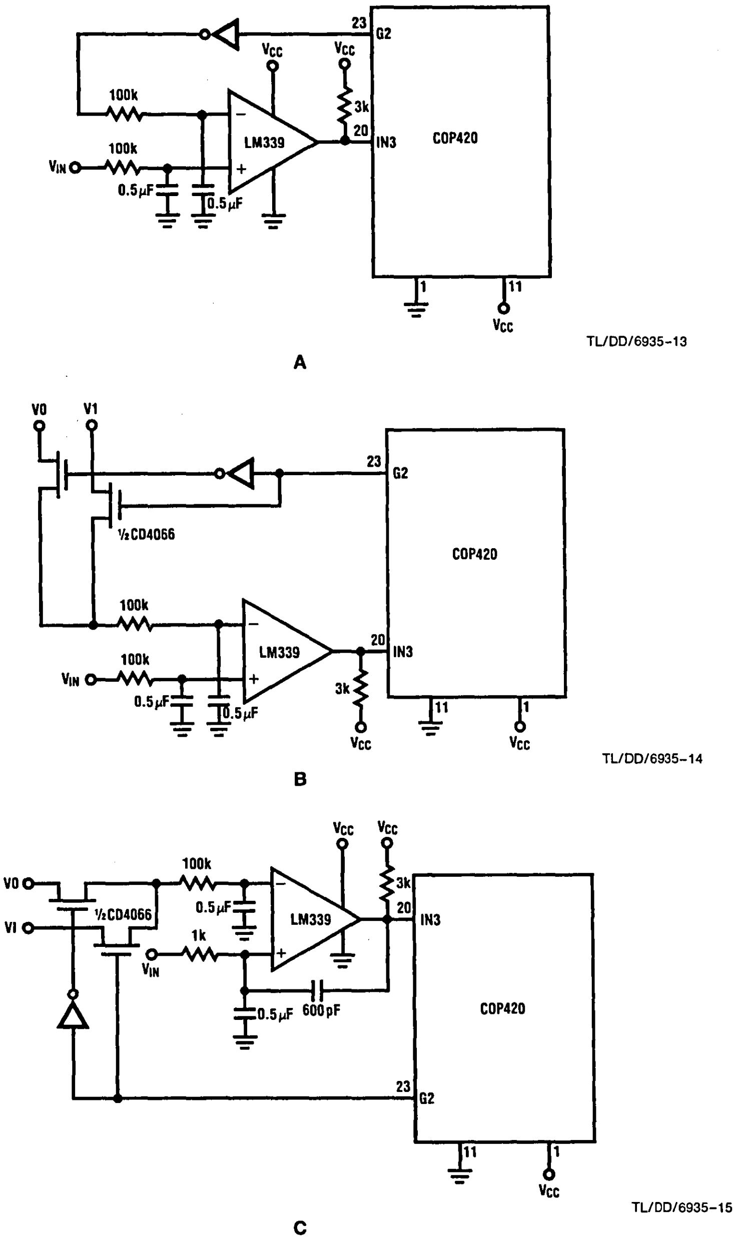


FIGURE 9. Improvements to Duty Cycle A/D

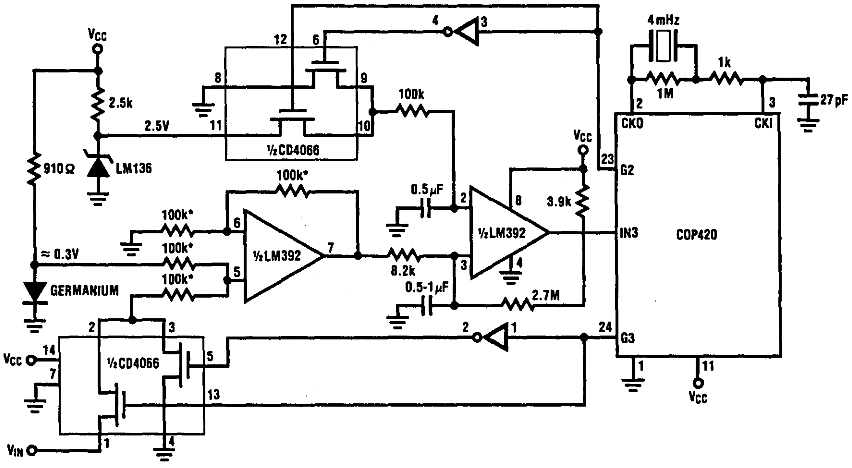
The circuit of *Figure 9B* makes a significant change to improve accuracy. Now the COP420 is controlling analog switches and switching in positive and negative references. Therefore the accuracy of the reference voltages is the controlling factor. Generally this will improve the accuracy over that obtained with *Figure 9A*. With the circuit of *Figure 9B*, with V0 = 1V (negative reference), and V1 = 3V (positive reference), 9 bit accuracy was achieved with a total count of 1024. V0 and V1 were arbitrarily chosen to place the input voltage approximately in the center of the allowable comparator input range with V_S = 5V. Remember, the accuracy of the references is controlling. The result can be no more accurate than the references. Furthermore, these references must be hard sources; i.e., they must not change when they are switched into the circuit as that contributes error into the result.

In *Figure 9C*, capacitive feedback was added to the comparator circuit and the series resistance to V_{IN} was decreased. The feedback added hysteresis and forced the comparator to slew at its maximum rate (significant errors are introduced if the comparator does not change state in a time shorter than the cycle time of the controller). Both of these changes resulted in increased accuracy of the result. With V0 = 0, V1 = 5V (V_{CC}) and V_{CC} held steady at 5.000V, an accuracy of 10 bits ± 1 bit was achieved over the input range of 0 to 3.5V.

It is obviously possible to use any combination of the configurations in *Figure 9* for a given application. What is used will depend on the user and his specific requirements.

Figure 10 illustrates a further refinement of the basic approach. This configuration can be used if greater accuracies are needed. The major change is the addition of a summing amplifier to the circuit for the purpose of adding a fixed offset voltage to the input voltage. This has the effect of moving the input voltage away from the negative reference (which is 0V here). This offset voltage should be stable as the changes in it will directly affect the result. The offset voltage should be chosen so as to place the effective input voltage (the voltage at the comparator input) approximately in the center of the range between the two references. The precise value of the offset is not critical nor is its source. The forward voltage drop across a germanium diode is used as the offset in *Figure 10*, but this offset can be generated in any convenient manner. The forward voltage drop of the germanium diode is approximately 0.3V. Given this and the negative reference of 0V and a positive reference of 2.5V, the input voltage is restricted to a range of 0 to 2V. Therefore, the effective input voltage (at the comparator input) is approximately 0.3V to 2.3V — well within the limits of the two references. The circuit also includes provision for an autozero self calibration procedure.

Note that the resistors in the summing amplifier should be matched. The absolute accuracy of these resistors is not significant, but their accuracy relative to one another can have a significant bearing on the result. The restriction is imposed so that the output of the summing amplifier is exactly the sum of the input voltage and the offset voltage. This requires unity gain through the amplifier and that the



TL/DD/6935-16

*Resistors should be matched

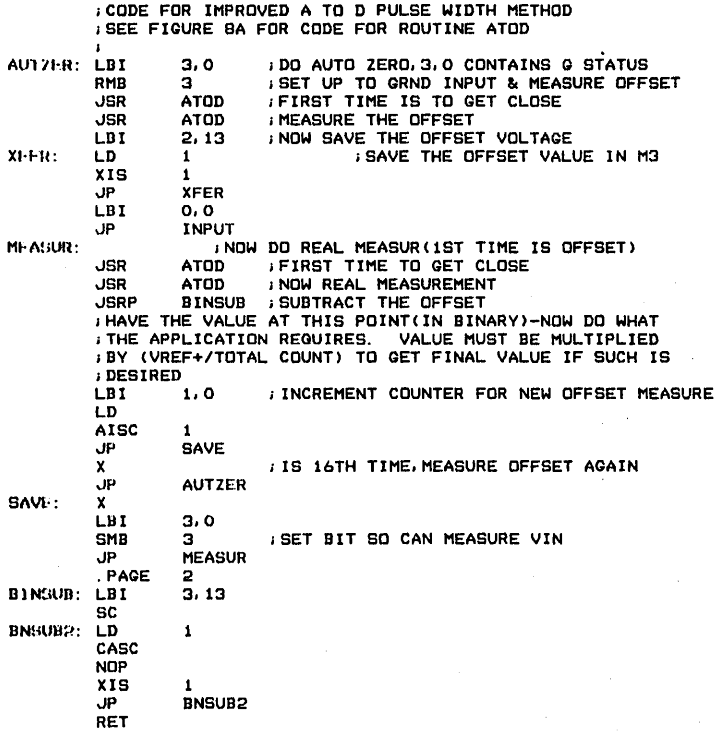
V_{CC} = +5V

0 ≤ V_{IN} ≤ 2V

FIGURE 10. Improved Duty Cycle A/D with Autozero

impedance in each summing leg be the same. These effects can become very serious if one is trying for significant accuracy—e.g., if 12 bit accuracy is being sought 1% matching of those resistors can introduce an error of 1% maximum. While 1% accurate is fairly good, it is significantly less than 12 bit accuracy. Related to this effect is a possible problem with the source impedance of the input voltage. If that impedance is significant in terms of its ratio to the summing resistor, errors are introduced just as if the resistors are mismatched. "Significant" is determined in terms of the desired system accuracy and the relative impedance values. The comparator section is using some feedback to provide hysteresis for stability and a low series resistance is used for the input to the comparator.

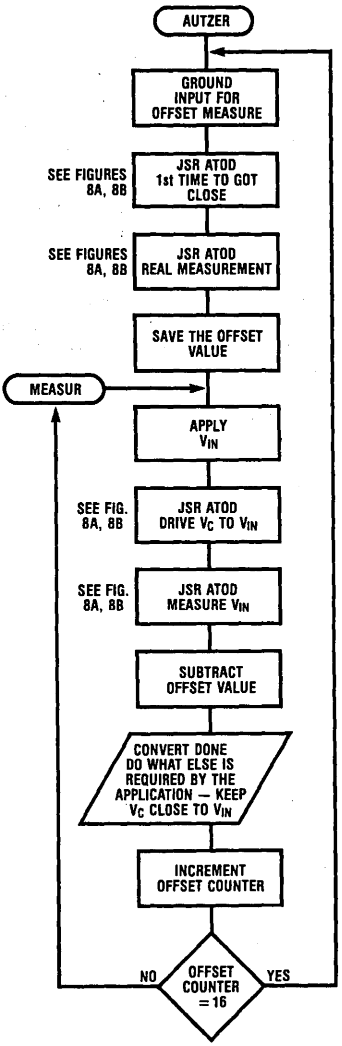
Most significantly, this configuration allows a true zeroing of the system. Through the additional analog switches shown, the COP420 can easily perform an autozero function by tying the input to ground and measuring the result. Thus the system offsets can be calculated, stored and subtracted from the result. This improves the accuracy and is also more forgiving on the choice of the comparator and op amp selected. Furthermore, the offset can be periodically recomputed by the COP420 thereby compensating for drift in system offsets. Nonetheless, the accuracy of the reference is the controlling factor. It is NOT possible to obtain an absolute (as opposed to ratiometric) accuracy of 12 bits without a reference that is accurate to 12 bits. The LM136 used in *Figure 10* is a 1% reference. Although not inherently accurate to 12 bits, the voltage of the LM136 may be trimmed to exact value by means of a variable resistor. The data sheet of the LM136 illustrates this connection. Under laboratory conditions, the circuit of *Figure 1* yielded 11 bit ±1 bit accuracy with a total count of 4096 over the input range of 0 to 2V. *Figure 11* indicates the flow chart and the code required to implement the technique of *Figure 10.*



```
         ;CODE FOR IMPROVED A TO D PULSE WIDTH METHOD
         ;SEE FIGURE 8A FOR CODE FOR ROUTINE ATOD
         ;
AUTZER:  LBI     3,0     ;DO AUTO ZERO,3,0 CONTAINS Q STATUS
         RMB     3       ;SET UP TO GRND INPUT & MEASURE OFFSET
         JSR     ATOD    ;FIRST TIME IS TO GET CLOSE
         JSR     ATOD    ;MEASURE THE OFFSET
         LBI     2,13    ;NOW SAVE THE OFFSET VOLTAGE
XFER:    LD      1               ;SAVE THE OFFSET VALUE IN M3
         XIS     1
         JP      XFER
         LBI     0,0
         JP      INPUT
MEASUR:               ;NOW DO REAL MEASUR(1ST TIME IS OFFSET)
         JSR     ATOD    ;FIRST TIME TO GET CLOSE
         JSR     ATOD    ;NOW REAL MEASUREMENT
         JSRP    BINSUB  ;SUBTRACT THE OFFSET
         ;HAVE THE VALUE AT THIS POINT(IN BINARY)-NOW DO WHAT
         ;THE APPLICATION REQUIRES.  VALUE MUST BE MULTIPLIED
         ;BY (VREF+/TOTAL COUNT) TO GET FINAL VALUE IF SUCH IS
         ;DESIRED
         LBI     1,0     ;INCREMENT COUNTER FOR NEW OFFSET MEASURE
         LD
         AISC    1
         JP      SAVE
         X               ;IS 16TH TIME,MEASURE OFFSET AGAIN
         JP      AUTZER
SAVE:    X
         LBI     3,0
         SMB     3       ;SET BIT SO CAN MEASURE VIN
         JP      MEASUR
         .PAGE   2
BINSUB:  LBI     3,13
         SC
BNSUB2:  LD      1
         CASC
         NOP
         XIS     1
         JP      BNSUB2
         RET
```

TL/DD/6935-46

FIGURE 11A. Duty Cycle A to D, Improved Method


AUTZER
GROUND INPUT FOR OFFSET MEASURE
SEE FIGURES 8A, 8B
JSR ATOD 1st TIME TO GOT CLOSE
SEE FIGURES 8A, 8B
JSR ATOD REAL MEASUREMENT
SAVE THE OFFSET VALUE
MEASUR
APPLY V_{IN}
SEE FIG. 8A, 8B
JSR ATOD DRIVE V_C TO V_{IN}
SEE FIG. 8A, 8B
JSR ATOD MEASURE V_{IN}
SUBTRACT OFFSET VALUE
CONVERT DONE DO WHAT ELSE IS REQUIRED BY THE APPLICATION — KEEP V_C CLOSE TO V_{IN}
INCREMENT OFFSET COUNTER
NO
OFFSET COUNTER =16
YES

TL/DD/6935-17

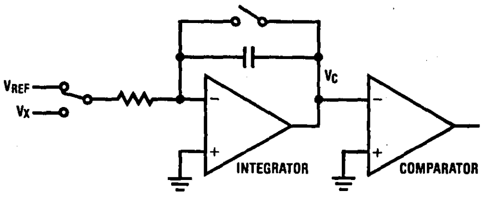
FIGURE 11B. Flow Chart for Improved Duty Cycle A/D

4.0 Dual Slope Integration Techniques

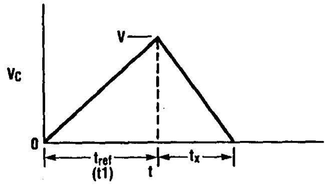
4.1 Mathematical Background

(Some of this background information is taken from National Semiconductor Linear Applications Note AN-155. The reader is referred to that document for other related general information.)

The basic approach of dual slope integration conversion techniques is to integrate a voltage across a capacitor for a fixed time, and then to integrate in the other direction with a known voltage until the starting point is reached. The ratio of the two times then represents the unknown voltage. Some of the math below in conjunction with *Figure 12* will illustrate the approach.



TL/DD/6935-18



TL/DD/6935-19

FIGURE 12. Dual Slope Integration—Basic Concept

$$I_X = C\frac{dV}{dt} = V_X/R$$

$$V_X = RC\frac{dv}{dt}$$

$$\int_0^{T1} V_X dt = \int_0^{V} RCdV$$

$$V_X T1 = RCV$$

$$V = V_X T1/RC = I_X T1/C$$

Similarly:

$$I_{REF} = C\frac{dV}{dt} = V_{REF}/R$$

$$V_{REF} = RC\frac{dV}{dt}$$

$$\int_{T1}^{T1 + T_X} V_{REF}dt = \int_{V}^{0} RCdV$$

$$V_{REF}T_X = -RCV$$

$$V = -V_{REF}T_X/RC$$

$$-V_{REF}T_X/RC = V_X T1/RC$$

$$V_X = -V_{REF}T_X/T1$$

Two important facts arise from the preceding mathematics. First of all, there is a linear relationship involved in determining the unknown voltage. Secondly, the negative sign in the final equation indicates that the reference and the unknown, relative to some point (which may be 0V or some bias voltage), have opposite polarity. Thus, if it is desired to measure 0 to +5V, the reference voltage must be −5V. If the input is restricted to 2.5 to 5V, the reference can be 0V as the integrator and comparator are biased at +2.5V (then the 0V is in fact −2.5V relative to the biasing voltage, and the input range is 0 to 2.5V relative to the same bias voltage).

There are some difficulties with dual polarity conversion using the dual slope method. It is clear from the math above that if the input voltage will be dual polarity, it is necessary to have two references—one of each polarity. The midrange biasing arrangement briefly described above eliminates the need for two different polarities but does not help very much since two references are still required—one at the positive value and one at the bias value. Ground is the other reference. Further, the need to select one of two references further complicates the circuitry involved to implement the approach. Also, the dual requirement brings up a difficulty with the bias currents of the integrator and comparator. They could add to the slope in one polarity and subtract in the other.

The only real operational difficulty in dual slope systems is establishing the initial conditions on the integrating capacitor. If this capacitor is not at the proper initial conditions, accuracy will be severely impaired. *Figure 12* indicates a switch across the capacitor as a means of initializing it. In a software driven system, the initialization can be accomplished by doing two successive conversions. The result of the first conversion is discarded. It is performed only to initialize the capacitor. The second conversion produces the valid result. One need only insure that there is not significant time lapse between the two conversions. They should take place immediately after one another.

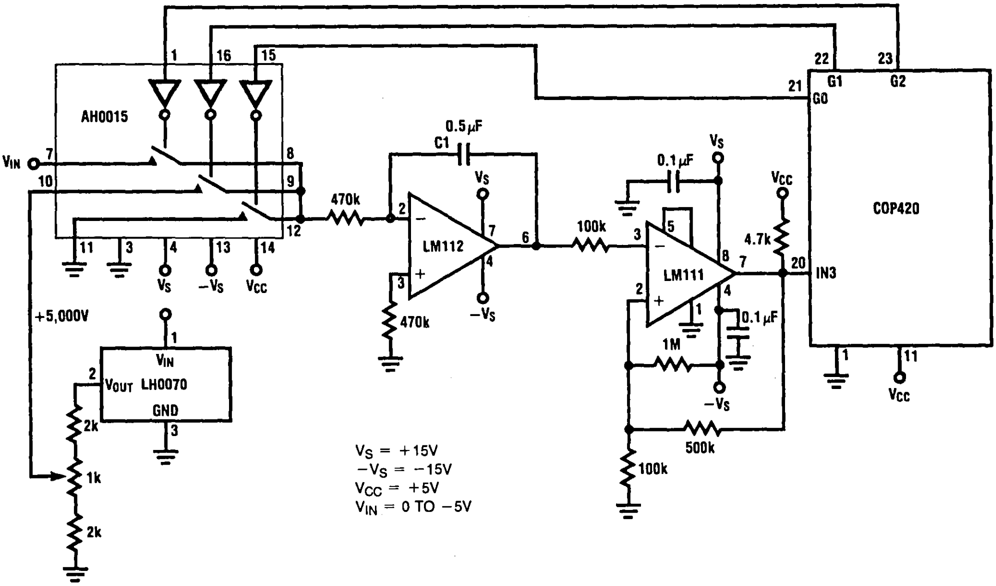
This approach obviously lengthens conversion time but it eliminates many problems. The alternative to this approach of two successive conversions is to take a great deal of care in insuring the initial state of the integrating capacitor and in selecting op amps and comparators with low offsets.

4.2 THE BASIC DUAL SLOPE TECHNIQUE

Figure 13 indicates an implementation of the basic dual slope technique. This is a single polarity system and thus requires only the single reference voltage. The circuit of *Figure 13* is perhaps not the cheapest way to implement such a scheme but it is representative and illustrates the factors that must be considered.

Consider first the means of initializing the integrating capacitor C1. The routine here connects the input to ground and does a conversion on zero volts as a means of initialization. Subsequently—and this is typical of the more usual technique—two conversions are performed. The first conversion is to initialize the capacitor. The second conversion yields the result. Some form of initialization or calibration procedure is required to achieve optimum accuracy from dual slope conversion schemes.

The comparator in this circuit is used in the inverting mode and has positive feedback as recommended in the LM111 data sheet. The voltage reference is the LH0070, which is a 0.01% reference. A resistive voltage divider on the IH0070 creates the 5V value. The use of the voltage divider brings up two difficulties (which can be overcome if the LH0070 is used at its full value, thus eliminating the divider, and the result properly scaled in the microcontroller or series integrating resistor increased). First, the impedance of the reference must be small relative to the series resistance used in the integrator. If this were not the case, the slopes would



V_S = +15V
$-V_S$ = −15V
V_{CC} = +5V
V_{IN} = 0 TO −5V

TL/DD/6935-20

FIGURE 13. Basic Dual Slope Integration A/D Scheme

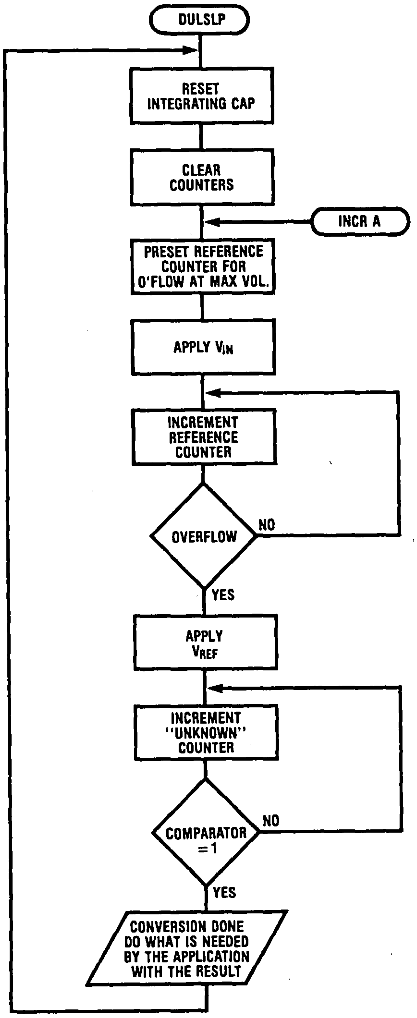
show an effect due to the difference in the R value between the applied reference voltage and the unknown input. (By the same token, the output impedance of the source supplying the unknown must also be small relative to that series integrating resistor). Secondly, the bias currents of the integrator may be such as to affect the reference voltage when it is coming from a simple resistor divider. Both problems are reduced if small resistor values are used in the divider. Note also that current mode switching would reduce the problem as well. It should be pointed out that the errors introduced by these problems are not gross deviations from the expected value. They are small errors that will not make much difference in the majority of applications. They are, however the kind of errors that can make the difference between a system accurate to 10 bits and one accurate to 12 bits (assuming all other factors are the same).

Figure 14 shows the flow chart and code required to implement the basic dual slope technique as shown in *Figure 13*. Under laboratory conditions an accuracy of 12 bits ±1 bit was achieved. The method is slow, with the maximum conversion time equal to 2 x T_{REF}. Notice that the accuracy of V_{CC} and that of the integrating resistor and capacitor are not involved in the accuracy of the result. The accuracy of V_{REF} is, of course, controlling if absolute accuracy—rather than ratiometric accuracy—is desired. The absolute accuracy of the circuit can be no better than the accuracy of the reference. If ratiometric accuracy is all that is required, there is no particular problem. The accuracy is merely relative to the reference. The R and C values do not impact the accuracy because the integration in both directions is being done through the same R and C. Results would be quite different if a different value of R or C was used for one of the slopes.

```
DULSLP:  OGI     1        ;HOLD THE INPUT TO GROUND TO RESET THE
         LBI     2,11     ;INTEGRATING CAPACITOR
         JSRP    CLEAR    ;CLEAR THE COUNTER
         JSR     INCRA    ;TO GET US CLOSE,NEXT READING IS REAL
CLEAR2:  LBI     2,11     ;NOW CLEAR THE COUNTER
         JSRP    CLEAR    ;MAKE SURE COUNTER CLEARED TO ZERO
   ;1,15 = 0 AND START AT 1,13 FOR COUNT = 4096
   ;1,15 = 14 AND START AT 1,12 FOR COUNT = 8192
   ;1,15 = 12 AND START AT 1,12 FOR COUNT = 16384
   ;FOLLOW SAME PATTERN FOR OTHER COUNTS
   ;
MEASUR:  JSR      INCRA    ;RUN THRU THE INCREMENTS
         ; NOW HAVE THE BINARY VALUE, USE IT AS IS OR
         ; MULTIPLY BY (Vref/TOTAL COUNT) TO CREATE THE VOLTAGE
         ; RESULT--THEN CONTINUE WITH THE OPERATION
         LBI      2,11
         JSRP     CLEAR    ;CLEAR THE COUNTER
         JSR      INCRA    ;TO GET CAP CLOSE TO 0 AGAIN
         JP       CLEAR2
 ;FOLLOWING SUBROUTINE INCRA IS THE REAL PART OF THE ROUTINE
 ;CONCERNED WITH THE COUNTING FOR THE CONVERSION.
INCRA:   LBI      1,15     ;R1 IS CLEARED PRIOR TO START
         STII     15       ;PRESET THE COUNTER FOR 4096
         OGI      4        ;APPLY VIN
INCR:    LBI      1,12
         SC
BINAD1:  CLRA
         ASC
         NOP
         XIS
         JP       BINAD1
         NOP               ;2 NOPS TO EQUALIZE TIMES
         NOP
         SKC
         JP       INCR
         OGI      2        ;DONE,NOW APPLY VREF
INCR2:   LBI      2,12     ;COUNT UNTIL COMPARATOR CHANGES
         SC
BINAD2:  CLRA
         ASC
         NOP
         XIS
         JP       BINAD2   ;STRAIGHT LINE THE ADD FOR SPEED
         ININ              ;SAVE WORDS BY USING G
         AISC     8        ;SEE IF IN3=1
         JP       INCR2    ;IN3 IS 0,KEEP COUNTING
OUTPUT:  OGI      1        ;KEEP INPUT AT 0
         RET
```

TL/DD/6935-47

FIGURE 14A. Dual Slope A/D Code


DULSLP
RESET INTEGRATING CAP
CLEAR COUNTERS
INCR A
PRESET REFERENCE COUNTER FOR O'FLOW AT MAX VOL.
APPLY V_{IN}
INCREMENT REFERENCE COUNTER
OVERFLOW
NO
YES
APPLY V_{REF}
INCREMENT "UNKNOWN" COUNTER
COMPARATOR =1
NO
YES
CONVERSION DONE DO WHAT IS NEEDED BY THE APPLICATION WITH THE RESULT

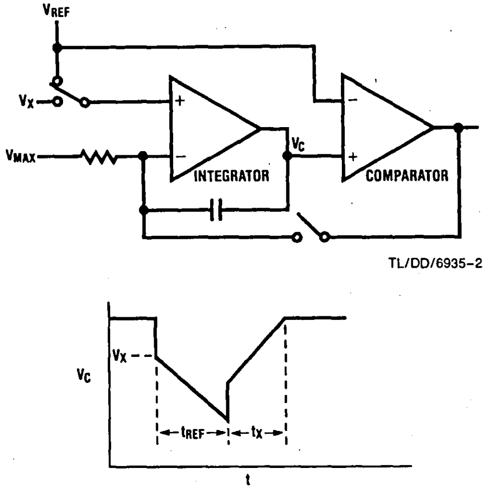
TL/DD/6935-21

FIGURE 14B. Basic Dual Slope A/D Flow Chart

4.3 MODIFIED DUAL SLOPE TECHNIQUE

General

The basic idea of the modified dual slope technique is the same as that of the basic approach. The modified approach eliminates the need for dual polarity references and is also more forgiving in the selection of the op amp and comparator required. *Figure 15* illustrates the basic idea.



TL/DD/6935-22

TL/DD/6935-23

FIGURE 15. Modified Dual Slope — Basic Concept

The math analysis is much the same:

$$I_X = C\frac{dV}{dt} = (V_X - V_{MAX})/R$$

$$V_X - V_{MAX} = RC\frac{dV}{dt}$$

$$(V_X - V_{MAX})T1 = RC$$

$$V = (V_X - V_{MAX})T1/RC$$

Similarly:

$$I_{REF} = C\frac{dV}{dt} = (V_{REF} - V_{MAX})/R$$

$$(V_{REF} - V_{MAX})T_X = -VRC$$

$$V = -(V_{REF} - V_{MAX})T_X/RC$$

$$(V_{MAX} - V_{REF})T_X = (V_X - V_{MAX})T1$$

$$V_X = V_{MAX} + (V_{MAX} - V_{REF})T_X/T1$$

The main difference between this and the basic approach is the offset voltage V_{MAX}. The main restriction is that all input voltage values (V_X) are less than V_{MAX}. It is also apparent that the total count is proportional to the difference between V_{MAX} and V_X. The only significant effect of this is, however, to slightly complicate the arithmetic required to arrive at a value for V_X.

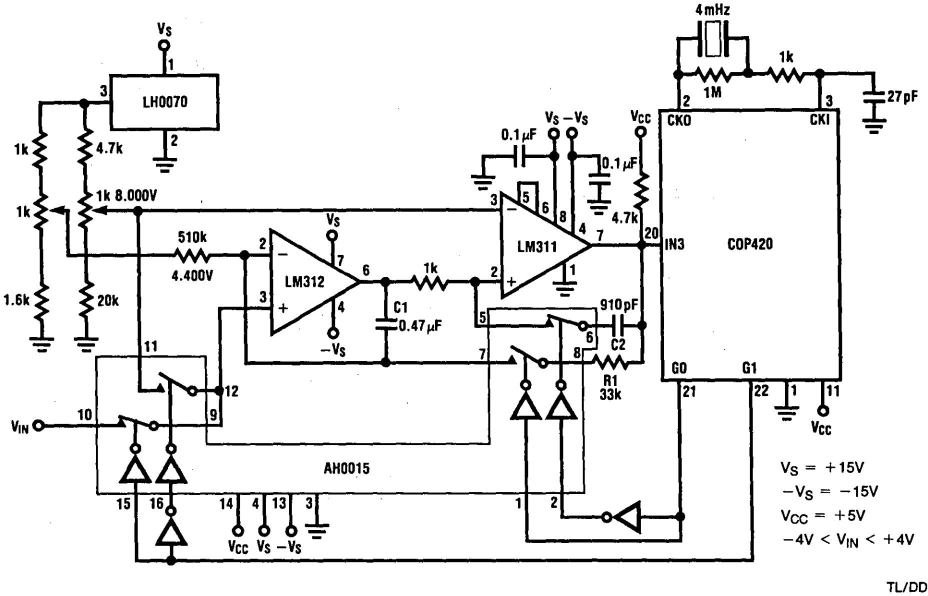
Given that the input voltage V_X is always less than V_{MAX}, the modified dual slope technique is automatic polarity. This fact comes straight out of the equation above. Thus dual polarity references are not required. However, two precise voltages are required: V_{MAX} and V_{REF}. However, the V_{MAX} value can be used for a zero adjust as indicated in *Figure 16*. This means that the V_{MAX} value need not be so precise as it will be adjusted in a calibration procedure to produce a zero output. This adjustment amounts to a compensation for the bias currents and offsets. Thus the COP420 can use the supposed value of V_{MAX} with V_{MAX} later being "tweaked" to give the proper result at zero input. In addition, the initialization loop for the integrating capacitor includes the comparator. Thus the intial condition on the capacitor becomes not zero but the sum of the offset voltages of the comparator and op amp. Thus the choice of these components is not critical in a modified dual slope approach.

An Example of the Modified Dual Slope Approach

Figure 16 illustrates an implementation of the modified dual slope technique. The system is calibrated by holding V_{IN} to ground and then adjusting V_{MAX} for a "0" result. Capacitor C1 is the integrating capacitor. Capacitor C2 is used only to cause a rapid transition on the comparator output. C2 is especially useful if an op amp is being used as the comparator stage. Resistor R1 is just part of the capacitor initializing loop. An LH0070 is being used to generate the reference voltage and the V_{MAX} value. The discussion previously about these being hard sources is equally relevant here. In fact, this problem was much more significant in this particular implementation and made the difference between a 10 and 12 bit system. As shown, the technique was accurate to 10 bits. Another bit was obtained when the V_{MAX} and V_{REF} values were buffered. It must be remembered that when trying to achieve accuracies of this magnitude board layout, parts placement, lead length, etc. become significant factors that must be specifically addressed by the user.

There are some other considerations in using this technique. The amount of time required to count the specified number of counts starts to become a significant factor. If it takes "too long" to do the counting, the capacitor can charge to either supply voltage depending on which direction it is integrating. This causes the wave shape shown in *Figure 15* to flatten out. This effectively limits the input range for all accuracy is lost once that waveform flattens out. In fact, this was the limiting factor on the accuracy in *Figure 16* as shown. Given the amount of time required for an increment of the counter for T_{REF} (or T_X), it was not possible to reach the 4096 counts required for 12 bit accuracy before the waveform flattened out. Decreasing the total count solves the problem at the expense of accuracy. It is therefore desirable to keep the loop time required for an increment as fast as possible. The code to implement *Figure 16* is shown in *Figure 17* and reflects that concern. The other way to solve the problem is to use a large value for R and C. This is the easiest solution and preserves accuracy. Its cost is increased conversion time.

Both the basic and modified dual slope schemes can be very accurate and are commonly used. They tend to be relatively slow. In many applications, however, speed is not a factor and these approaches can serve very well. There are various approaches to dual slope analog to digital conversion which try to improve speed and/or accuracy. These are usually multiple ramping schemes of one form or another. The heart of the approach is the basic scheme described above. It is not the purpose here to delve into all the possible ways that dual slope conversion may be accomplished. The control software is not significantly different regardless of which particular variation is used. The basic ramping control is the same as that indicated here.



TL/DD/6935-24

FIGURE 16. Modified Dual Slope Integration

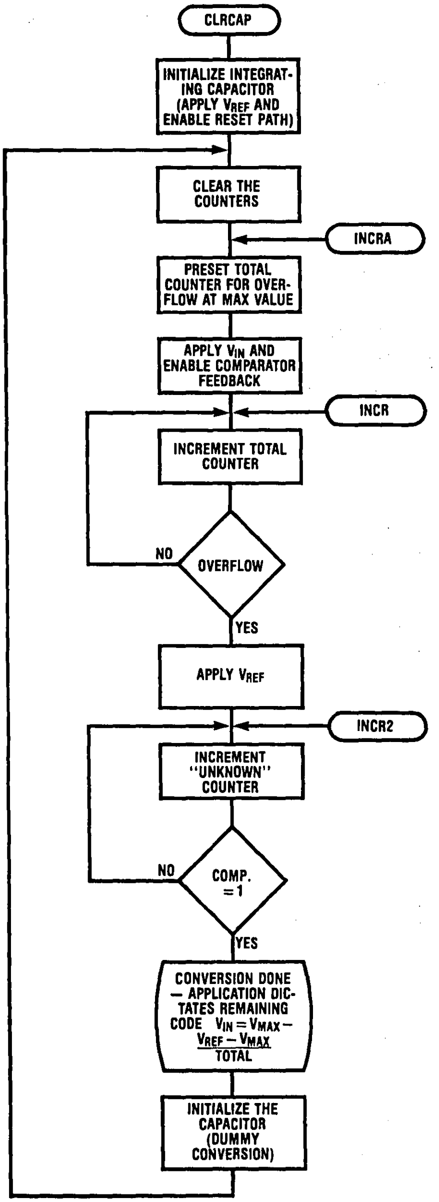
The number of components required to implement a dual slope scheme is not related to the desired accuracy. The approach is generally tolerant as to the op amps and comparators used as long as proper care is given to the initialization of the integrating capacitor.

Precise references are not required if a ratiometric system is all that is required. Cheaper switches can be safely used. The dual slope scheme controlled by a COPS microcontroller can be a very cost effective solution to an analog to digital conversion problem.

```
CLRCAP: OGI     1       ;APPLY VREF AND ENABLE RESET PATH
CLEAR2: LBI     2,11    ;NOW CLEAR THE COUNTER
        JSRP    CLEAR
  ;1,15=15,1,14=4 AND START AT 1,12 FOR COUNT = 3072
  ;1,15 =15 AND START AT 1,12 FOR COUNT = 4096
  ;1,15 = 14 AND START AT 1,12 FOR COUNT = 8192
  ;1,15 = 12 AND START AT 1,12 FOR COUNT = 16384
  ;FOLLOW SAME PATTERN FOR OTHER COUNTS
  ;
MEASUR: JSR     INCRA   ;RUN THRU THE INCREMENTS
        ;HAVE THE VALUE AT THIS POINT, DO WHAT THE APPLICATION
        ;REQUIRES--REMEMBER,TO CREATE REAL VALUE MUST MULTIPLY
        ;RESULT BY (VREF-VMAX)/TOTAL COUNT AND THEN SUBTRACT
        ;THAT RESULT FROM VMAX--DO IT IN DECIMAL OR BINARY,WHICHEVER
        ;IS BEST FOR THE APPLICATION
        LBI     1,11    ;MAKE SURE SPACE IS CLEARED
        JSRP    CLEAR
        LBI     2,11
        JSRP    CLEAR
        JSR     INCRB     ;FOR TEST-KEEP IT CLOSE
        LBI     1,11      ;MAKE SURE COUNTER IS CLEARED
        JSRP    CLEAR
        JP      CLEAR2
INCRA:  LBI     1,14
        STII    4         ;PRESET HERE FOR SMALLER COUNT
        STII    15      ;PRESET THE COUNTER FOR 4096
INCRA1: OGI     2       ;APPLY VIN AND ENABLE FEEDBACK
INCR:   LBI     1,12
        SC
BINAD1: CLRA
        ASC
        NOP
        XIS
        JP      BINAD1
        NOP             ;2 NOPS TO EQUALIZE TIMES
        NOP
        SKC
        JP      INCR
        OGI     0       ;DONE,NOW APPLY VREF
INCR2:  LBI     2,12    ;COUNT UNTIL COMPARATOR CHANGES
        SC
BINAD2: CLRA
        ASC
        NOP
        XIS
        JP      BINAD2  ;STRAIGHT LINE THE ADD FOR SPEED
        ININ            ;SAVE WORDS BY USING G
        AISC    8       ;SEE IF IN3=1
        JP      INCR2   ; IN1 IS 0,KEEP COUNTING
OUTPUT: OGI     1       ;CLEAR THE CAPACITOR,APPLY VREF
        RET
INCRB:  LBI     1,14    ;MAKE THE PASS FOR CAP INIT SHORT
        STII    7
        STII    15
        JP      INCRA1
```

TL/DD/6935-48

FIGURE 17A. Modified Dual Slope Code


CLRCAP
INITIALIZE INTEGRATING CAPACITOR (APPLY V_{REF} AND ENABLE RESET PATH)
CLEAR THE COUNTERS
INCRA
PRESET TOTAL COUNTER FOR OVERFLOW AT MAX VALUE
APPLY V_{IN} AND ENABLE COMPARATOR FEEDBACK
INCR
INCREMENT TOTAL COUNTER
NO
OVERFLOW
YES
APPLY V_{REF}
INCR2
INCREMENT "UNKNOWN" COUNTER
NO
COMP. =1
YES
CONVERSION DONE — APPLICATION DICTATES REMAINING CODE $V_{IN} = V_{MAX} - \frac{V_{REF} - V_{MAX}}{TOTAL}$
INITIALIZE THE CAPACITOR (DUMMY CONVERSION)

TL/DD/6935-25

FIGURE 17B. Modified Dual Slope Flow Chart

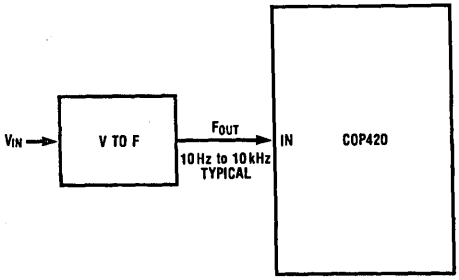
5.0 Voltage to Frequency Converters, VCO's

5.1 BASIC APPROACH

The basic idea of this scheme is simply to use the COP420 to measure the frequency output of a voltage to frequency converter or VCO. This frequency is in direct relation to the input voltage by the very nature of such devices. There are really only two limiting factors involved. First of all, the maximum frequency that can be measured is defined in the microcontroller by the amount of time required to test an input and increment a counter of the proper length. With the COP420 this upper limit is typically 10 to 15 kHz. The other limiting factor is simply the accuracy of the voltage to frequency converter or VCO. This accuracy will obviously affect the accuracy of the result.

Two basic implementations are possible and their code implementation is not significantly different. First, the number of pulses that occur within a given time period may be counted. This is straightforward and fairly simple to implement. The crucial factor is how long that given time period should be. To get the maximum accuracy from this implementation the time period should be one second. Such a time period would allow the distinction between the frequencies of 5000 Hz and 5001 Hz for example (assuming the V to F converter was that accurate or precise). Decreasing the amount of time will decrease the precision of the result. The alternate approach is to measure (by means of a counter) the amount of time between two successive pulses. This period measurement is only slightly more complicated than the pulse counting approach. The approach also makes it possible to do averaging of the measurement during conversion. This will smooth out any changes and add stability to the result. The time measurement technique is also faster than the pulse counting approach. Its accuracy is governed by how finely the time periods can be measured. The greater the count that can be achieved at the fastest input frequency — shortest period — the more accurate the result.

Figure 18 illustrates the basic concept. *Figure 19A* shows the flow charts and code implementation for both of the approaches discussed above. Note that whatever type of V to F converter is used, the code illustrated in *Figure 19A* is not significantly changed. In the code of *Figure 19A*, the interrupt is being used to test an input and thereby decreases the total time loop.



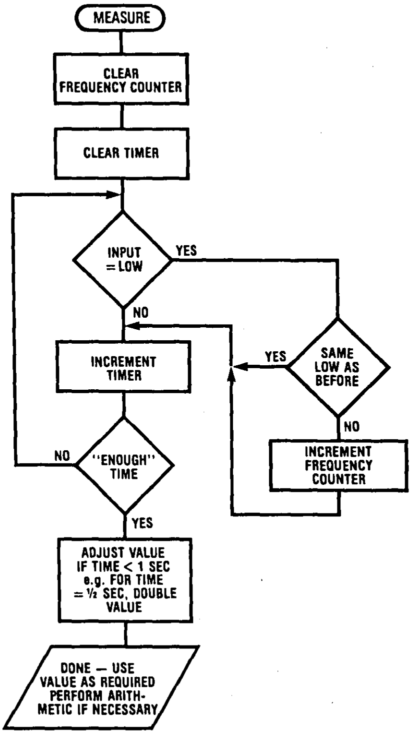
TL/DD/6935-26

FIGURE 18. V to F Converter — Basic Concept

```
MEASUR:            ;MEASURE BY COUNTING PULSES OF V TO F
;
        LEI     2       ;ENABLE INTERRUPT
        LBI     1,14    ;PRESET TIME FOR 122 COUNTS
        STII    5       ;APPROX ONE HALF SECOND
        STII    8       .
TIME:   SKT             ;USE INTERNAL TIMER TO FIND
        JP      TIME    ;THE 1/2 SECOND
BINPL1: LBI     1,14    ;HAVE GOT IT,INCREMENT COUNTER
        SC
BINADD: CLRA
        ASC
        NOP
        XIS
        JP      BINADD
        SKC             ;NOW SEE IF DONE
        JP      TIME    ;NO COUNTER OVERFLOW,CONTINUE
        LEI     0       ;DONE,DISABLE INTERRUPT
FIN:    ;AT THIS POINT HAVE THE VALUE--CONVERT IT TO DECIMAL OR
        ;SEND IT OUT OR PROCESS IT FURTHER, WHATEVER IS REQUIRED
        ;BY THE APPLICATION.  ARITHMETIC IS REQUIRED TO CREATE THE
        ;VOLTAGE VALUE, USUALLY A SIMPLE MULTIPLY
        ;MAY HAVE TO DOUBLE THE RESULT TO COMPENSATE LOOKING FOR
        ;ONLY 1/2 SECOND IN THIS CASE
        ;
        JP      MEASUR    ;DO IT OVER AGAIN
        .=X'OFF         ;SET ADDRESS TO OFF FOR INTERRUPT
INTENT: NOP             ;ADDRESS OFF MUST BE NOP FOR INTERRUPT
INTRPT: LBI     2,12    ;DO ADD OF THE VALUE FOR FREQ CNT
        SC
INTR1:  CLRA            ;STRAIGHT LINE THE CODE FOR SPEED
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        X
        LEI     2       ;ENABLE THE INTERRUPT AGAIN
        RET
```

TL/DD/6935-49

FIGURE 19A. V to F by Counting Pulses


MEASURE
CLEAR FREQUENCY COUNTER
CLEAR TIMER
INPUT = LOW
YES
NO
INCREMENT TIMER
SAME LOW AS BEFORE
YES
NO
INCREMENT FREQUENCY COUNTER
"ENOUGH" TIME
NO
YES
ADJUST VALUE IF TIME < 1 SEC e.g. FOR TIME = ½ SEC, DOUBLE VALUE
DONE — USE VALUE AS REQUIRED PERFORM ARITHMETIC IF NECESSARY

TL/DD/6935-27

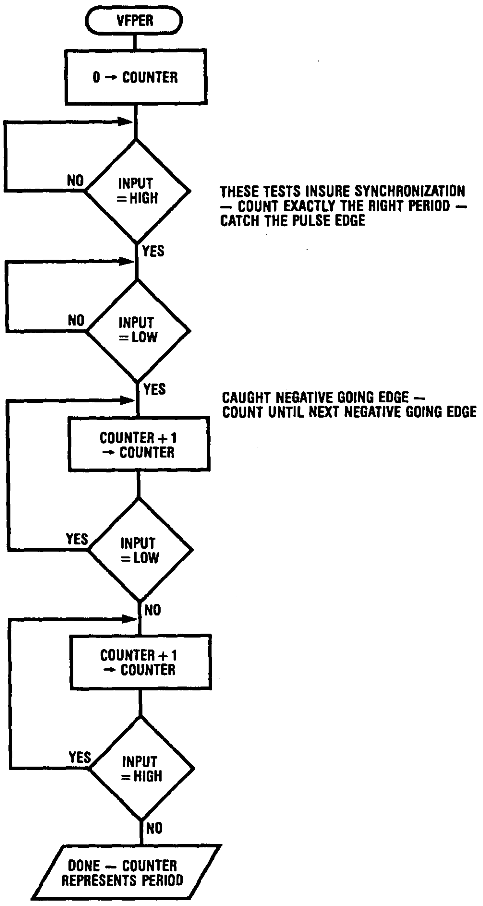
FIGURE 19B. V to F by Counting Pulses

```
        ;USE INTERRUPT FOR CATCHING THE PULSE EDGE

VFPER:  LBI     0,12    ;CLEAR COUNTER SPACE AND FLAG
        STII    0
        STII    0
        STII    0
        STII    0
        LBI     0,12
        LEI     2       ;NOW ENABLE THE INTERRUPT
WAIT:   SC              ;DUMMY WAIT LOOP,WAITING FOR SIGNAL TO
        LBI     0,12    ;INTERRUPT THE CONTROLLER
        JP      WAIT
        .=X'OFF         ;SET ADDRESS TO OFF--INTERRUPT ENTRY POINT
INTENT: NOP             ;REQUIRED FOR INTERRUPT ENTRY
COUNT:  LBI     0,12    ;NOW CHECKING TO SEE IF SECOND INTERRUPT
        SKMBZ   0       ;I.E.,ARE WE DONE?
        JP      DONE
        SMB     0       ;SET BIT FOR NEXT INTERRUPT
        LEI     2       ;ENABLE INTERRUPT AGAIN
PLUS1:  LBI     0,13    ;NOW START COUNTING
        SC
        CLRA            ;STRAIGHT LINE THE CODE FOR SPEED
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        XIS
        CLRA
        ASC
        NOP
        X
        JP      PLUS1
DONE:   ;FINISHED WHEN GET HERE--THE COUNT REPRESENTS THE PERIOD
        ;WITH ABOVE CODE, THE ACTUAL PERIOD IS THE COUNT MULTIPLIED
        ;BY 15(THE NUMBER OF WORDS TO INCREMENT BY 1) PLUS AN OVERHEAD
        ;OF 9 CYCLE TIMES =  24 CYCLE TIMES.  AT 4us THIS IS 96 us
        ;OR A FREQUENCY OF JUST OVER 10KHz.  MAX COUNT HERE IS 4095.
        ;THIS GIVES A MAXIMUM PERIOD = 61434 CYCLE TIMES(=245.736ms AT
        ;4us).  THIS CORRESPONDS TO A FREQUENCY OF JUST OVER 4Hz
        ;NOTE,THIS IS 12 BIT RESOLUTION
```

TL/DD/6935-50

FIGURE 19C. A to D with VF Converter/VCO by Measuring Period

3



THESE TESTS INSURE SYNCHRONIZATION — COUNT EXACTLY THE RIGHT PERIOD — CATCH THE PULSE EDGE

CAUGHT NEGATIVE GOING EDGE — COUNT UNTIL NEXT NEGATIVE GOING EDGE

TL/DD/6935-28

FIGURE 19D. V to F—Measure Period

5.2 THE LM131/LM231/LM331

The LM131 is a standard product voltage to frequency converter with a linear relationship between the input voltage and the resultant frequency. The reader should refer to the data sheet for the LM131 for further information on the device itself and precautions that should be taken when using the device. *Figure 20* is the basic circuit for using the LM131. *Figure 21* represents improvements that increase the accuracy (by increasing the linearity) of the result. Note that these circuits have been taken from the data sheet of the LM131 and the user is referred there for a further discussion of their individual characteristics. With the LM131 the frequency output is given by the relationship:

$$F_{OUT} = (V_{IN}/2.09)\ (1/R_TC_T)\ (R_S/RL)$$

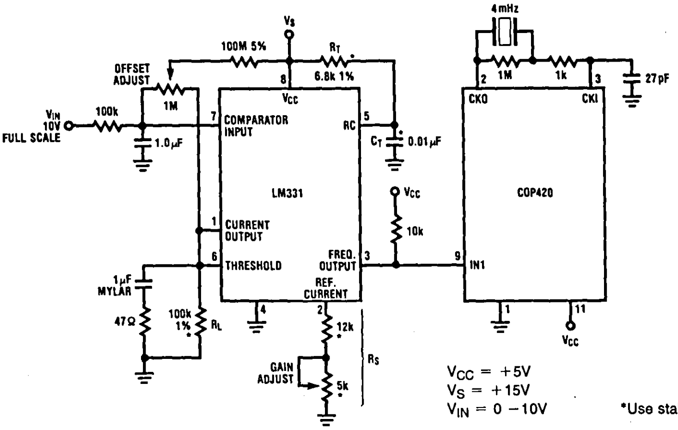
It is clear from the expression above that the accuracy of the result depends upon the accuracy of the external components. The circuit may be calibrated by means of a variable resistance in the R_S term (a gain adjust) and an offset adjust. The offset adjust is optional but its inclusion in the circuit will allow maximum accuracy to be obtained. The standard calibration procedure is to trim the gain adjust (R_S) until the output frequency is correct near full scale. Then set the input to 0.01 or 0.001 of full scale and trim the offset adjust to get F_{OUT} to be correct at 0.01 or 0.001 of full scale. With that calibration, the circuit of *Figure 20* is accurate to within ±0.03% typical and ±0.14% maximum. The circuit of *Figure 21* attains the spec limit accuracy of ±0.01%.

5.3 VOLTAGE CONTROLLED OSCILLATORS (VCO's)

A VCO is simply another form of voltage to frequency converter. It is an oscillator whose oscillation frequency is dependent upon the input voltage. Numerous designs for VCO's exist and the reader should refer to the data sheets and application notes for various op-amps and VCO devices. The code in *Figure 19* is still applicable if a VCO is used. The only possible difficulty that might be encountered is if the relationship between frequency and input voltage is non-linear. This does not affect the basic code but would affect the processing to create the final result. A sample circuit, taken from the data sheet of the LM358, is shown in *Figure 22*. The accuracy of the VCO is the controlling factor.

5.4 A COMBINED APPROACH

Elements of the period measurement and pulse counting techniques can be combined to produce a system with the advantages of both schemes and with few problems. Such a system is only slightly more complicated in terms of its software implementation than the approaches mentioned above. Note that in a microcontroller driven system, no additional hardware beyond the voltage to frequency converter is required to implement this approach. Basically, the microcontroller establishes a viewing window during which time the microcontroller is both measuring time and counting pulses. The result can be very precise if two conditions are met. First, when the microcontroller determines that it needs the conversion information, the microcontroller does not begin counting time or pulses until the first pulse is received from the VFC (first pulse after the microcontroller "ready"). Note, the COPS microcontroller could provide a "start conversion" pulse to enable the VFC if such an arrangement were desirable. The time would be counted for a fixed period and the number of pulses would be counted. After the fixed period of time the controller would wait for the next pulse from the VFC and continue to count time until that pulse is received. The ratio of the total time to the number of pulse is a very precise result provided that all the system times are slow enough that the microcontroller can do its job. The speed limits mentioned previously apply here. It is clear that the total time is not fixed. It is some basic time period plus some variable time. This is a little more complicated than simply using a fixed time, but it allows greater accuracies to be achieved. Also, the approach takes approximately the same amount of time for all conversions. It is also faster than the simple pulse counting scheme.

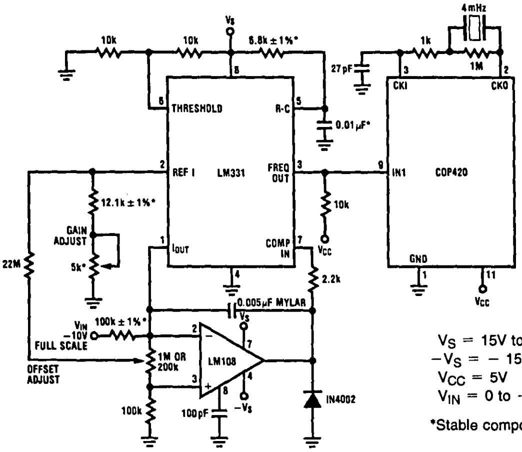


VCC = +5V
VS = +15V
VIN = 0 −10V

*Use stable components

TL/DD/6935-29

FIGURE 20. Basic LM331 Connection

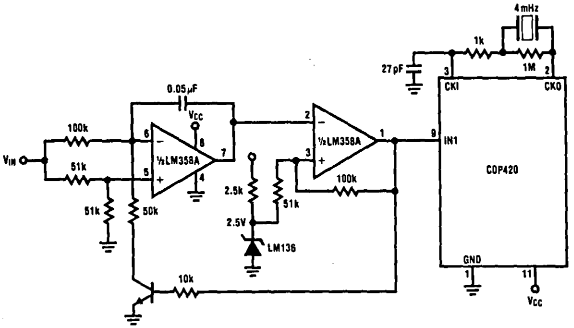


VS = 15V to 5V
−VS = − 15V to −5V
VCC = 5V
VIN = 0 to −10V

*Stable components should be used

TL/DD/6935-30

FIGURE 21. A to D with Precision Voltage to Frequency Converter



VCC = +5V
VIN = 0-5V

TL/DD/6935-31

FIGURE 22. A to D with VCO

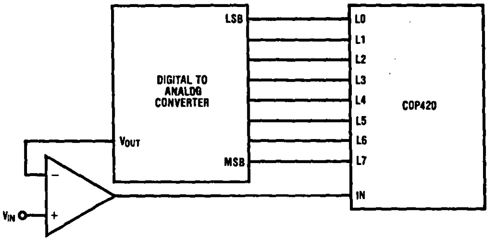
COP NOTE 1

6.0 Successive Approximation

6.1 BASIC APPROACH

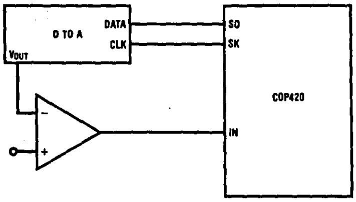
The successive approximation technique is one of the more standard approaches in analog to digital conversion. It requires a counter or register (here provided by the COP420), a digital to analog converter, and a comparator. *Figure 23A/B* illustrates the basic idea with the COP420. In the most basic scheme, the counter is reset to zero and then incremented until the voltage from the digital to analog converter is equal to the input voltage. The equality is determined by means of the comparator. *Figure 24B* illustrates the flow chart and code for this most basic approach. The preferred approach is illustrated in *Figure 25A/B*. This is the standard binary search method. The counter or register is set at the midpoint and the "delta" value set at one half the midpoint. The "delta" value is added or subtracted from the initial guess depending on the output of the comparator. The "delta" value is divided by 2 before the next increment or decrement. The method repeats until the desired resolution is achieved. While this approach is somewhat more complicated than the basic approach it has the advantage of always taking the same amount of time for the conversion regardless of the value of the input voltage. The conversion time for the basic approach increases with the input voltage. The preferred approach is almost always faster than the basic approach. The basic approach is faster only for those voltages near zero where it has only a few increments to perform.

The accuracy of the approach is governed by the accuracy of the digital to analog converter and the comparator. Thus, the result can be as accurate as one desires depending on the choice of those components. Digital to analog converters of various accuracies are readily available as standard parts. Their cost is usually in direct relation to their accuracy. The reader should refer to the National Semiconductor Data Acquisition Handbook for some possible candidates for digital to analog converters. It is not the purpose here to compare those parts. The COPS interface to these parts is generally straightforward and follows the basic schematics shown in *Figure 23*. The user should take note and make sure the input and output ports of the converter are compatible — in terms of voltages and currents — with the COPS device. This is generally not a problem as most of the parts are TTL compatible on input and output. The precautions and restrictions as to the use of any given device are governed by that device and are indicated in the respective data sheets.



TL/DD/6935-32

FIGURE 23A. Basic Parallel Implementation



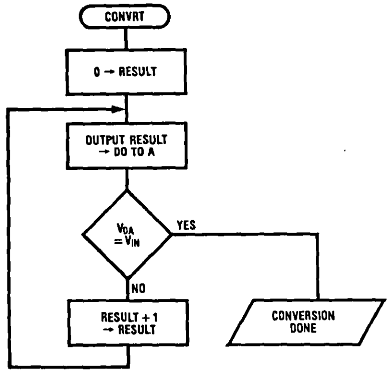
TL/DD/6935-33

FIGURE 23B. Basic Serial Implementation

```
         ;8 BIT SUCCESSIVE APPROXIMATION--BASIC SCHEME
         ;COMPARATOR INPUT TO COP = IN3
         ;OUTPUTS TO D TO A ARE L7 THRU LO WITH L7 = MSB,LO = LSB

CONVRT:  LBI     2,14     ;SET THE RESULT VALUE TO ZERO
         STII    0
         STII    0
         LEI     4        ;ENABLE THE L PORT AS OUTPUTS
         JP      OUTPUT
INCR:    SC               ;ROUTINE FOR INCREMENTING THE RESULT VALUE
PLUS1:   CLRA
         LBI     2,14
         ASC
         NOP
         XIS
         JP      PLUS1
OUTPUT:  LBI     2,15     ;SEND THE RESULT VALUE,STORED IN 2,15-2,14 TO
         LD               ;Q AND THEREBY OUT THROUGH L
         XDS
         CAMQ
         JSR     DELAY    ;THIS IS ANY CONVENIENT ROUTINE TO MAKE SURE
                          ;THAT THE COP DOES NOT TEST THE COMPARATOR UNTIL
                          ;THE D TO A CONVERTER HAS HAD ENOUGH TIME TO DO
                          ;THE CONVERSION--THE AMOUNT OF TIME REQUIRED
                          ;IS CLEARLY DEPENDANT UPON THE D TO A CONVERTER
                          ;USED
         ININ             ;NOW READ THE COMPARATOR INPUT TO COP
         AISC    8        ;COULD SAVE A WORD IF USE G LINE AS INPUT
         JP      INCR     ;INPUT VOLTAGE STILL > CONVERTED ANALOG VOLTAGE

         ;CONVERSION DONE AT THIS POINT--THE COMPARATOR HAS CHANGED STATE
         ;HENCE,CONVERTED ANALOG VOLTAGE > INPUT VOLTAGE--SO STOP
```

TL/DD/6935-51

FIGURE 24A. Code for Basic Approach of Successive Approximation



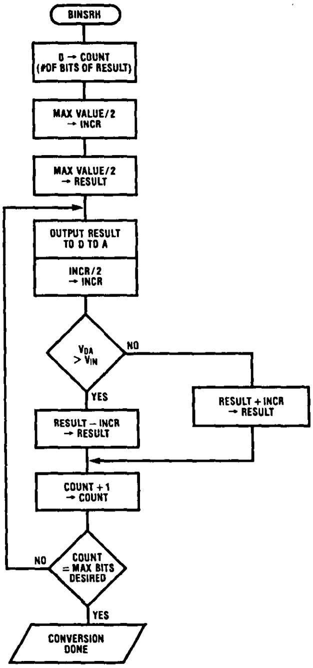
TL/DD/6935-34

FIGURE 24B. Basic Approach, Successive Approximation

```
        ;8 BIT BINARY SEARCH SUCCESSIVE APPROXIMATION
        ;INPUT TO COP IS IN3,L BUS IS OUTPUT TO D TO A,L7=MSB,LO=LSB
        ;COMPARATOR=0 WHEN D TO A VOLTAGE > VIN, OTHERWISE = 1

BINSRH: LBI     3,14    ;SET INCREMENT = MAX VALUE/2(WILL BECOME
        STII    0       ;MAX VALUE/4 BEFORE FIRST USE)
        STII    8
        LBI     2,14    ;SET INITIAL VALUE OF RESULT TO MAX VALUE/2
        STII    0
        STII    8
        LEI     4       ;ENABLE THE L BUS AS OUTPUTS
        LBI     1,15    ;NOW SET UP THE BIT COUNTER-OVERFLOW WHEN 8 BITS
        CLRA
        AISC    9       ;DO IT THIS WAY FOR COMPATIBILITY WITH INCREMENT
OUTPUT: X       3       ;SAVE THE BIT COUNTER VALUE AND POINT TO RESULT
        LD
        XDS             ;SEND THE RESULT TO Q AND HENCE TO L
        CAMQ
DIVIDE: LBI     3,15    ;DIVIDE THE INCREMENT VALUE BY 2, CAN BE DONE
DIVA:   LD              ;IN SEVERAL WAYS SINCE THIS IS A VERY SPECIAL
        AISC    8       ;PURPOSE DIVIDE FUNCTION
        JP      DIV1    ;ALSO,DO THE DIVIDE HERE TO GIVE THE D TO A TIME
        STII    4       ;TO DO THE DIGITAL TO ANALOG CONVERSION
        JP      TEST
DIV1:   AISC    4
        JP      DIV2
        STII    2
        JP      TEST
DIV2:   AISC    2
        JP      DIV3
        STII    1
        JP      TEST
DIV3:   LBI     3,14
        AISC    1
        JP      DIVA
        STII    8
        STII    0
        ;DEPENDING ON THE D TO A USED, MAY NEED MORE DELAY HERE
        ;MUST BE SURE THE RESULT IS STEADY BEFORE TEST THE COMPARATOR
TEST:   LBI     3,14
        ININ
        AISC    8       ;COULD SAVE A WORD IF USED G LINE AS INPUT
        JP      INCR
DECR:   SC              ;INPUT LESS THAN D TO A CONVERTED VOLTAGE
SUB:    LD      1       ;SUBTRACT THE INCREMENT VALUE FROM RESULT
        CASC
        NOP
        XIS     1
        JP      SUB
        JP      BITPL1
INCR:   RC              ;INPUT > D TO A CONVERTED VOLTAGE
ADD:    LD      1       ;ADD THE INCREMENT VALUE TO RESULT VALUE
        ASC
        NOP
        XIS     1
        JP      ADD
BITPL1: LBI     1,15    ;NOW INCREMENT BIT COUNTER TO SEE IF DONE
        LD
        AISC    1
        JP      OUTPUT
        ;CONVERSION DONE AT THIS POINT
```

TL/DD/6935-52

FIGURE 25A. Binary Search Successive Approximation Code



TL/DD/6935-35

FIGURE 25B. Binary Search Successive Approximation Flow Chart

6.2 SOME COMMENTS ON RESISTOR LADDERS

If the user does not wish to use one of the standard digital to analog converters, he can always build one of his own. One of the most standard methods of doing so is to use a resistor ladder network of some form. *Figure 26* illustrates the basic forms of binary ladders for digital to analog converters. The figures also show the transition from the basic binary weighted ladder in *Figure 26A* to the standard R-2R ladder *Figure 26C*.

Consider *Figure 26A*. The choice of the terminating resistor is made by hypothesizing that the ladder were to go on ad infinitum. It can then be shown that the equivalent resistance at point X in that figure would be equal to 128R, the same value as the resistor to the least significant bit output. This fact is used to create the intermediate ladder of *Figure 26B*. This step is done because it is usually undesirable to have to find the multitude of resistor values required in the basic binary ladder. Thus, the modification in *Figure 26B* significantly reduces the number of resistor values required. As stated earlier, the resistance looking down the ladder at point X in *Figure 2* is equal to the resistor connected to the binary output at that point; here the value is 2R. Remembering the objective is to minimize the number of different values required, if we simply use the same R-2R arrangement as before with a termination of 2R we get an effective resistance at point Y of *Figure 26B* or 0.5R. This means that a serial resistance of 1.5R is required to maintain the integrity of the ladder. If we carry this on through 8 bits, the circuit of *Figure 26B* results. From this it is only a small step to create the standard R-2R network. The analysis is the same as done previously.

There is absolutely no restriction that the ladders must be binary. A ladder for any type of code can be constructed with the same techniques. Ladders comparable to *Figures 26A* and *26B* are shown in *Figure 27* for a standard 8421 BCD code. With the BCD code, the input must be considered in groups of digits with four bits creating one digit. This is the direct analog of 1 binary digit per unit. We need four inputs to create one decimal digit. Thus the resistor values in each decimal digit are 10 times the values in the previous decimal digit just as the resistor value for each successive binary digit was twice the value for the preceding binary digit. Note that this analysis can be easily extended to any code. The termination resistance is calculated in the same manner—assume the decimal digit groupings extend out to infinity. It can be shown that the resistance of the ladder at point X in *Figure 27A* is 480R. Thus *Figure 27A* represents the basic 8241 BCD ladder for three digit BCD number. This termination resistance will vary with where it is placed. Basically this resistance is equal to nine times (for a decimal ladder) the parallel resistance of the last digit implemented. (This relation can be shown mathematically if one desired, the multiplier is a function of the type of ladder used—multiplier = 1 for binary systems, 9 for decimal systems, etc.) Thus the termination resistance would be 48R if the network were terminated after the 2nd digit and 4.8R if the network were terminated after the 1st digit implemented. In

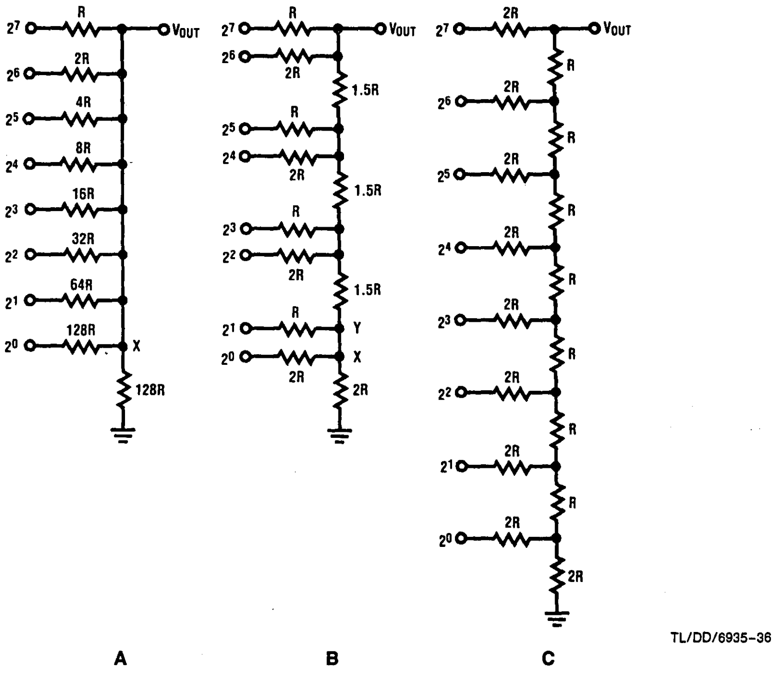


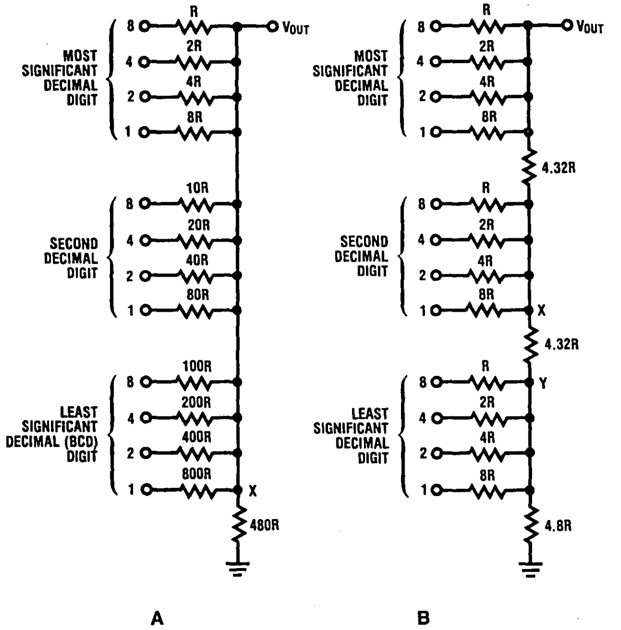
FIGURE 26. Binary Ladders

Figure 27B we are attempting to use only the resistor values for one decimal digit. This means that the last terminating resistor must be a 4.8R by the analysis above. Thus at point X in *Figure 27B* we must have an equivalent of resistance of 4.8R. The equivalent resistance at point Y of *Figure 27B*, looking down from the ladder, is 0.48R. Thus the other series resistance must be 4.32 R (4.8R−0.48R). Thus the network of *Figure 27B* results.

Generally, ladders can be very effective tools when understood and used properly. They can be significantly more involved than indicated here. There are a number of texts and articles that cover the subject very nicely and the reader is referred to them if more information on ladder design, the use of ladders, and advanced techniques with ladders is desired.

One final note is of some interest. The ladders may be readily constructed for any type of code to create the analog voltage. Note that there is no restriction that the code, or the ladder network, be linear. Thus, effective use of ladder networks may significantly reduce system difficulties and complexities caused by the fact that the analog to digital conversion is being performed on a voltage source that changes nonlinearly, for example a thermistor temperature probe. By using the properly designed ladder network, the nonlinearity can effectively be eliminated from consideration in the code implementation of the analog to digital conversion.

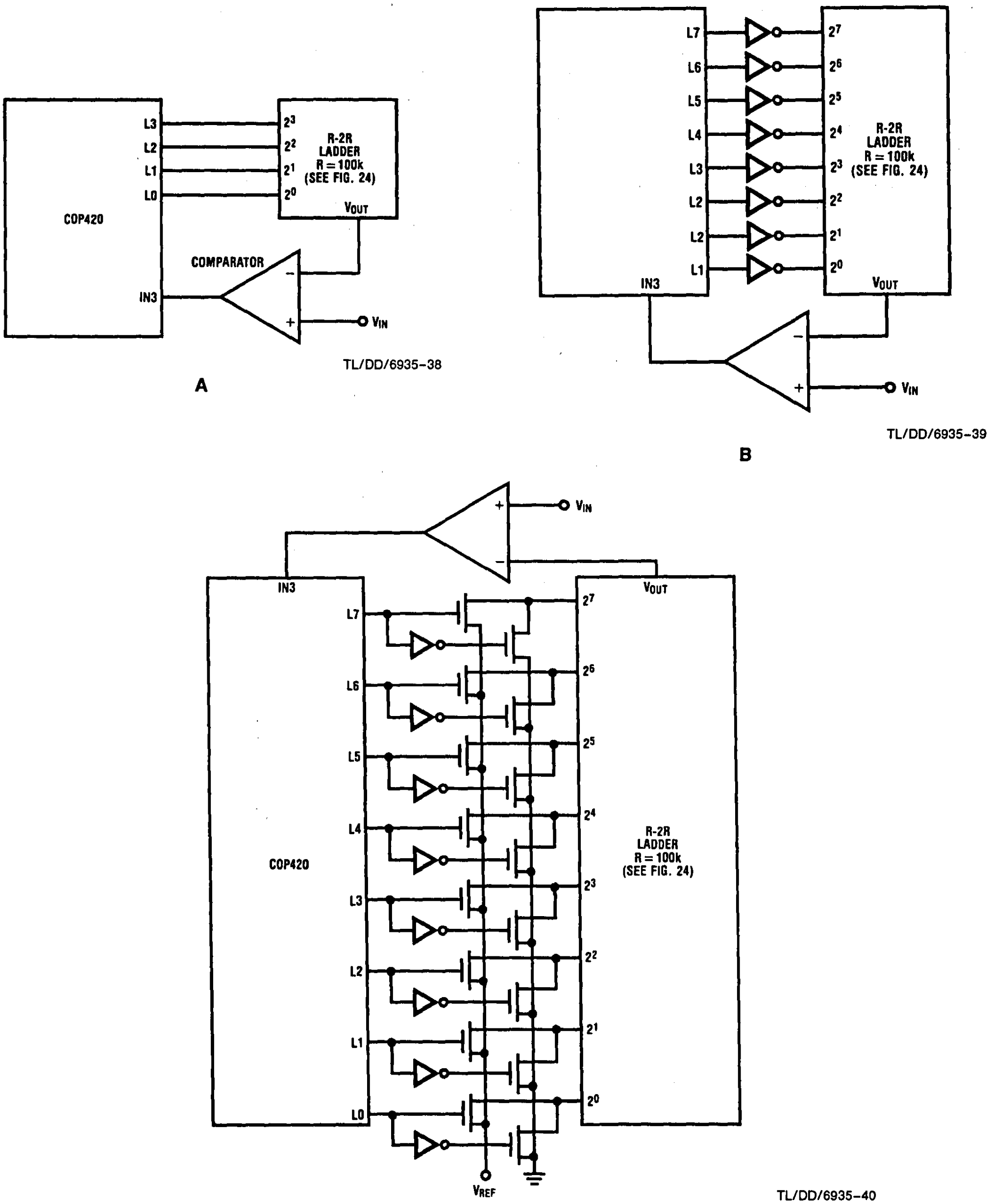
The accuracy of ladders is a direct function of the accuracy of the resistors and the accuracy of the voltage source inputs. This is obvious since the analog voltage is in fact created by means of equivalent voltage dividers created when the various inputs are on or off. It is also essential that the ladder sources be the precise same value at all inputs to the ladder network. If this is not the case, errors will be introduced. In addition, the output impedance of the voltage source should be as small as possible. The success of the ladder scheme depends on the ratios of the resistance values. Inaccuracies are introduced if those ratios are disturbed. Some possible implementations of the successive approximation approach with a ladder network used for the digital to analog conversion are indicated in *Figure 28.*


MOST SIGNIFICANT DECIMAL DIGIT
8
4
2
1
R
2R
4R
8R
V_{OUT}
SECOND DECIMAL DIGIT
10R
20R
40R
80R
LEAST SIGNIFICANT DECIMAL (BCD) DIGIT
100R
200R
400R
800R
X
480R
MOST SIGNIFICANT DECIMAL DIGIT
SECOND DECIMAL DIGIT
LEAST SIGNIFICANT DECIMAL DIGIT
4.32R
X
4.32R
Y
4.8R
A
B

TL/DD/6935-37

FIGURE 27. 8421 BCD Ladders

Note that these are functional diagrams. Feedback or hysteresis for comparator stabilization are not shown. The reader should be aware that his particular application may require that these factors be considered. *Figure 28A* is the simplest scheme and also the least accurate. With little or no load, the high output level of the L buffer should be very close to V_{CC} and the low level close to ground. Also the output impedance of the buffers must be considered. Therefore, rather large resistor values are used—both to keep the load very small and to dwarf the effect of the output impedance. With the configuration in *Figure 28A*, four bit accuracy is about the best that can be achieved. By being extremely careful and using measured values, an additional bit of accuracy may be obtained but care must be used. However, the schematic of *Figure 28A* is very simple. *Figure 28B* represents the next step of improvement. Here we have placed CMOS buffers in the network. This eliminates the output impedance and reduces the level problems of the circuit of *Figure 28A*. The CMOS buffer will swing rail to rail, or nearly so. The accuracy of V_{CC} and the resistor network is then



TL/DD/6935-38

A

TL/DD/6935-39

B

TL/DD/6935-40

C

FIGURE 28. Interfaces to Ladder Networks

controlling. Using 1% resistors and holding V_{CC} constant, the user should be able to achieve 7 to 8 bit accuracy without much difficulty. Remember, however, that V_{CC} is one of the controlling factors. If V_{CC} is ±5%, there is no point in using 1% resistors since the V_{CC} tolerance swamps their effect. *Figure 28C* is the final and most accurate approach. Naturally enough, it is the most expensive. However, one can get as accurate as one desires. Here, an accurate reference is required. That reference is switched into the network by means of the analog switch. Alternately, ground may be connected to the input. Now the user need only consider the accuracy of the reference and the accuracy of the resistors. However, the on impedance of the switches must be considered. It is necessary to make this on impedance as low as possible so as not to alter the effective resistor values.

7.0 "Offboard" Techniques

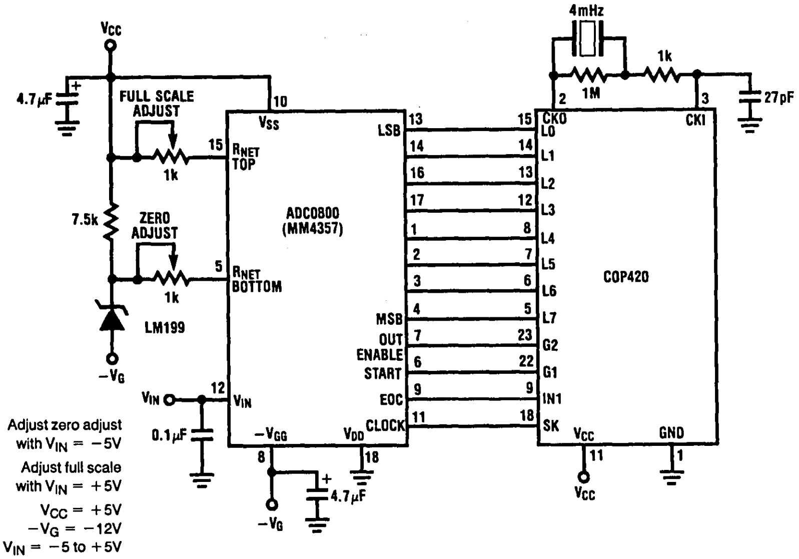
7.1 GENERAL COMMENTS

This section is devoted to a few illustrations of interfacing the COP420 to standard, stand alone analog to digital converters. These standard converters are used as peripherals to the COPS device. Whenever the microcontroller requires a new reading of some analog voltage, it simply initiates a read of the peripheral analog to digital converter. As a result, the accuracies and restrictions in using the converters are governed by those devices and not by the COPS device. These techniques are generally applicable to other A to D converters not mentioned here and the user should not have difficulty in applying these principles to other devices. It should be pointed out that in almost every instance, the choice of COP420 inputs and outputs is arbitrary. Obviously, when there is an 8-bit bus it is natural, and most efficient, to use the L port to interface to the bus. Generally, the G lines have been used as outputs rather than the D lines simply because the G lines are, in many instances, somewhat easier to control. The choice of input line is also free. If the interrupt is not otherwise being used, it may be possible to utilize this feature of IN1 for reading a return signal from the converter. However, this is by no means required. If there is a serial interface it is clearly more efficient to use the serial port of the COP420 as the interface. If a clock is required, SK is the natural choice.

7.2 ADC0800 INTERFACE

The ADC0800 is an 8-bit analog to digital converter with an 8-bit parallel output port with complementary outputs. The ADC0800 requires a clock and a start convert pulse. It generates an end of conversion signal. There is an output enable which turns the outputs on in order to read the 8-bit result.

The reader is referred to the data sheet for the ADC0800 for more information on the device. The circuit of *Figure 29* illustrates the basic implementation of a system with the ADC0800. The interface to the COP420 is straightforward. The appropriate timing restrictions on the control signals are easily met by the microcontroller.



Adjust zero adjust with V_{IN} = −5V
Adjust full scale with V_{IN} = +5V
V_{CC} = +5V
$-V_G$ = −12V
V_{IN} = −5 to +5V

TL/DD/6935-41

FIGURE 29. Simple A/D with ADC0800

Figure 30 is the flow chart and code required to do the interfacing. As can be seen, the overhead in the COP420 device is very small. The choice of inputs and outputs is arbitrary. The only pin that is more or less restricted is the use of SK as the clock for the converter. SK is clearly the output to use for that function as, when properly enabled, it provides pulses at the instruction cycle rate.

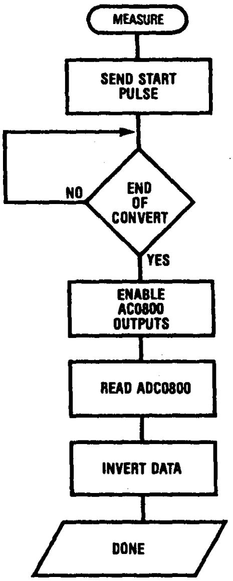
7.3 ADC0801/2/3/4 INTERFACE

The ADC0801 family of analog to digital converters is very easy to interface and is generally a very useful offboard converter. The interface is not significantly different from that of the ADC0800, but the ADC0801 famliy are a much better device. The four control signals are somewhat different, although there are still four control lines. Here we have a chip select, a read, a write, and an interrupt signal. All are negative going signals. Start conversion is the ANDing of chip select and write. Output enable is the ANDing of chip select and read. The interrupt output is an end convert signal of sorts. The device may be clocked externally or an RC may be connected to it and it will generate its own clock for the conversion. In addition the device has differential inputs

```
MEASUR: LEI     0       ;FLOAT THE L LINES
        SC
START2: CLRA            ;MAKE SURE SO STAYS ZERO
        XAS             ;MAKE SURE SK STAYS CLOCK
        OGI     2       ;SEND START PULSE
        OGI     0
        LBI     2,13
READI1: ININ
        AISC    14      ;WAIT FOR EOC SIGNAL
        JP      READI1
        OGI     4       ;HAVE EOC,ENABLE OUTPUTS
        INL             ;READ THE L LINES
        X
        COMP            ;CREATE PROPER POLARITY
        XDS
        COMP
        X
        OGI     0       ;DISABLE ADC0800 OUTPUT
        ;HAVE THE RESULT AT THIS POINT--USE IT IN WHATEVER
        ;MANNER IS REQUIRED BY THE APPLICATION
        LBI     2,10
        JSRP    CLRR
        JP      MEASUR
```

TL/DD/6935-53

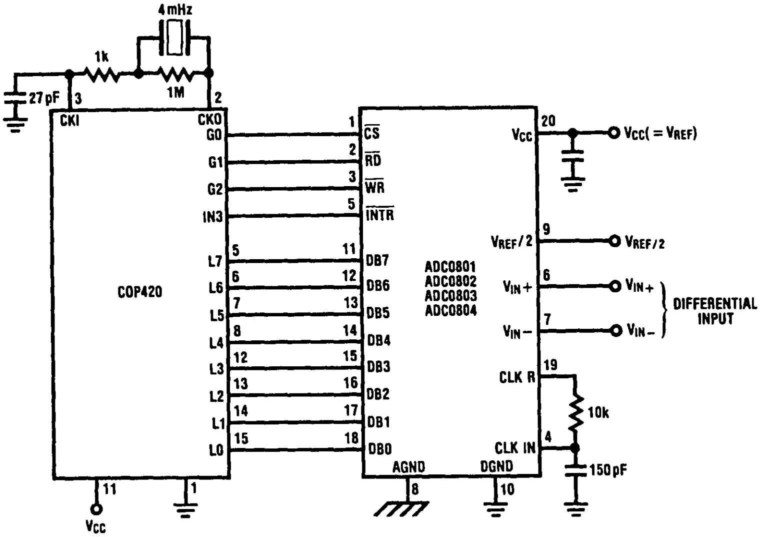
FIGURE 30A. A to D with ADC0800



TL/DD/6935-42

FIGURE 30B. ADC0800 Interface Flow

which allow the 8-bit conversion to be performed over a given window or range of input voltages. The reader should refer to the ADC0801 family data sheet for more information. *Figure 31* indicates a basic interface of the ADC0801 family to the COP420. Again, the interface is simple and straightforward. The code required to interface to the device is minimal. *Figure 32* illustrates the flow chart and code required to do the interface.



TL/DD/6935-43

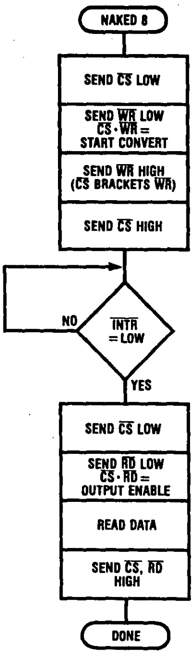
FIGURE 31. COP420—ADC0801 Family Interface

```
         ; INTERFACE TO NAKED 8
         ;
NAKED8:  OGI     15     ;SET ALL G LINES HIGH(USUALLY DONE AT
                        ;POWER UP
         LEI     0      ;TRI STATE THE L LINES FOR READING
LOOP:    OGI     14     ;SEND CHIP SELECT LOW(CS BRACKETS OTHER SIGNAL)
         OGI     10     ;CS LOW AND WR LOW = START CONVERSION
         OGI     14     ;RAISE WR
         OGI     15     ;RAISE CS, NAKED 8 IS NOW CONVERTING
LOOP2:   ININ           ;WAIT FOR THE INTR SIGNAL--COULD SAVE THIS TES
         AISC    8      ;IF USED IN1 AND THE INTERRUPT FEATURE OF COP4
         JP      READ   ;INTR IS LOW, DATA IS READY
         JP      LOOP2
READ:    LBI     0,0    ;SET UP RAM LOCATION FOR READ
         OGI     14     ;SEND CS
         OGI     12     ;SEND CS AND READ = OUTPUT ENABLE
         NOP            ;WAIT-NEED WAIT ONLY 125NS, BUT 1 CYCLE IS MIN
                        ;TIME WE CAN WAIT
         INL            ;READ THE L LINES
         OGI     15     ;TURN OFF THE NAKED 8--CS AND RD HIGH
         ;
         ;DONE AT THIS POINT, DO WHATEVER IS REQUIRED  WITH THE RESULT
         ;
```

TL/DD/6935-54

FIGURE 32A. COP420/ADC0801 Family Sample Interface Code

COP NOTE 1



TL/DD/6935-44

FIGURE 32B. COP420/ADC0801 Family Interface Flow

8.0 Conclusion

Several analog to digital techniques using the COPS family have been presented. These are by no means the only techniques possible. The user is limited only by his imagination and whatever parts he can find. The COPS family of parts is extremely versatile and can readily be used to perform the analog to digital conversion in almost any method. Generally, those techniques where the COPS device is doing the counting or timekeeping are slow. However, those techniques are generally slow inherently. The fastest methods are those where the conversion is being done offboard and the COPS device is merely reading the result of the conversion when required. Also, an attempt has been made to illustrate the lower cost techniques of analog to digital conversion. This, by itself, restricts most of the techniques described to about 8-bits accuracy. As was mentioned several times, the greater the accuracy that is desired the more accurate the external circuits must be. Ten and twelve-bit accuracies, and more, require references that are accurate. These get very expensive very rapidly. There is nothing inherent in the COPS devices that prevents them from being used in accurate systems. The precautions are to be taken in the system regardless of the microcontroller. The only problem is that, in those accurate systems where the COPS device is doing the timekeeping and counting, this increased accuracy is paid for by increased time to perform the conversion.

Several devices have been used in conjunctions with the COPS device in the previous sections. It is again recommended that the user refer to the specific data sheets of those devices when using any of those circuits. It must again be mentioned that the standard precautions when dealing with analog signals and circuits must be taken. These are described in the National Semiconductor Linear Applications Handbook and in the data sheets for the various linear devices. These precautions are especially significant when greater accuracy is desired.

The COPS family of microcontrollers has shown itself to be very versatile and powerful when used to perform analog to digital conversions. Most techniques are code efficient and the microcontroller itself is almost never the limiting factor. It is hoped that this document will provide some guidance when it is necessary to perform analog to digital conversion in a COPS system.

9.0 References

1. "Digital Voltmeters and the MM5330", National Semiconductor Application Note AN-155.
2. Walker, Monty, "Exploit Ladder Network Design Potential". Part One of two part article on ladder networks. Magazine and date unknown.
3. Wyland, David C., "VFC's give your ADC design high resolution and wide range". *EDN*, Feb. 5, 1978.
4. Redfern, Thomas P., "Pulse Modulation A/D Converter" *Society of Automotive Engineers Congress and Exposition Technical paper #780435*, March 1978.
5. National Semiconductor Linear Applications Handbook, 1978.
6. National Semiconductor Linear Databook, 1980.
7. National Semiconductor Data Acquisition Handbook, 1978.

The COP444L Evaluation Device 444L-EVAL

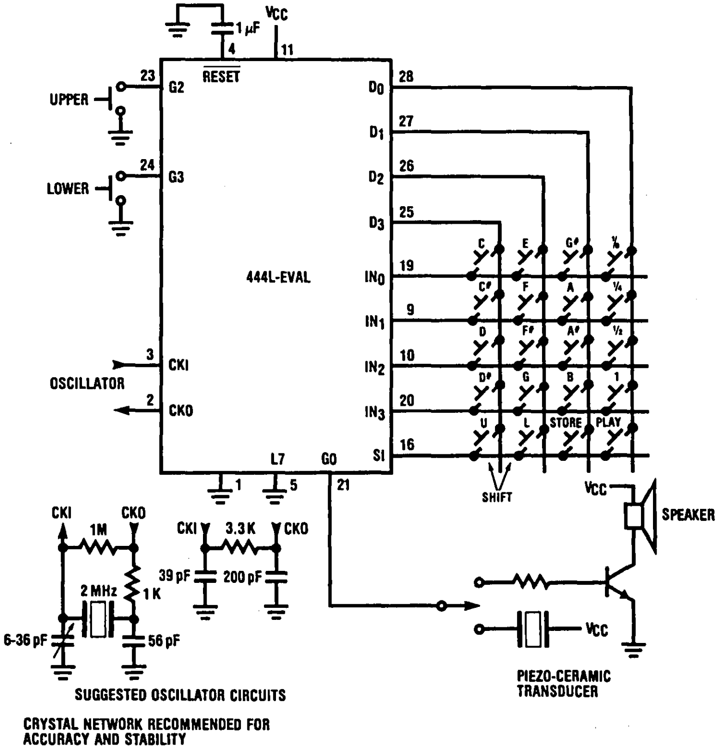
National Semiconductor
COP Note 4
Leonard A. Distaso

The 444L-EVAL is a preprogrammed COP444L intended to demonstrate operating characteristics and facilitate user familiarization and evaluation of the COP444L and the COPS™ family in general.

The 444L-EVAL has two mutually exclusive operating modes: an up/down counter/timer or a simple music synthesizer. The state of pin L7 at power up determines the operating mode.

1.0 THE 444L-EVAL AS A SIMPLE MUSIC SYNTHESIZER

Figure 1 indicates the connection of the 444L-EVAL as a simple music synthesizer. As the diagram indicates, the connections required for operation are minimal. The oscillator may be a crystal circuit using CKI and CKO; an external oscillator to CKI; or an RC network using CKI and CKO. As should be expected, the crystal circuit provides the greatest frequency stability and precision. The RC network will provide an acceptable oscillation frequency but that frequency will be neither precise nor stable over temperature and voltage. The external oscillator, of course, is as good as its source. The frequencies for the various notes and delay times are set up assuming that the oscillator frequency is 2 MHz. Three modes of operation are available in the music synthesizer mode: play a note; play one of four stored tunes; or record a tune for subsequent replay.



TL/DD/6937–1

FIGURE 1. 444L-EVAL as Simple Music Synthesizer

1.A. PLAY A NOTE

Twelve keys, representing the twelve notes in one octave, are labeled "C" through "B". Depressing a key causes a square wave of the corresponding frequency to output at GO. The user may drive a piezo-ceramic transducer directly with this signal. With the appropriate buffering, the user may use this signal to drive anything he wishes. A simple transistor driver is sufficient to drive a small speaker. The user can be as simple or as complex as he desires at this point—e.g. he can do some wave shaping, add an audio amplifier, and drive a high quality speaker.

The 444L-EVAL has a range of two and one-half octaves: the basic octave on the keyboard (which is middle C and the 11 notes above it in the chromatic scale), one full octave above the basic octave and one-half octave below the basic octave. The notes in the basic octave are played by depressing the appropriate key (one key at a time—the keyboard has no rollover provisions). A note in the upper octave is played by first depressing and releasing the U SHIFT key and then depressing the note key. Similarly, a note in the lower one-half octave is played by first depressing and releasing the L SHIFT key and then depressing the note key. Two other shift keys are present: UPPER and LOWER. All notes played while the UPPER key is held down will be in the upper octave. Similarly, note F# through B when played while the LOWER key is held down will be in the lower one-half octave. The lower octave notes C through F are not present and depressing any of these 6 keys while the LOWER key is held down or after depressing the L SHIFT key will play the note in the basic octave.

1.B. PLAY STORED TUNE

The 444L-EVAL can play four preprogrammed tunes. Depressing PLAY followed by "⅛", "¼", "½", or "1" will cause one of these tunes to be played. The tunes are:

PLAY 1 —Music Box Dancer
PLAY ½ —Santa Lucia
PLAY ¼ —Godfather Theme
PLAY ⅛ —Theme from Tchaikowsky Piano Concerto #1

1.C. RECORD A TUNE

Any combination of notes and rests up to a total of 48 may be stored in RAM for later replay. A note is stored by depressing the appropriate key(s), followed by the duration of the note (1⁄16 note, ⅛ note, 3⁄16 note, ¼ note, ⅜ note, ½ note, ¾ note, whole(1) note), followed by STORE. A rest is stored by selecting the duration and depressing STORE. The rests or durations of 1⁄16, 3⁄16, ⅜, and ¾ are obtained by first depressing L SHIFT and then ⅛, ¼, ½, or 1 respectively. When the tune is complete press PLAY followed by STORE. The tune will be played for immediate audition. Subsequent depression of PLAY and then STORE will play the last stored tune.

Only one tune may be stored, regardless of length. Attempts to store a new or second tune will erase the previously stored tune. There are no editing features in this mode. (In a "real system" of this type some form of editing would be desirable. It would not be difficult to add editing features.)

Note: The accuracy of the tones produced is a function of the oscillator accuracy and stability. The crystal oscillator, or an accurate, stable external oscillator is recommended.

2.0. THE 444L-EVAL AS AN UP/DOWN COUNTER/TIMER

By connecting pin L7 to V_{CC} and providing power and oscillator the 444L-EVAL functions as an 8 digit binary/BCD up/down counter. In addition, an approximate 1 Hz signal is produced by the device. The 444L-EVAL can drive a single digit LED display directly. With the appropriate driver (COP472, COP470, MM5450/5451) the device can drive a 4 digit LCD, VF, or LED display. Any combination of these displays can be connected at any given time.

The binary/BCD and and up/down modes are controlled by the states of input pins IN0 and IN2 as indicated below:

IN0 = 1 (Default state) —BCD counter
IN0 = 0 —Binary Counter
IN2 = 1 (Default state) —Count Up
IN2 = 0 —Count Down

The up/down control may be changed at any time. Changing the binary-BCD control during operation clears the counter before counting begins in the new mode.

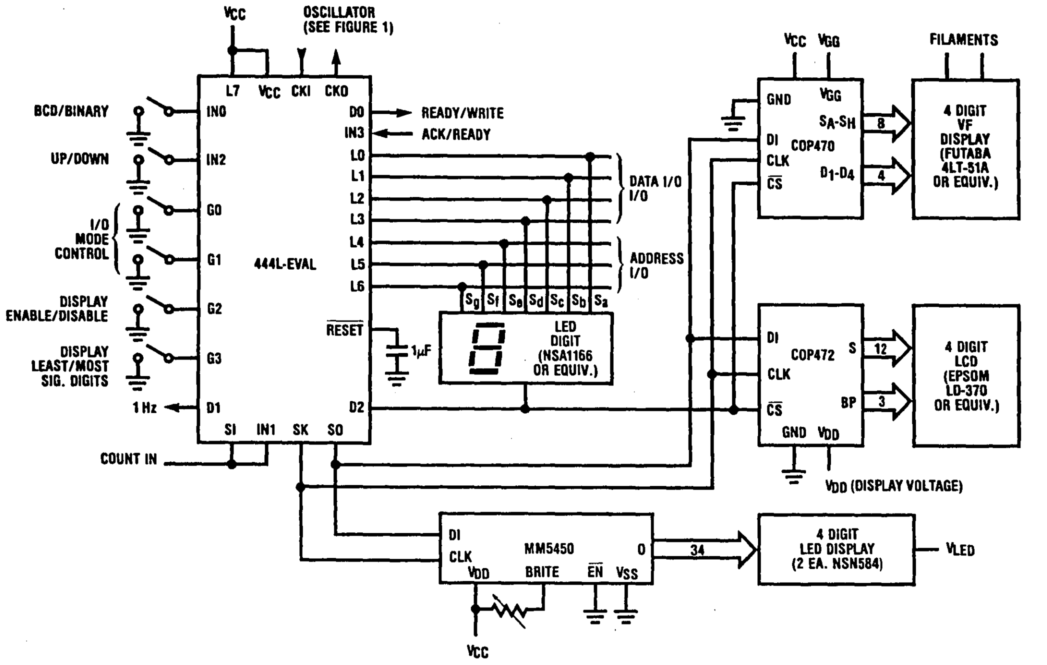
Pins G2 and G3 provide display control to the user. He can choose to view either the most significant 4 digits of the counter or the least significant 4 digits of the counter. Further, the user can disable the update of the 4 digit displays. The controls are as follows:

G2 = 1 (Default state) —Enable update of 4 digit displays
G2 = 0 —Disable update of 4 digit displays
G3 = 1 (Default state) —Display least significant 4 digits of counter
G3 = 0 —Display most significant 4 digits of counter

The single digit LED display displays the least significant digit of the counter. (Note, the direct drive capability for the single digit LED display refers to a small LED digit—NSA1541A, NSA1166k, or equivalent.)

2.A. I/O MODE

The 444L-EVAL has the capability to allow the user to read or write the 8 digit counter through the L port. In the I/O mode, the single digit LED display is disabled. The 4 digit displays are not affected. In this mode pins D0 and IN3 are used for the handshaking sequence. D0 is a Ready/Write signal from the 444L-EVAL to the outside; IN3 is a Write/Acknowledge from the outside to the 444L-EVAL. Data I/O is via L0–L3 with L0 being the least significant bit. Data is standard BCD for the BCD counter mode or standard hex for the binary counter mode. The digit address is on pins L4–L6 with L4 being the least significant bit. Digit address



TL/DD/6937-2

FIGURE 2. 444L-EVAL In Counter Mode

0 is the least significant digit of the counter; digit address 7 is the most significant digit of the counter. The I/O modes are controlled by pins G0 and G1 as follows:

G0	G1	
0	0	Output data with handshake, single digit LED off
0	1	Input data with handshake, single digit LED off
1	0	Auto output, no handshake, single digit LED on
1	1	Default condition, No I/O, single digit LED displays least significant digit of counter

2.A.1. Output Data with Handshake

With this mode selected the 444L-EVAL will output data with a handshake sequence. Note that the outputting of data is relatively slow as the device is counting and updating displays between successive digit outputs.

Before data is output, or the next digit of the counter is output, the 444L-EVAL must see IN3 (Acknowledge or ready from the external world high). The Ready/Write pin (D0) is assumed to be high at this point. With D0 high and IN3 high, the device will output the data and digit address. After the data and address are output, the D0 line—functioning as a write strobe here—goes low. The 444L-EVAL then expects the signal at IN3 to go low indicating that the external world has read the data. When the device sees IN3 go low, D0 will be brought high indicating that the sequence is ready to repeat as soon as IN3 goes high again. The counter digits are output sequentially from least significant digit (digit address 0) through most significant digit (digit address 7). The sequence will continuously repeat as long as this mode is selected.

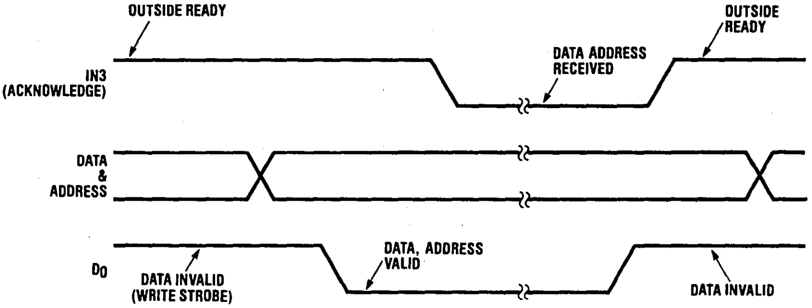
2.A.2. Input Data with Handshake

The 444L-EVAL will take data supplied to it and load the counter. The sequence is similar to that described above for the output mode. The external device(s) supplies both the data and the digit address where that data is to be loaded. When sending data to the 444L-EVAL, the external circuitry must test that the device is ready to receive data (D0 high). Then the data and address should be presented at the L port. Then the Write signal (IN3) should be driven low. The 444L-EVAL will read the data and then drive D0 low. When D0 goes low, the external circuitry should bring IN3 high. After IN3 returns high, the 444L-EVAL will signal it is ready to receive data by sending D0 high. Note that this sequence is relatively slow. The 444L-EVAL is performing several operations between successive read operations.

2.A.3. Automatic Output Mode

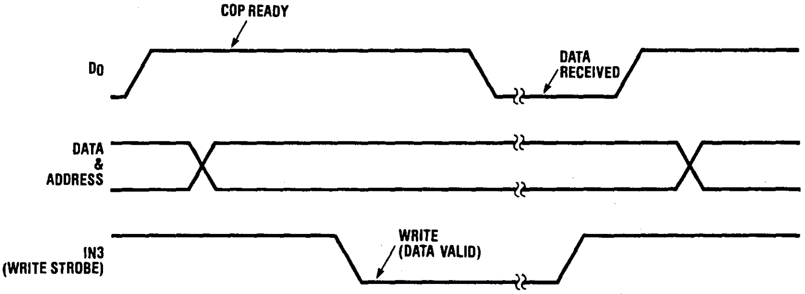
In the automatic output mode, the single digit LED is on. It is not displaying the least significant digit of the counter in this mode. The display is on so that the user can connect this LED digit, select the automatic output mode, and observe the states of the L lines without having to put more sophisticated equipment or circuitry external to the 444L-EVAL. Segments a through d are pins L0 thorugh L3; segments, e, f, g are pins L4, L5, and L6. Thus the user can observe the digit address changing and observe the corresponding data.

In this mode, the state of pin IN3 is irrelevant. The 444L-EVAL sequentially outputs the digits of the counter. D0 goes high when the data and address is being changed. D0 goes low when the data is valid. As in the other I/O modes, the process is slow. There is about 4 to 5 milliseconds between the successive digit outputs.



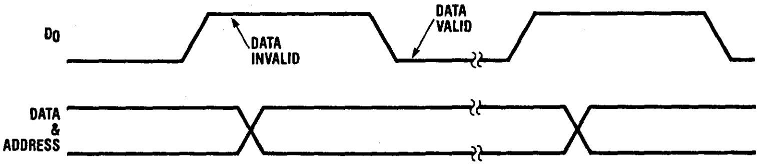
TL/DD/6937-3

FIGURE 3A. Relative Timing—Output Handshake



TL/DD/6937-4

FIGURE 3B. Relative Timing—Input Handshake



TL/DD/6937-5

FIGURE 3C. Relative Timing—Automatic Output

3.0 SELECTED OPTIONS

The 444L-EVAL has the following options selected:

GND	Option 1	= 0	
CKO	Option 2	= 0	CKO is clock generator output to crystal
CKI	Option 3	= 0	CKI oscillator input divide by 32
RESET	Option 4	= 0	Load device to V_{CC} on RESET
L7	Option 5	= 0	Standard output on L7
L6	Option 6	= 2	High current LED direct segment drive on L6
L5	Option 7	= 2	High current LED direct segment drive on L5
L4	Option 8	= 2	High current LED direct segment drive on L4
IN1	Option 9	= 0	Load device to V_{CC} on IN1
IN2	Option 10	= 0	Load device to V_{CC} on IN2
V_{CC}	Option 11	= 1	4.5V to 9.5V operation
L3	Option 12	= 2	High current LED direct segment drive on L3
L2	Option 13	= 2	High current LED direct segment drive on L2
L1	Option 14	= 2	High current LED direct segment drive on L1
L0	Option 15	= 2	High current LED direct segment drive on L0
SI	Option 16	= 0	Load device to V_{CC} on SI
SO	Option 17	= 2	Push-pull output on SO
SK	Option 18	= 2	Push-pull output on SK
IN0	Option 19	= 0	Load device to V_{CC} on IN0
IN3	Option 20	= 0	Load device to V_{CC} on IN3
G0	Option 21	= 0	Very high current standard output on G0
G1	Option 22	= 2	High current standard output on G1
G2	Option 23	= 4	Standard LSTTL output on G2
G3	Option 24	= 4	Standard LSTTL output on G3
D3	Option 25	= 0	Very high current standard output on D3
D2	Option 26	= 0	Very high current standard output on D2
D1	Option 27	= 0	Very high current standard output on D1
D0	Option 28	= 0	Very high current standard output on D0
	Option 29	= 0	Standard TTL input levels on L
	Option 30	= 0	Standard TTL input levels on IN
	Option 31	= 0	Standard TTL input levels on G
	Option 32	= 0	Standard TTL input levels on SI
	Option 33	= 1	Schmitt trigger inputs on RESET
	Option 34	= 0	CKO input levels, not used here
	Option 35	= 0	COP444L
	Option 36	= 0	Normal RESET operation

4.0 CONCLUSION

The 444L-EVAL demonstrates much of the capability of the COP444L. It does not indicate the limits of the device by any means. The I/O features were included to demonstrate that capability. The fact that they are slow is due strictly to the program. If such I/O capability were a necessary part of an application it could be accomplished much much faster than was done here. The counter modes are quite versatile and are generally self explanatory. It was fairly easy to provide a counter with the versatility of that included here. The music synthesis mode demonstrates clearly the program efficiency of the device.

The 444L-EVAL is intended for demonstration. There is no question that aspects of its operation could be improved and tailored to a specific application. It is unlikely that this particular combination of features would be found in any one application. It is also interesting to note that the program memory in the device is not full. There is still a significant amount of room left in the ROM. This should serve to make it clear that the capabilities of the device have not been stretched at all in order to include these demonstration functions.

Oscillator Characteristics of COPS™ Microcontrollers

National Semiconductor
COP Note 5

Table of Contents

1.0 INTRODUCTION

COPS microcontrollers will operate with a wide variety of oscillator circuits. This paper focuses on two of the oscillator options available on COPS microcontrollers: the internal RC oscillator, and the crystal or inverter oscillator. The typical behavior of the RC oscillator with temperature and voltage (and typical values of R and C) is documented. For the crystal or inverter option, circuit configurations (RC, RL, RLC, R, LC, L) are presented which will allow the microcontroller to operate properly without the use of ceramic resonator or crystal.

The passive components used were inexpensive, uncompensated devices: standard carbon resistors, ceramic or foil capacitors, and air core or iron core inductors. To provide reasonably clear data on the characteristics of the microcontroller itself, no attempt at compensation for the external components was made.

2.0 RC OSCILLATOR OPTION

With the RC oscillator option selected, the graphs in *Figures 1* through *6* indicate the variation of the instruction cycle time of the microcontroller with temperature and voltage. Typical R and C values, as recommended in the respective device data sheets, were used. The graphs are composite graphs reflecting the worst case variations of the devices tested. Therefore, the graphs show a percentage change of the instruction cycle time from a base or reference value. Where the results are plotted against voltage the reference is the value at $V_{CC} = 5V$. Where the results are plotted against temperature, the reference is the value at $T = 20°C$. A positive percent variation indicates a longer instruction cycle time and therefore a slower oscillator frequency. Similarly, a negative percent variation indicates a shorter instruction cycle time and therefore a faster oscillator frequency.

The measurements were taken by holding the RESET pin of the device low and measuring the period of the waveform at pin SK. In this mode the SK period is the instruction cycle time. For divide by 4 the oscillator frequency is given by the following:

$$\text{frequency} = \frac{4}{\text{SK period}}$$

Measurements were taken at temperatures between −40°C and +85°C and at V_{CC} values between 4.5V and 9.5V. However, the reader must remember that the COP400 series is specified only between 0°C and +70°C. The reader must also remember that the COP420 is specified at V_{CC} levels between 4.5V and 6.3V only. The data here is usable for the COP300 series, which is specified at the extended temperature range of −40°C to +85°C. However, the reader must keep in mind the generally more restricted V_{CC} range for some of the various COP300 series microcontrollers.

The graphs in *Figures 1* through *6* reflect the variation of the microcontroller only. The resistor and capacitor were not in the temperature chamber with the COPS device. Obviously, the results will be affected by the variation of the R and C with temperature. However, this can vary dramatically with the type of components used. The user will have to combine the data here with the characteristics of the external components used to determine what type of variation may be expected in his system.

3.0 CRYSTAL OR INVERTER OPTION

With the crystal or inverter option selected on the COPS microcontroller there is, effectively, an inverter between the CKI and CKO pins. CKI is the input to the inverter and CKO is the output. Various passive circuits were connected between CKI and CKO and the results documented. Of the operational circuits, a subset was tested over temperature with the microcontroller only in the temperature chamber. A smaller subset was tested over temperature with both the microcontroller and the oscillator network in the temperature chamber.

The data with the oscillator network in the temperature chamber is obviously highly dependent on the particular components used. This data was taken with standard, inexpensive, uncompensated components. Neither high precision nor high stability components were used. This data is included only to provide the user with some very general indication of how the oscillator frequency may vary with temperature in a real system.

3.1 COP420/COP402

Except for the ROM, the COP420 and COP402 are equivalent devices. The internal circuitry of each device is identical. Therefore, data taken for one of the devices is equally applicable to the other. The following discussion will refer to the COP420 but all such references apply equally well to the COP402. Similarly, the graphs for the COP420 apply to the COP402 and *vice versa.*

With the crystal option selected, the COP420 oscillator circuitry will readily oscillate with almost any circuit configuration between CKI and CKO. What difficulty there is lies in finding the network of the device. With the appropriate divide option selected, oscillator frequencies between 800 kHz and 4 MHz are valid for the COP420. No data was taken for any network that produced an oscillation frequency outside the valid range.

3.1.1 L, LC, and RLC Networks

Various L, LC, and RLC networks were connected with varying results. Certain networks produced results much more stable than the RC networks; others were no better than the RC networks. With a single inductor connected between CKI and CKO, frequencies between 1 MHz and 4 MHz were easily obtained. However, the input gate capacitance at CKI (typically 5 pF to 10 pF) and the series resistance of the inductance become factors that impact the oscillation frequency and its stability over temperature.

The addition of a capacitor between CKI and ground tends to reduce the effects of the internal gate capacitance. For the single L, single C network of this type, the capacitor value should be greater than about 50 pF to begin to effectively swamp out the effects of the input gate capacitance. As might be expected, LC combinations which had their resonant frequencies within the valid COP420 frequency range produced the best results.

The addition of another capacitor(s) to the basic two-component LC network, as shown in *Figure III.1*, produced very good results. Varying the capacitor values in these networks — especially those capacitors between CKI and ground and CKO and ground — provided a great deal of control over the oscillation frequency. In *Figure III.1*, varying C1 from 25 pF to 0.01 μF produced oscillation frequencies between about 3 MHz and 1.6 MHz (C2 = 25 pF, L = 56 μH). In *Figure III.2*, with C1 = 330 pF, L = 56 μH, and C2 = 27 pF, varying C3 between 10 pF and 0.003 μF produced oscillation frequencies between about 2 MHz and 1.1 MHz. Varying C2 in *Figure 111.3* produced a similar kind of control.

As the graphs indicate, various types of RLC networks were also tried. The range of possible usable circuits here is limited only by the user's imagination and his favorite type of RLC oscillator circuit. When their resonant frequency is within the valid frequency range of the COP420, LC and RLC networks can be a very effective substitute for a crystal. The only potential problem is that a good RLC, or even LC, oscillator circuit may not be a cost-effective substitute for a crystal in a COP420 system. The user will have to make that determination.

3.2 COP420L

The valid input frequency range for the COP420L, with the appropriate divide option selected, is between 200 kHz and 2.097 MHz. With the crystal option selected the COP420L oscillated much less readily than the COP420.

The LC networks gave outstanding results with the COP420L. With the simple two-component LC network shown in the graphs, holding C at 50 pF and varying L from 200 μH to 700 μH gave oscillation frequencies from about 2 MHz to 1 MHz. Holding L at 390 μH and varying C from 10 pF to 700 pF gave oscillation frequencies of about 2 MHz to 1.6 MHz. Similar results were obtained when a capacitor was placed in parallel with the inductance.

3.3 COP410L

The COP410L has a valid input frequency range of 200 kHz to 530 kHz.

The LC networks also gave very good results. With the simple LC network shown in the graphs, holding L at 4700 μH and varying C from 25 pF to 0.003 μF gave oscillation frequencies of about 460 kHz to 225 kHz.

3.4 GENERAL NOTES

With the crystal or inverter option selected on COPS microcontrollers, a wide variety of networks may be used in place of the ceramic resonator or crystal.

LC and RLC networks can be used in any of the devices. Appropriately designed, these networks will provide a stable oscillation frequency for the microcontroller. The user will have to allow for the variation of the external components with temperature when using these networks. The problems with networks such as these is that they may not be cost-effective alternatives to the crystal or resonator, especially if high stability, temperature compensated components are used. The user will have to make the determination of cost-effectiveness.

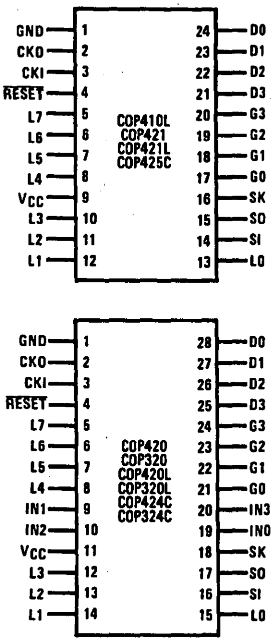
A final note is that all of these networks place a load on the CKO output. If the signal from CKO is needed elsewhere in the system and a circuit similar to one of those discussed in this document is used, it will probably be necessary to buffer the CKO output.

4.0 Conclusion

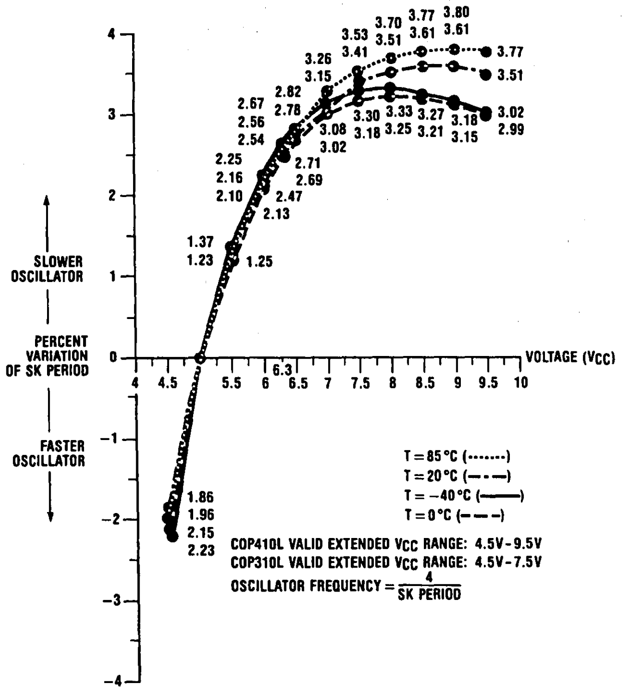
The networks described are generally simple and inexpensive and have all been observed to be functional.

The results obtained provide greater flexibility in the oscillator selection in a COPs system and gives the user some general indication as to what may be expected with the various circuits described.

COP Microcontroller Pinouts



TL/DD/6938-1



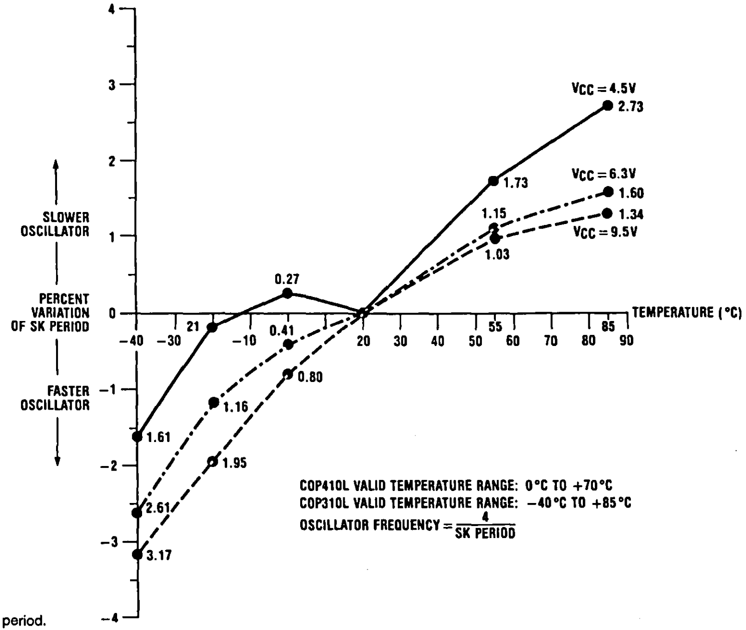
TL/DD/6938-2

Note 1: Base period at V_{CC} = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 1. COP310L/COP410L RC Oscillator Variation with V_{CC}



COP410L VALID TEMPERATURE RANGE: 0°C TO +70°C
COP310L VALID TEMPERATURE RANGE: −40°C TO +85°C

OSCILLATOR FREQUENCY = $\frac{4}{\text{SK PERIOD}}$

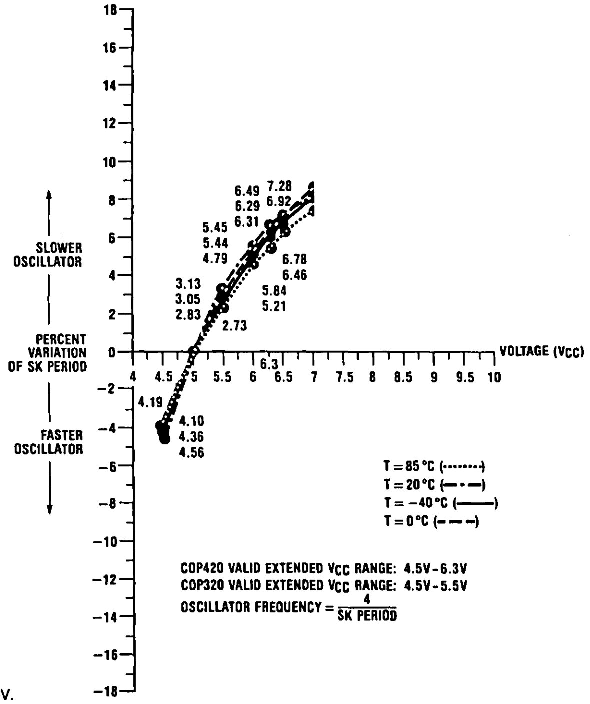
TL/DD/6938-3

Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 2. COP310L/COP410L RC Oscillator Variation with Temperature



COP420 VALID EXTENDED V_{CC} RANGE: 4.5V-6.3V
COP320 VALID EXTENDED V_{CC} RANGE: 4.5V-5.5V

OSCILLATOR FREQUENCY = $\frac{4}{\text{SK PERIOD}}$

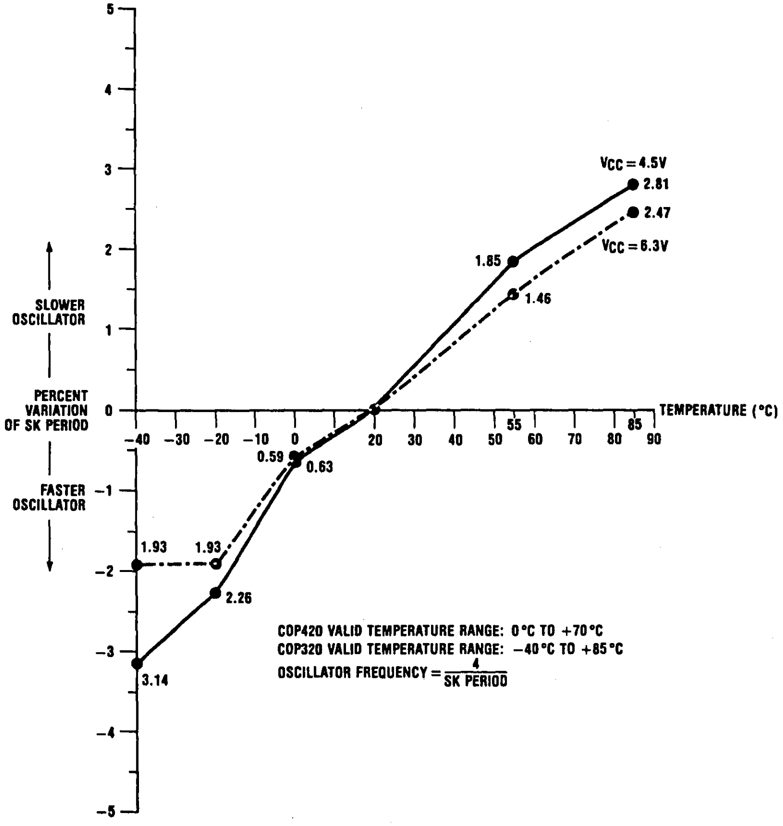
TL/DD/6938-4

Note 1: Base period at V_{CC} = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 3. COP320/COP420 RC Oscillator Variation with V_{CC}



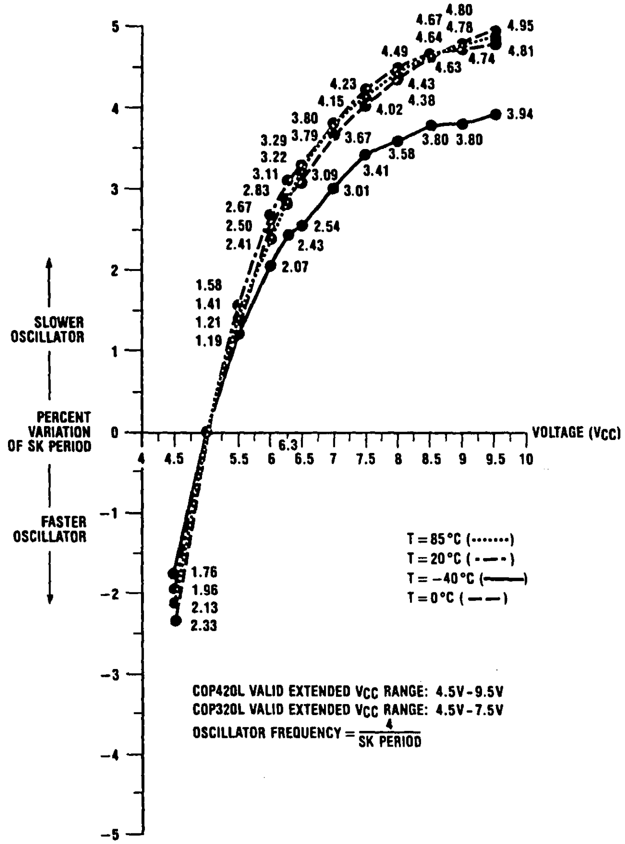
TL/DD/6938-5

Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 4. COP320/COP420 RC Oscillator Variation with Temperature



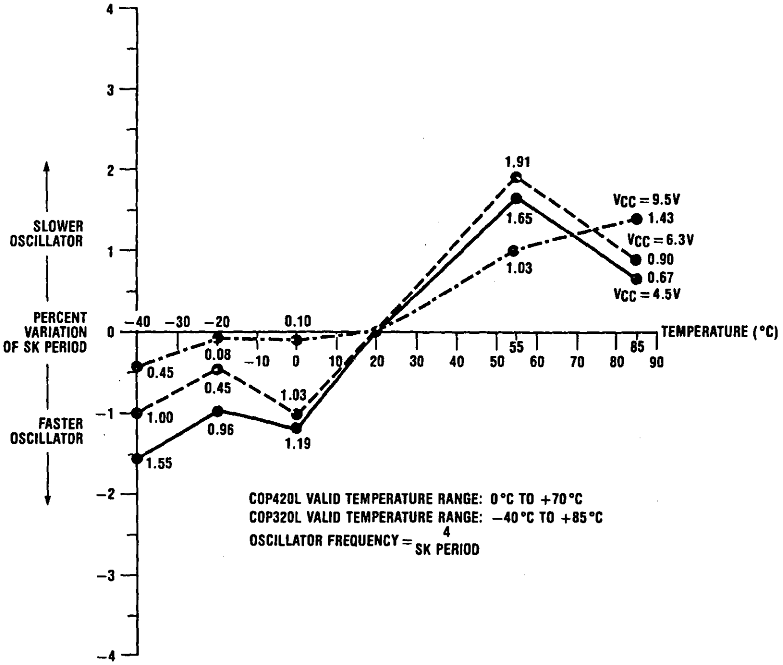
TL/DD/6938-6

Note 1: Base period at V_{CC} = 5.0V.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 5. COP320L/COP420L RC Oscillator Variation with V_{CC}



TL/DD/6938-7

Note 1: 20°C = base period.

Note 2: Device variation only. Graph does not include RC variation with temperature.

Note 3: SK period = instruction cycle time.

FIGURE 6. COP320L/COP420L RC Oscillator Variation with Temperature

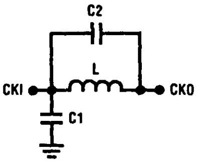


FIGURE III.1

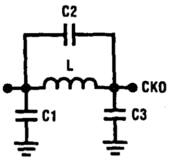


FIGURE III.2

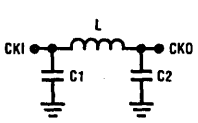
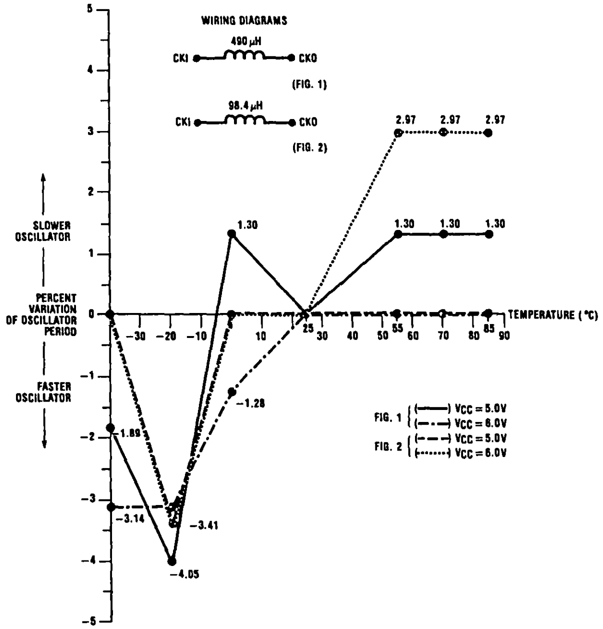


FIGURE III.3

TL/DD/6938-8

COP402



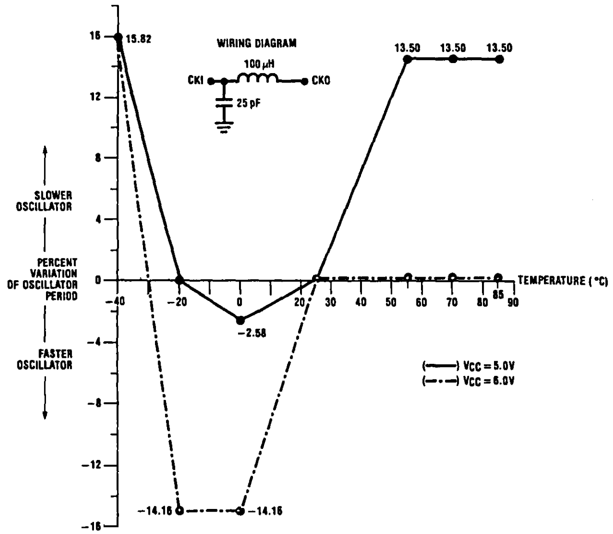
Note 1: 25°C = base period.

TL/DD/6938-9

Note 2: Device variation only. Graph does not include "L" variation with temperature.

FIGURE 7

COP402



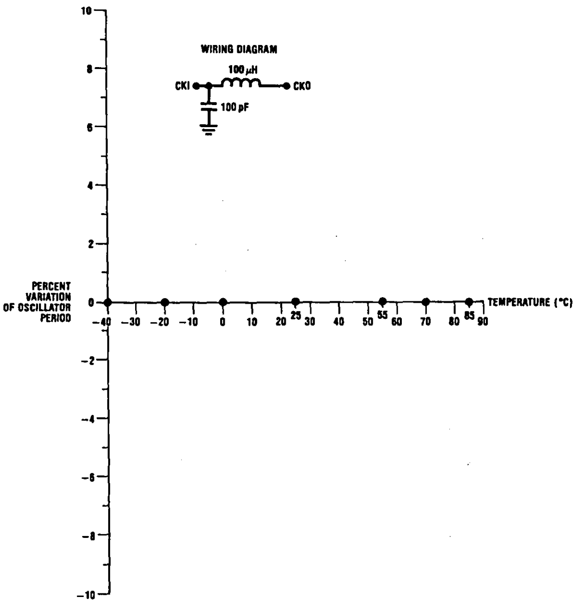
Note 1: 25°C = base period.

TL/DD/6938-10

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 8

COP420



TL/DD/6938-11

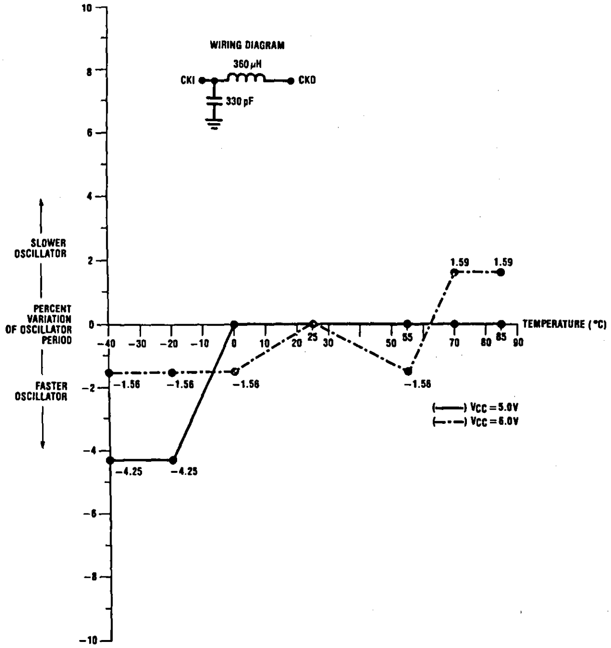
Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

No measurable variation over temperature.

FIGURE 9

COP420



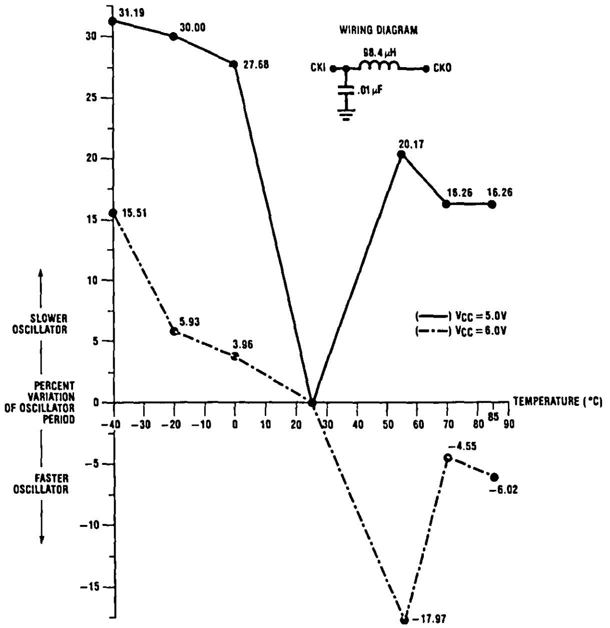
TL/DD/6938-12

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 10

COP402



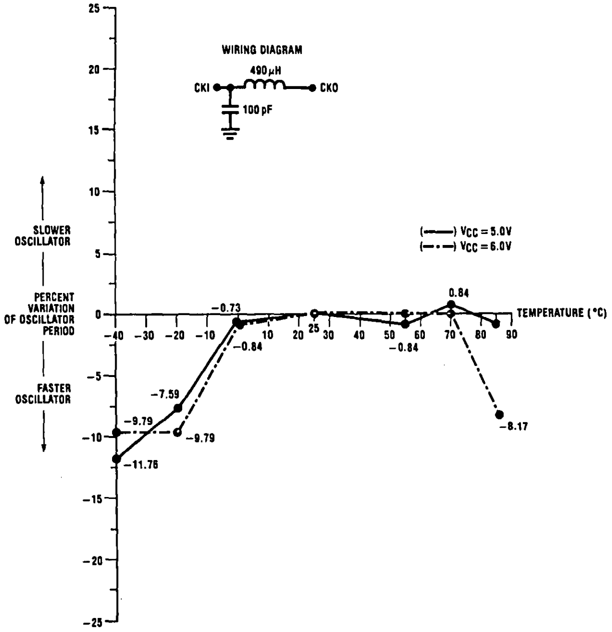
TL/DD/6938-13

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 11

COP402



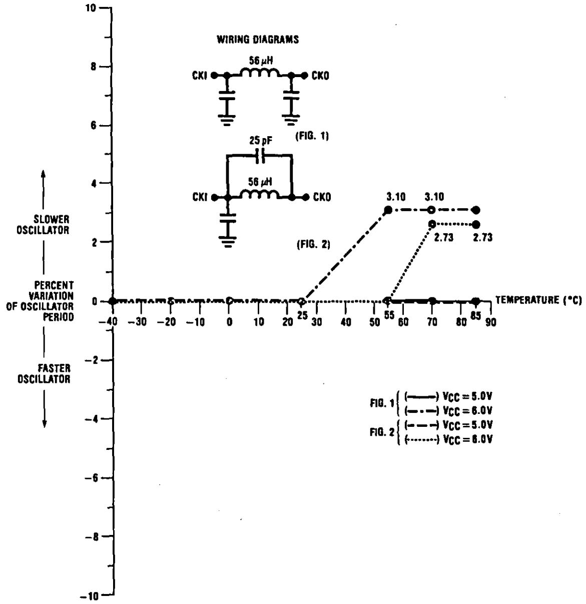
TL/DD/6938-14

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 12

COP402



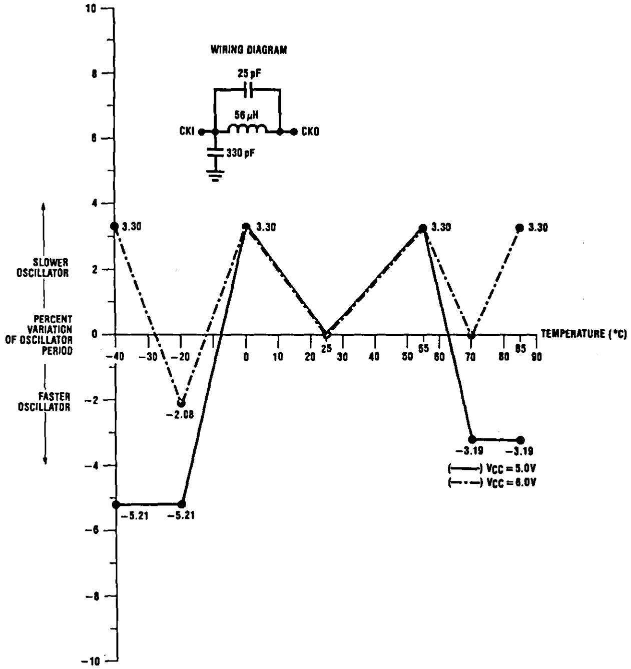
TL/DD/6938-15

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 13

COP402

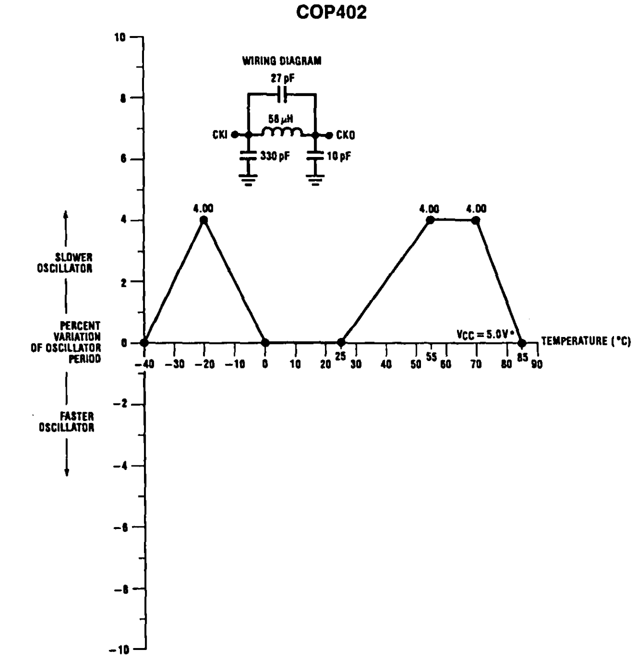


TL/DD/6938-16

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 14



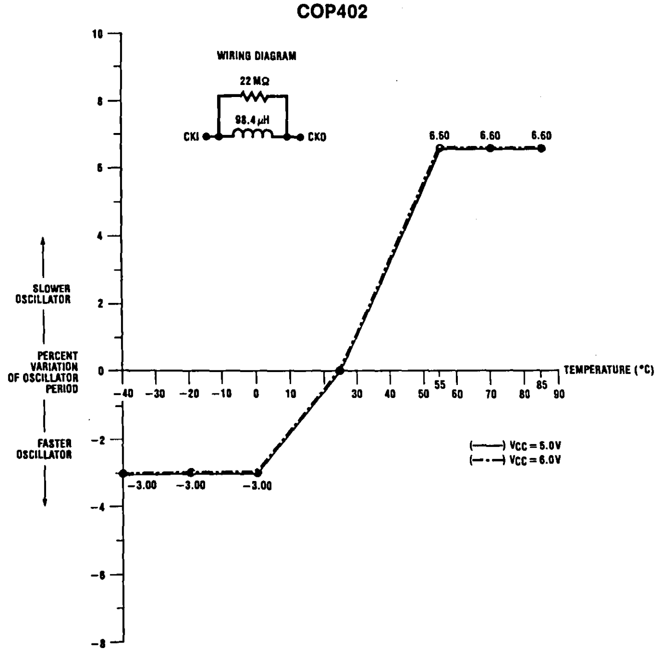
TL/DD/6938-17

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

*No variation at 6V.

FIGURE 15



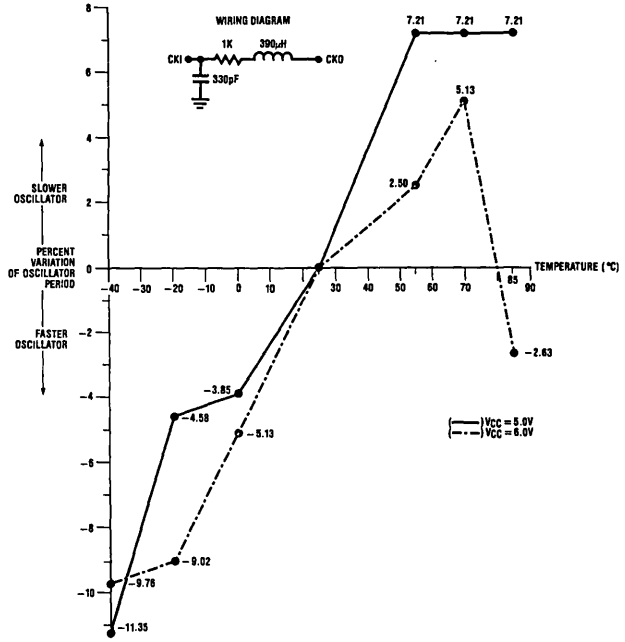
TL/DD/6938-18

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RL variation with temperature.

FIGURE 16

COP402



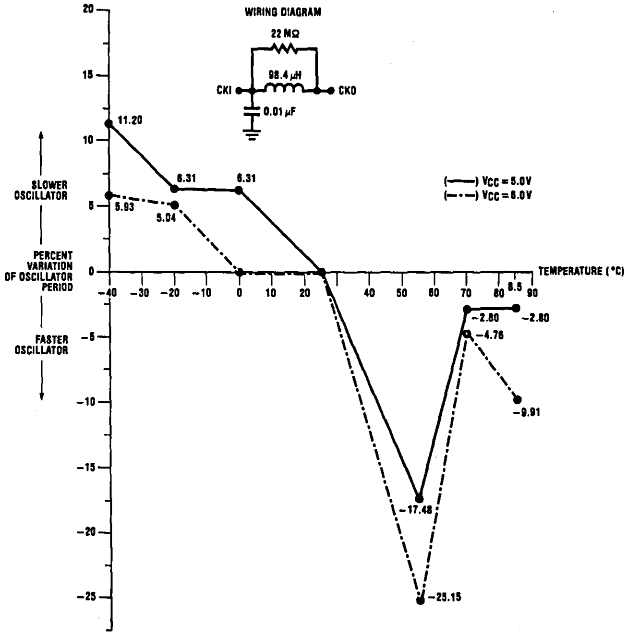
TL/DD/6938-19

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 17

COP402



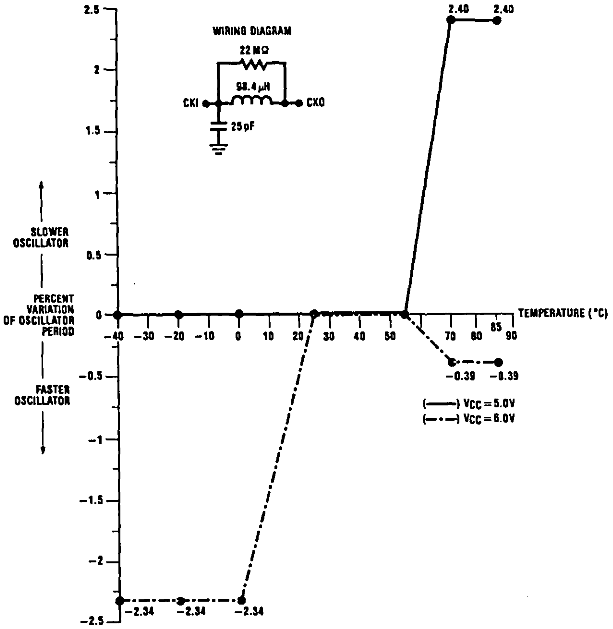
TL/DD/6938-20

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 18

COP402



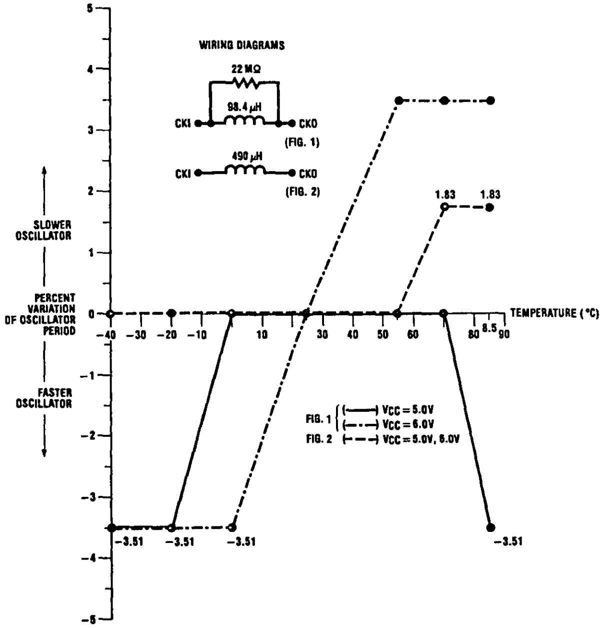
TL/DD/6938-21

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include RLC variation with temperature.

FIGURE 19

COP402



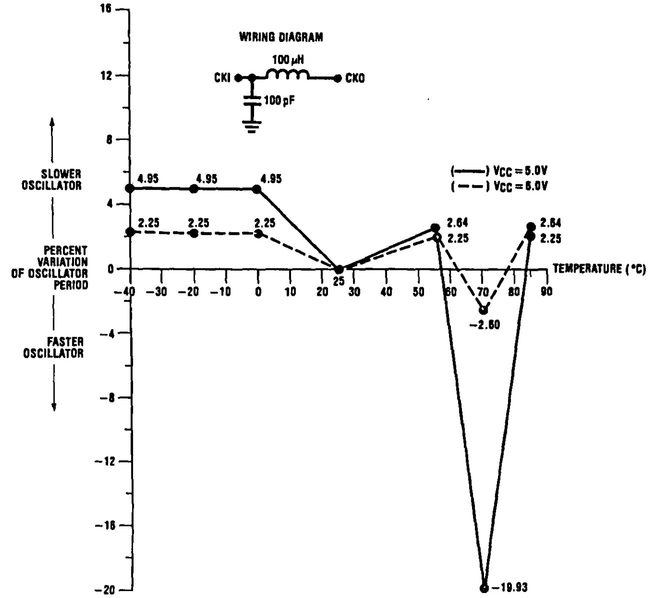
TL/DD/6938-22

Note 1: 25°C = base period.

Note 2: RL in oven with COP402.

FIGURE 20

COP420



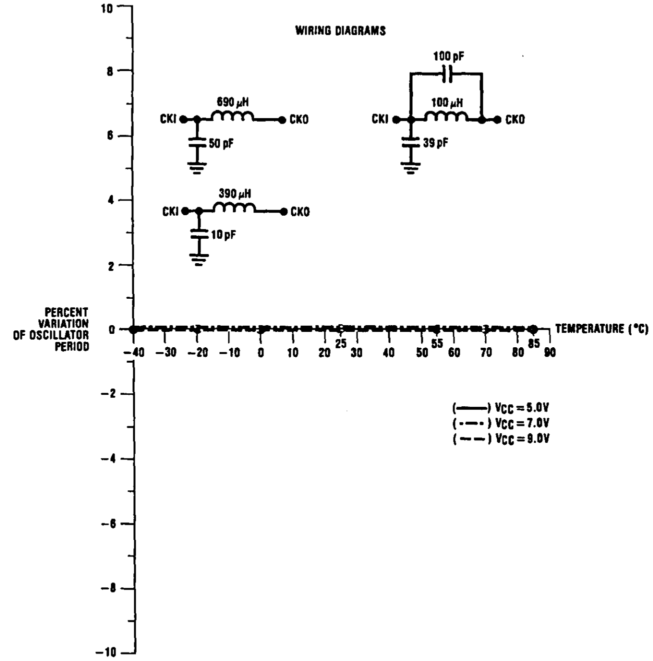
Note 1: 25°C = base period.

Note 2: LC in oven with COP402.

TL/DD/6938-23

FIGURE 21

COP420L



TL/DD/6938-26

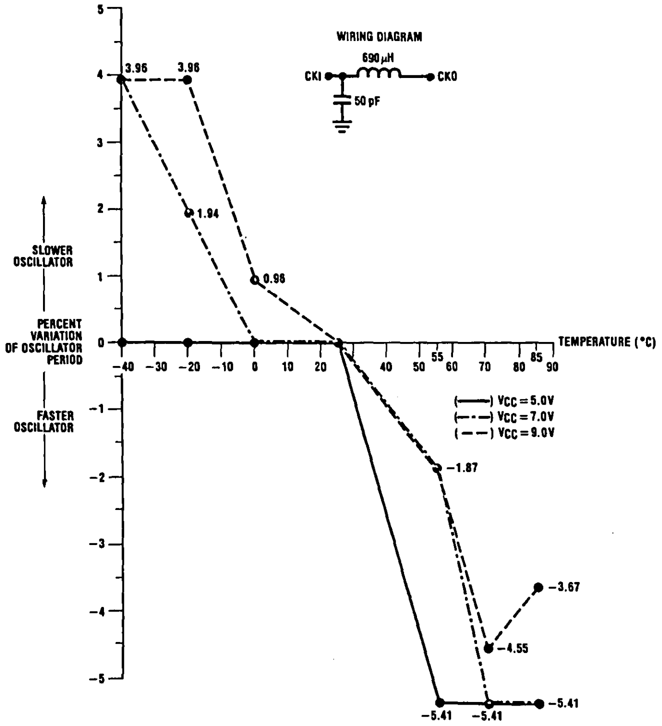
Note 1: No measurable variation for all three circuits above.

Note 2: 25°C = base period.

Note 3: Device variation only. Graph does not include LC variation with temperature.

FIGURE 22

COP420L



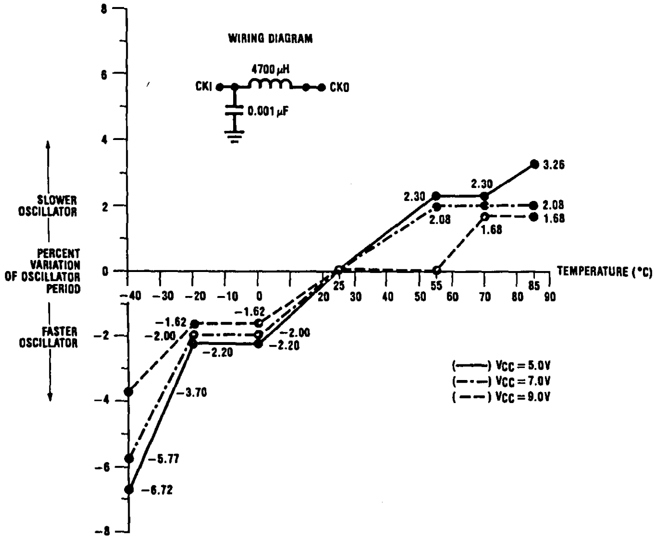
TL/DD/6938-28

Note 1: 25°C = base period.

Note 2: LC in oven with COP420L.

FIGURE 23

COP410L



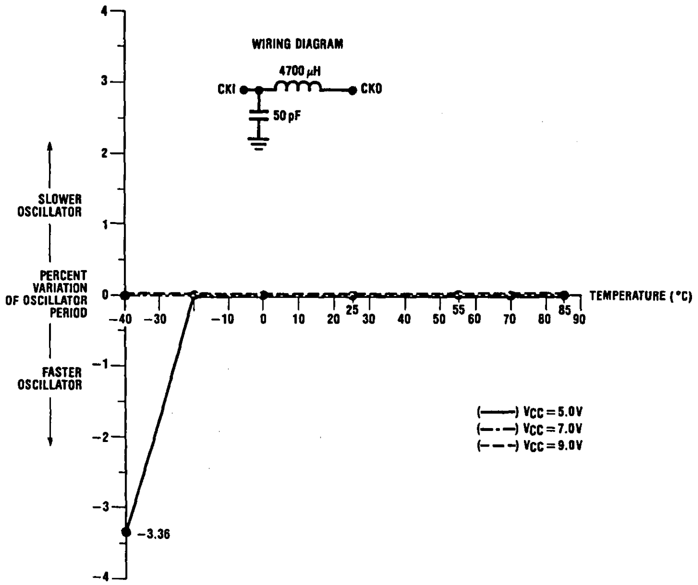
TL/DD/6938-29

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 24

COP410L



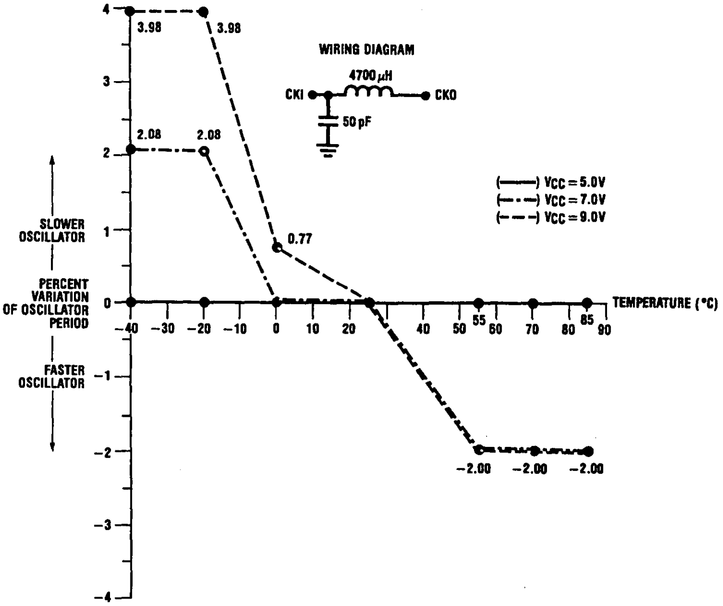
TL/DD/6938-30

Note 1: 25°C = base period.

Note 2: Device variation only. Graph does not include LC variation with temperature.

FIGURE 25

COP410L



TL/DD/6938-32

Note 1: 25°C = base period.

Note 2: LC in oven with COP410L.

FIGURE 26

Triac Control Using the COP400 Microcontroller Family

National Semiconductor
COP Note 6

Table of Contents

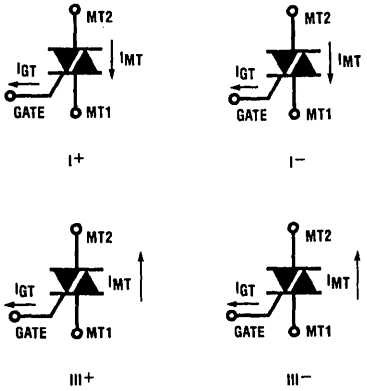
1.0 Triac Control

The COP400 single-chip controller family members provide computational ability and speed which is more than adequate to intelligently manage power control. These controllers provide digital control while low cost and short turnaround enhance COPS™ desirability. The COPS controllers are capable of 4 μs cycle times which can provide more than adequate computational ability when controlling 60 Hz line voltage. Input and output options available on the COPS devices can contour the device to apply in many electrical situations. A more detailed description of COPS qualifications is available in the COP400 data sheets.

The COPS controller family may be utilized to manage power in many ways. This paper is devoted to the investigation of low cost triac interfaces with the COP400 family microcontroller and software techniques for power control applications.

1.1 BASIC TRIAC OPERATION

A triac is basically a bidirectional switch which can be used to control AC power. In the high-impedance state, the triac blocks the principal voltage across the main terminals. By pulsing the gate or applying a steady state gate signal, the triac may be triggered into a low impedance state where conduction across the main terminals will occur. The gate signal polarity need not follow the main terminal polarity; however, this does affect the gate current requirements. Gate current requirements vary depending on the direction of the main terminal current and the gate current. The four trigger modes are illustrated in *Figure 1*.



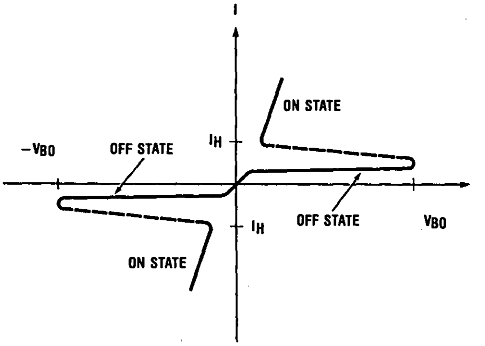
I+ I− III+ III−

TL/DD/6939–1

FIGURE 1. Gate Trigger Modes. Polarities Referenced to Main Terminal 1.

The breakover voltage (V_{BO}) is specified with the gate current (I_{GT}) equal to zero. By increasing the gate current supplied to the triac, V_{BO} can be reduced to cause the triac to go into the conduction or on state. Once the triac has entered the on state the gate signal need not be present to sustain conduction. The triac will turn itself off when the main terminal current falls below the minimum holding current required to sustain conduction (I_H).

A typical current and voltage characteristic curve is given in *Figure 2*. As can be seen, when the gate voltage and the main terminal 2 (MT2) voltages are positive with respect to MT1 the triac will operate in quadrant 1. In this case the trigger circuit sources current to the triac (I+ MODE).



TL/DD/6939-2

FIGURE 2. Voltage-Current Characteristics

After conduction occurs the main terminal current is independent of the gate current; however, due to the structure of the triac the gate trigger current is dependent on the direction of the main terminal current. The gate current requirements vary from mode to mode. In general, a triac is more easily triggered when the gate current is in the same direction as the main terminal current. This can be illustrated in the situation where there is not sufficient gate drive to cause conduction when MT2 is both positive and negative. In this case the triac may act as a single direction SCR and conduction occurs in only one direction. The trigger circuit must be designed to provide trigger currents for the worst case trigger situation. Another reason ample trigger current must be supplied is to prevent localized heating within the pellet and speed up turn-on time. If the triac is barely triggered only a small portion of the junction will begin to conduct, thus causing localized heating and slower turn-on. If an insufficient gate pulse is applied damage to the triac may result.

1.2 TRIGGERING

Gate triggering signals should exceed the minimum rated trigger requirements as specified by the manufacturer. This is essential to guarantee rapid turn-on time and consistent operation from device to device.

Triac turn-on time is primarily dependent on the magnitude of the applied gate signal. To obtain decreased turn-on times a sufficiently large gate signal should be applied. Faster turn-on time eliminates localized heat spots within the pellet structure and increases triac dependability.

Digital logic circuits, without large buffers, may not have the drive capabilities to efficiently turn on a triac. To insure proper operation in all firing situations, external trigger circuitry might become necessary. Also, to prevent noise from disturbing the logic levels, AC/DC isolation or coupling techniques must be utilized. Sensitive gate triacs which require minimal gate input signal and provide a limited amount of main terminal current may be driven directly. This paper will focus on $120V_{AC}$ applications of power control.

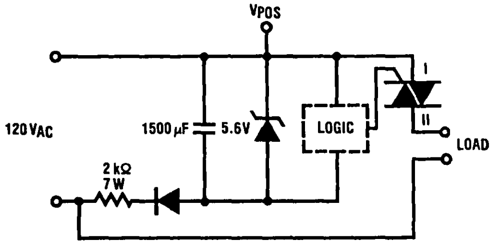
1.3 ZERO VOLTAGE DETECTION

In many applications it is advantageous to switch power at the AC line zero voltage crossing. In doing this, the device being controlled is not subjected to inherent AC transients. By utilizing this technique, greater dependability can be obtained from the switching device and the device being switched. It is also sometimes desirable to reference an event on a cyclic basis corresponding to the AC line frequency. Depending on the load characteristics, switching times need to be chosen carefully to insure optimal performance. Triac controlled AC switching referenced to the AC 60 Hz line frequency enables precise control over the conduction angle at which the triac is fired. This enables the COPS device to control the power output by increasing or decreasing the conduction angle in each half cycle.

A wide variety of zero voltage detection circuits are available in various levels of sophistication. COPS devices, in most cases, can compensate for noisy or semi-accurate ZVD circuits. This compensation is utilized in the form of debounce and delay routines. If a noisy transition occurs near zero volts the COPS device can wait for a valid transition period specified by the maximum amount of noise present. Some software considerations are presented in the software section and are commented upon. The minimal detection circuit is shown in *Figure 9*.

1.4 DIRECT COUPLE

Isolation associated problems can be overcome by means of direct AC coupling. One such method is illustrated in *Figure 3*. This circuit incorporates a half-wave rectifier in conjunction with a filter capacitor to provide the logic power supply. The positive half-cycle is allowed to drop across the zener diode and be filtered by the capacitor. This creates a low cost line interface; however, only a limited supply current is available. In order to control the current capabilities of this circuit the series resistor must be modified. However, as more current is required, the power that must be dissipated in the series resistor increases. This increases the power dissipation requirements of the series resistor and the system cost. For applications which require large current sources an alternative method is advisable. In order to assure consistent operation, power supply ripple must be minimized. COPS devices can be operated over a relatively wide power supply range. However, excessive ripple may cause an inadvertent reset operation of the device.

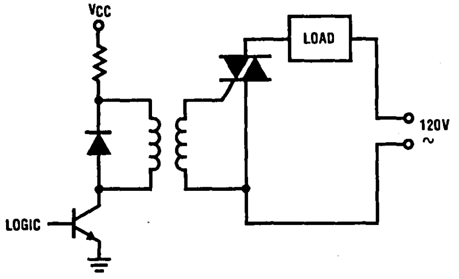


TL/DD/6939-3

FIGURE 3. AC Direct Couple

1.5 PULSE TRANSFORMER INTERFACE

Digital logic control of triacs is easily accomplished by triggering through pulse transformers or optical coupling. The energy step-up gained by using a pulse transformer should provide a more than adequate gate trigger signal. This complies with manufacturers' suggested gate signal requirements. Pulse transformers also provide AC/DC isolation necessary in control logic interfaces. Minimal circuit interface to the pulse transformer is required as shown in *Figure 4*. Optical coupling circuits provide isolation, and in some cases adequate gate drive capabilities.



TL/DD/6939-4

FIGURE 4. Pulse Transformer Interface

A logic controlled pulse is applied to the base of the transistor to switch current through the primary of the pulse transformer. The transformer then transfers the signal to the secondary and causes the triac to fire. The energy transfer that is now available on the secondary is more than adequate to turn on the triac in any of its operating modes. When the pulse transformer is switched off a reverse EMF is generated in the primary coil which may cause damage to the transistor. The diode across the primary serves to protect the collector junction of the switching transistor. Another major advantage is AC isolation; the gate of the triac is now completely isolated from the logic portion of the circuit.

1.6 FALSE TURN-ON

When switching an inductive load, voltage spikes may be generated across the main terminals of the triac which have the potential of a non-gated turn-on of the triac. This creates the undesirable situation of limited control of the system. In a system with an inductive load the voltage leads the current by a phase shift corresponding to the amount of inductance in the motor. As the current passes near zero, the voltage is at a non-zero value, offset due to the phase shift. When the principal current through the triac pellet decreases to a value not capable of sustaining conduction the triac will turn off. At this point in time the voltage across the terminals will instantaneously attain a value corresponding to the phase shift caused by the inductive load. The rapid decay of current in the inductor causes an L dI/dT voltage applied across the terminals of the triac. Should this voltage exceed the blocking voltage specified for the triac, a false turn-on will occur.

In order to avoid false turn-on, a snubber network must be added across the terminals to absorb the excess energy generated by this situation. A common form of this network is a simple RC in series across the terminals. In order to select the values of the network it is necessary to determine the peak voltage allowable in the system and the maximum dV/dT stress the triac can withstand. One approach to obtaining the optimal values for R_S and C_S is to model the effective circuit and solve for the triac voltage. The snubber in conjunction with the load can now be modeled as an RLC network. Due to the two storage elements (L motor, C snubber) a second order differential equation is generated. Rather than approach this problem from a computer standpoint it becomes much easier to obtain design curves generated for rapid solution of this problem. These design curves are available in many triac publications. (For instance, see RCA application note AN 4745.)

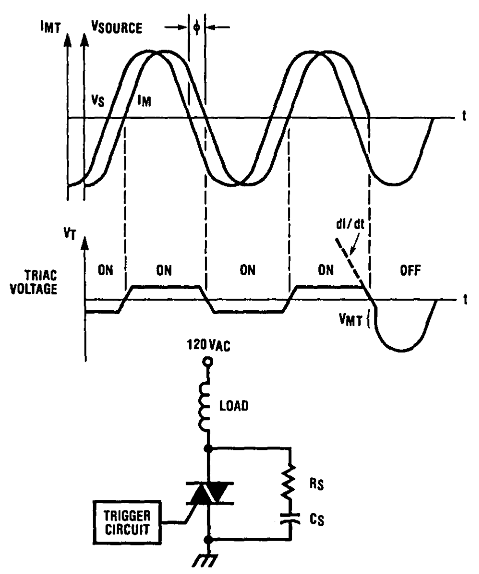
2.0 Software Techniques

2.1 ZERO VOLTAGE DETECTION

In order to intelligently control triacs on a cyclic basis, an accurate time base must be defined. This may be in the form of an AC, 60 Hz sync pulse generated by a zero voltage detection circuit or a simple real time clock. The COP400 series microcontrollers are suited to accommodate either of these time base schemes while accomplishing auxiliary tasks.

Zero voltage detection is the most useful scheme in AC power control because it affords a real time clock base as well as a reference point in the AC waveform. With this information it is possible to minimize RFI by initiating power-on operations near the AC line voltage zero crossing. It is also possible to fire the triac for only a portion of the cycle, thus utilizing conduction angle manipulation. This is useful in both motor control and light intensity control.

Sophisticated zero voltage detection circuits which are capable of discriminating against noise and switch precisely at zero crossing are not necessary when used in conjunction with a COPS device. COPS software is capable of compensating for noisy or semi-accurate zero voltage detection circuits. This can be accomplished by introducing delays and debounce techniques in the software routines. With a given reference point in the AC waveform it now becomes easy to divide the waveform to efficiently allocate processing time. These techniques are illustrated in the code listing at the end of this paper.



TL/DD/6939-5

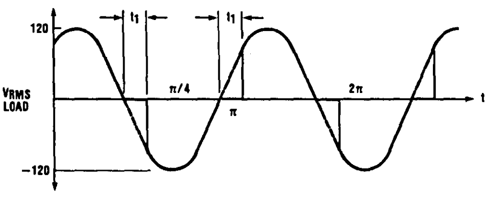
FIGURE 5. Current Lag Caused by Inductive Load, Snubber Circuit

2.2 PROCESSING TIME ALLOCATIONS

Half Cycle Approach

In order to accomplish more than triac timing, dead delay time must be turned into computation time. It appears that the controller is occupied totally by time delays, which leaves a very limited amount of additional control capability. There are, however, many ways to accomplish auxiliary tasks simultaneously.

On each half cycle an initial delay is incorporated to space into the cycle. This dead time may be put to use and very little voltage to the load is sacrificed. For example, if the load is switched on at $\pi/4$ RAD, the maximum applied RMS voltage to the load is $114V_{RMS}$ (assuming $V_{SUPPLY} = 120V_{RMS}$). This is illustrated in the figure below.



TL/DD/6939-6

FIGURE 6. Full Cycle Approach

If a delay of $\pi/4$ RAD (45 degrees) is inserted after each zero crossing detection the RMS voltage to the load can be determined in the following manner:

$$V_{LOAD} = \sqrt{\frac{(120\sqrt{2})^2}{(2)\,\pi}}\,(2)\int_{\pi/4}^{\pi}\sin^2(a)\,da$$

$$V_{LOAD} = \sqrt{\frac{(120\sqrt{2})^2}{(2)\,\pi}}\,(2)\,(1.428)$$

$$V_{LOAD} = 114.4\ V_{RMS}$$

$$\pi/4 \text{ RAD} = 45 \text{ degrees} \quad @60\text{ Hz} \quad t = 2.08\text{ ms}$$

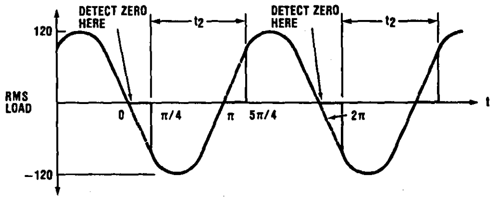
As can be seen the dead time on each half cycle can be 2.08 ms and the load will still see 114.4 V_{RMS} of a V_{SUPPLY} of 120 V_{RMS}. If this approach is implemented the initial delay of 2.08 ms can be used as computation time. The number of instructions which can be executed when operating at 4 μs instruction cycle time is:

2.08 ms/4 μs = 520 instructions

(130 instructions at 16 μs cycle time)

Full Cycle Approach

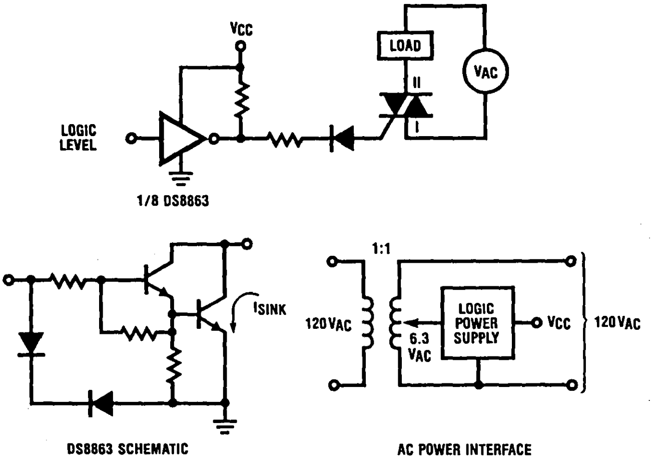
The methods of half cycle and full cycle triggering are very similar in procedure. The main difference is that all timing is referenced from only one (of the two) zero voltage detection transition in each full AC cycle. For most all applications, when varying the conduction angle it is desirable to fire at the same conduction angle each half cycle to maintain a symmetric applied voltage. In order to accomplish this the triac may be fired twice from one reference point. When applying this technique an 8.33 ms delay must be executed to maintain the symmetric applied voltage. This approach provides the most auxiliary computation time in that the 8.33 ms delay may be turned into computational time. The basic flow for this technique is illustrated below.



TL/DD/6939-7

FIGURE 7. Full Cycle Approach

In the above example the zero crossing pulse is debounced on the one-to-zero transition, thus marking the beginning of a full cycle. Once this transition has been detected, an ini-



TL/DD/6939-8

FIGURE 8. Steady State Triggering

tial delay of $\pi/4$ RAD is incorporated and the triac is fired. At this time exactly 8.33 ms is available until the triac need be triggered again. This will provide a symmetric voltage to the load only if the delay is 8.33 ms. During this period the number of instructions which can be executed when operating at 4 μs is:

$$8.33\ \text{ms}/4\ \mu\text{s} = 2082$$

$$(520\ \text{instructions at}\ 16\ \mu\text{s})$$

An alternative approach may be to take the burden from the COPS device by using peripheral devices such as static display controllers, external latches, etc.

2.3 STEADY STATE TRIGGERING

It is possible to trigger a triac with a steady state logic level. This is accomplished by allowing the triac gate to sink or source current during the desired on-time. When utilizing this method it becomes easier to trigger the triac and leave it on for many cycles without having to execute code to retrigger. This approach is advantageous when the triac must be fired is for relatively long periods and conduction angle firing is not desired, thus more time is available to accomplish auxiliary tasks. A steady state on or off signal and external circuitry can accomplish triac firing and free the processor for other tasks. If it is desired to use a pulse transformer, an external oscillator must be gated to the triac to provide the trigger signal. A pulse train of 10 to 15 kHz is adequate to fire the triac each half cycle. This calls for external components and is relatively costly. If isolation associated problems can be tolerated or overcome (dual power supply transformers, direct AC coupling, etc.), a simple buffer may be utilized in triggering the triac. This method is illustrated in *Figure 8.* The National Semiconductor DS8863 display driver is capable of steady state firing of the triac. National offers many buffers capable of driving several hundred milliamps, which are suitable for driving triacs. On the market today there are many suppliers of sensitive gate triacs which may be triggered directly from a COPS device or in conjunction with a smaller external buffer.

The DS8863 display driver is capable of sinking up to 500 mA, which is adequate to drive a standard triac. In the off state the driver will not sink current. When a logic "1" is applied to the input the device will turn on. Keeping the device off (output "1") will prevent the triac from turning on because the buffer does not have the capability of sourcing current. A series resistor limits the current from the triac gate and the diode isolates the negative spikes from the gate. Since the drive circuit will only sink current in this configuration, the triac will be operating in the I- and III- modes.

3.0 Triac Light Intensity Control Code

The following code is not intended to be a final functional program. In order to utilize this program, modifications must be made to specialize the routines. This is intended to illustrate the method and is void of control code to command a response such as intensify or deintensify. The control is up to the user and full understanding of the program must be attained before modifications can be implemented.

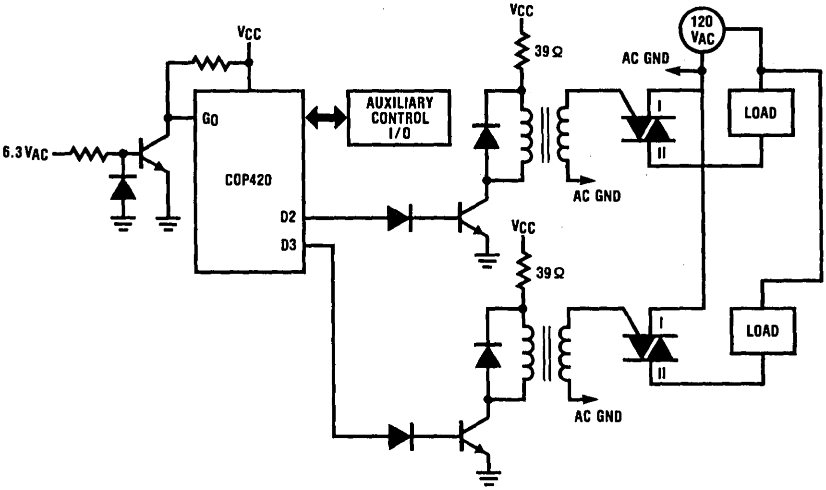
This program is a general purpose light intensifying routine which may be modified to suit light dimmer applications. The delay routines require a 4.469 μs cycle time which can be attained with a 3.578 MHz crystal (CKI/16 option). This program divides the half cycle of a 60 Hz power line into 16 levels. Intensity is varied by increasing or decreasing the conduction angle by firing the triac at various levels. The program will increase the conduction angle to a maximum specified intensity in a fixed amount of time. The time required to intensify to the maximum level is dependent on the number of fire-times per level that is specified (FINO). This code illustrates a half cycle approach and relies on the parameters specified by the programmer in the control selection.

Zero crossings of the 60 Hz line are detected and software debounced to initiate each half cycle; thus the triac is serviced on every half cycle of the power line. A level/sublevel approach is utilized to vary the conduction angle and provide a prolonged intensifying period. The maximum intensity is specified by the "LEVEL" RAM location and time required to get to that level is specified by the "FINO" RAM location. Once a level has been specified, the remaining time in the half cycle is then divided into sublevels. The sublevels are increased in steps to the maximum level. The "FINO" RAM location contains the number of times that the triac will be fired per sublevel, thus creating the intensity time base. There are 15 valid sublevels and up to 15 fire-times per sublevel. Both these parameters may be increased to provide better resolution and longer intensify periods. To make the triac de-intensity (dim) the sublevels need only to be decremented rather than incremented. If this is done, the conduction angle will start out at the maximum level and dim by means of stepping down the sublevels. When modifying this routine to incorporate more resolution or increased versatility, care must be taken to account for transfer of control instructions to and from the delay routines.

The following is a schematic diagram of the COPS interface to 120V_{AC} lamps. The program will intensify or de-intensify the lamps under program control.

3.1 TRIAC LIGHT INTENSIFY ROUTINE

This program intensifies a light source by varying the conduction angle applied to the load. The maximum level of intensity is stored in "LEVEL," and the time to get to that level is specified by "FIND." Both these parameters may be altered to suit specific applications. To cause the program to de-intensify the light source, the sublevels must be decremented rather than incremented.



TL/DD/6939-9

FIGURE 9. Triac Interface for COPS Program

```
; TRIAC LIGHT INTENSIFY ROUTINE
;
;
; THIS PROGRAM INTENSIFIES A LIGHT SOURCE BY VARYING THE
; CONDUCTION ANGLE APPLIED TO THE LOAD. THE MAX LEVEL
; OF INTENSITY IS STORED IN 'LEVEL' AND THE TIME TO GET TO
; THAT LEVEL IS SPECIFIED BY 'FIND'. BOTH THESE PARAMETERS
; MAY BE ALTERED TO SUIT SPECIFIC APPLICATIONS. TO CAUSE
; THE PROGRAM TO DE-INTENSIFY THE LIGHT SOURCE, THE
; SUBLEVELS MUST BE DECREMENTED RATHER THAN
; INCREMENTED.
;
;
          TEMP1     =1,0      ; TEMPORARY DELAY COUNTER
          FIND      =0,9      ; NUMBER OF FIRE TIMES
          LEVEL     =0,0      ; MAX LEVEL
          SUBLEV    =1,10     ; SUBLEVEL COUNT
          TEMP      =1,11     ; TEMPORARY DELAY COUNTER
;
; HERE THE OPERATING PARAMETERS ARE DEFINED AND LEVEL
; INITIATION IS SPECIFIED
;
          .FORM
          .PAGE     0
          CLRA                ; REQUIRED
CLRAM:    LBI       3,15      ; ROUTINE TO CLEAR ALL RAM
CLR;      CLRA
          XDS
          JP        CLR
          XABR
          AISC      15
          JP        BEGG
          XABR
          JP        CLR
;
; THIS SECTION INITIATES CONTROL ON POWER UP OR RESET
; AND SYNCHRONIZES THE COPS DEVICE TO THE 60 HZ AC LINE
;
BEGG:     OGI       15        ; OUTPUT 15 TO G PORTS TO PULL
                              ; UP ZERO CROSSER INPUT
          LBI       LEVEL     ; SPECIFY MAX LEVEL
          STII      7
          JSR       OUT       ; COPY TO TEMP1
BEG:      SKGBZ     0         ; SYNC UP TO 60 HZ
          JP        HI        ; READY NOW
          JP        BEG       ; WAIT TILL G IS 1
;
; THIS SECTION PROVIDES THE DEBOUNCE FOR THE ZERO
; VOLTAGE DETECTION INPUT AND COMPENSATES FOR THE
; OFFSET OF THE DETECTION CIRCUIT
;
HI:       SKGBZ     0         ; TEST GO FOR ZERO CROSS
          JP        HI        ; HIGH LEVEL
; GETS HERE ON FIRST TRANSITION
          CLRA                ; START OF DEBOUNCE DELAY
          AISC      1
          JP        .-1
; DID A LITTLE DELAY, IS IT STILL 0
          SKGBZ     0         ; TEST FOR 0
          JP        HI        ; FALSE ALARM
; MUST HAVE HAD SOME NOISE GO BACK AND WAIT FOR TRUE ZC
DOIT:     JMP       INT       ; VALID TRANSITION, SERVICE
                              ; TRIAC
LO:       SKGBZ     0         ; DEBOUNCE IN 0 TO 1
          JP        DDD       ; MAY HAVE SOMETHING THERE
          JP        LO        ; NO WAIT HERE FOR A BIT
DDD:      CLRA                ; GOING TO WAIT AND SEE
          AISC      1
          JP        .-1
          SKGBZ     0         ; WELL, DO WE HAVE A CLEAN
                              ; TRANSITION
          JP        DELL      ; YES, GO TO MAIN ROUTINE
```

```
          JP        LO        ; FALSE ALARM, TRY AGAIN
DELL:     CLRA                ; DO A DELAY TO COMPENSATE
DEL:      NOP
          NOP                 ; FOR NON SYMMETRIC ZC
          NOP
          AISC
          JP        DEL       ; KEEP DELAY GOING
          JP        DOIT      ; GO TO MAIN ROUTINE

          .FORM
          .PAGE     1
;
;
; THIS IS THE MAIN ROUTINE FOR THE INTENSIFY/DE-INTENSIFY
; OPERATIONS. TRANSFER OF CONTROL TO THIS SECTION
; OCCURS AFTER ZERO VOLTAGE CROSSING EACH HALF CYCLE.
; THIS MAKE USE OF TEMP REGISTERS THUS PARAMETERS
; NEED NOT BE REDEFINED FOR EACH OPERATION.
;
;
INT:      CLRA
          ADT                 ; DELAY INTO WAVEFORM
          LBI       TEMP      ; USE TEMP REG
          X
          JSRP      PORT      ; DO DELAY
POINT:    LDD       LEVEL     ; POINT TO LEVEL TO INITIATE
                              ; DELAY
                              ; DELAY TO MAX LEVEL
          XAD       TEMP      ; USE TEMP DIGIT TO DELAY
TAMP:     LBI       TEMP
          LD
          AISC      15        ; ARE WE AT THE LEVEL ?
          JP        ATLEV     ; MADE IT TO THE LEVEL
          X                   ; NO
          JSRP      DE5       ; DO SERIES OF .5MS TO GET
                              ; THERE
          JP        TAMP      ; KEEP DOING IT
ATLEV:    LDD       SUBLEV    ; AT MAX FIRE LEVEL
          XAD       TEMP      ; INIT FOR SUBLEVEL DELAY
JK:       LBI       TEMP
          LD
          AISC      1         ; AT SUB LEVEL ?
          JP        TRE       ; NO DO DELAY
          JP        SBLEV     ; YES
TRE:      X
          JSRP      SPDL      ; VARIABLE DELAY
          JP        JK
SBLEV:    LBI       FIND
          JSRP      DEC       ; DEC FIRE NUMBER
          AISC      1         ; TEST IF FIND AT 15
MAXLEV:   JMP       FIRE      ; NO KEEP FIRING AT THAT LEVEL
          LBI       SUBLEV    ; YES INC SUBLEVEL
          CLRA
          AISC      14        ; IS MAX SUBLEV REACHED
          SKE
          JP        THERE     ; NO INC SUBLEV
          JP        MAXLEV    ; YES FIRE IT
THERE:    JSRP      INC       ; GO TO NEXT SUBLEVEL
          LBI       FIND
          STII      14        ; SET FIRE TIME
          JP        MAXLEV    ; GO FIRE

          .FORM
          .PAGE     2
```

```
; SUBROUTINE PAGE
INC:      CLRA
          AISC      1
          JP        ADEX      ; GO ADD ONE TO DIGIT
DEC:      CLRA                ; 0 TO A
          COMP                ; CREATE A 15
ADEX:     ADD                 ; ADD A TO RAM
          X                   ; PUT BACK (D − 1 IN A NOW)
          RET
DE5:      LBI       0,10      ; DELAY ROUTINE
          CLRA                ; WILL BE REPLACED LATER
          AISC      3
          JP        .−1
          LD
          XIS
          JP        .−5
          RET                 ; DONE DELAY
FIRE:     LBI       0,15      ; PULSE D OUTPUT
          OBD
          NOP
          NOP
          NOP
          LBI       0,0
          OBD
          SKBGZ     0         ; TEST WHICH DEBOUNCE IS
                              ; NEEDED
          JMP       HI        ; DEBOUNCE ONE TO ZERO
          JMP       LO        ; DEBOUNCE ZERO TO ONE
SPDL:     LBI       TEMP1     ; TEMP1 IS A TEMP REG
PORT:     LD                  ; VALUE IN TEMP1 DICTATES
          AISC      1         ; THE AMOUNT OF DELAY
          JP        FOY
OUT:      LBI       LEVEL     ; ALSO USED TO COPY LEVEL
          LD        1         ; RESTORE LEVEL
          X
          RET
FOY:      X
          JP        PORT
          .END
```

Testing of COP400 Family Devices

National Semiconductor
COP Note 7

Table of Contents

This note will provide some insight into the test mode, the mechanics of testing, and the philosophy of how to implement a test of the COP-400 microcontrollers. Other than the obvious, (verifying that the part meets the specifications), the reason for the test must be considered. Somewhat different criteria may hold, depending on the objective. The manufacturer wafer sort or final test can differ from an incoming inspection at the user's plant, or a field reject test. The first two tests have limited interest as this is not a justification of the testing done on the part during manufacture. Rather, this is a guide for those doing user functional testing.

1.0 INTRODUCTION

Since the introduction of the very first semiconductor devices, testing has been a major problem and expense in their production and use. As the complexity has risen, testing has become a more significant factor. With today's single chip microcontrollers like the COPS™ devices this is particularly true as one has a complete computer system in a chip. In order to reduce the testing burden, the facilities to ease the testing have been built into the COPS devices. With the test ability built into the device for production test, the user need only follow set procedures to verify the chip at incoming inspection or field test.

2.0 PHILOSOPHY

The basic test philosophy requires that four major areas be exercised. These areas are:

1) Synchronize the device and tester.

2) Test the internal logic and I/O.

3) Test the RAM.

4) Verify the ROM program.

If the devices perform all of these four properly, the device is good. This is a reasonable assumption with a standard device that has a debugged test routine and is ROM programmed. A custom circuit just going into production might not have the accumulated test background. By attacking the problem on a "sum of the parts" approach, one need not do any exhaustive functional test on routine production parts. This will be a major gain where lengthy time consuming or time dependent routines are involved. If one attempts to do a functional test of the chip, a sequence that is unique to the application is needed. Thus, a test program must be written and debugged for each ROM pattern. Further, a test box/board must be designed, built, debugged, documented, and maintained for each one. If testing has been considered from the beginning, the chip will have built-in capabilities to exercise the various parts of it. The different functional parts and instructions are tested to verify proper operation at the voltage and frequency limits.

3.0 BUILT-IN TEST FEATURES

The first step in testing the COP400 devices is to understand the built-in test control features. This will involve the SI/O and the L lines. The SO pin has been designed to be the control node for testing. The pin will normally be in an active low state and when forced high externally, places the chip in the test mode. It should be noted that this output can sink considerable current and one should not force the pin to the V_{CC} rail. By limiting the voltage to the 2.0/3.0V range one can not damage the device where the application of a higher voltage could. When forced into the test mode the SI pin controls the sub mode of the chip. With SI high the data placed on the L port is used as an instruction. When SI is low (and the L output is enabled) the contents of the ROM will be dumped out through the L port. Certain other internal functions have been implemented to allow these modes but these are not part of the basic operation. Included in this category is the activation of the skip signal to prevent the program counter from jumping out of sequence by executing a program control instruction.

3.1 Sync Between Tester and DUT

In order to be able to test a COPS chip, the tester must be in sync with the device under test (DUT). By using an external oscillator the two may be run at the same frequency. This is true regardless of the option or type of oscillator chosen for the chip. Even the RC configuration may be overridden with an external signal that meets the level requirements. In addition to running at the same frequency, the chip and tester must be in sync on a bit basis. See *Figure 1.* The supportive features mentioned above include the condition of the SK signal being a bit (instruction) clock until stopped by software in the program. Hence, one can start the tests based on an edge change of SK. It is important that this be accurate because all data I/O changes will be relative to the SK timing (see the appropriate device data sheet).

It should also be noted that the oscillator frequency is programmed to a rate of 4-32 higher than SK. If one is building a test fixture for more than one device, some method must be available to enter this number. If one is testing a COP420 or COP421 near its upper limit it would be wise to do the SK sync operation at a lower rate and then increase the input frequency. This is desirable because the phase relationship is close to TTL propagation delays at the upper limit. Implementation of the area could be a preset counter that is gated on after a zero to one transition is seen on SK. Continual comparison could be made but once in sync, there should not be any need for the comparison as they should remain in sync.

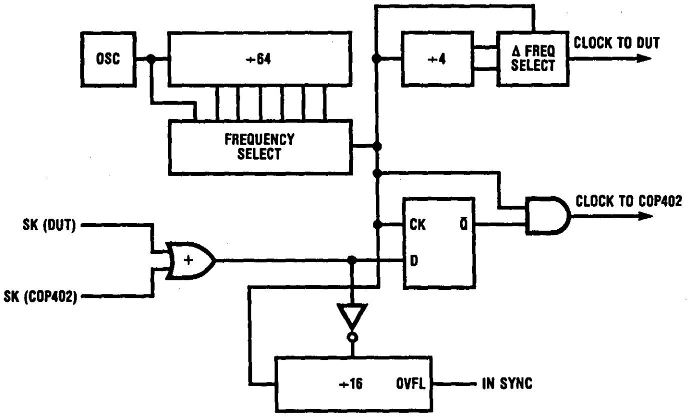
The basic use of this "sync counter" is to derive the proper timing for loading data and instructions into the chip and verify the outputs. The COP402 data sheet should be used as a guide for these times, modified properly for the L and C parts. For those designing testers, it is suggested that one not attempt to test worse case timing changes as these could be very difficult to implement. Like other parametric tests these should in general be left to the professional test equipment.

3.2 Internal Logic Test

With the device and the tester in sync, actual testing may begin. See the sequence control circuit of *Figure 2.* To place the chip into the test mode the SO output is pulled to a one level (between 2.0 and 3.0 volts). It should be pulled with a circuit that will limit the upper voltage to 3V as this output can have a significant current sink capability. On power up (or after reset) the SO line is set to a zero by the internal logic. An internal sense line will detect the forced condition and provide test control. A delay of 10 ms should be taken after power-up to allow the power on reset circuit to time out before instructions can be executed. If the reset pin is activated in mid-program for some reason, several instructions cycle times should be ignored to insure complete operation.

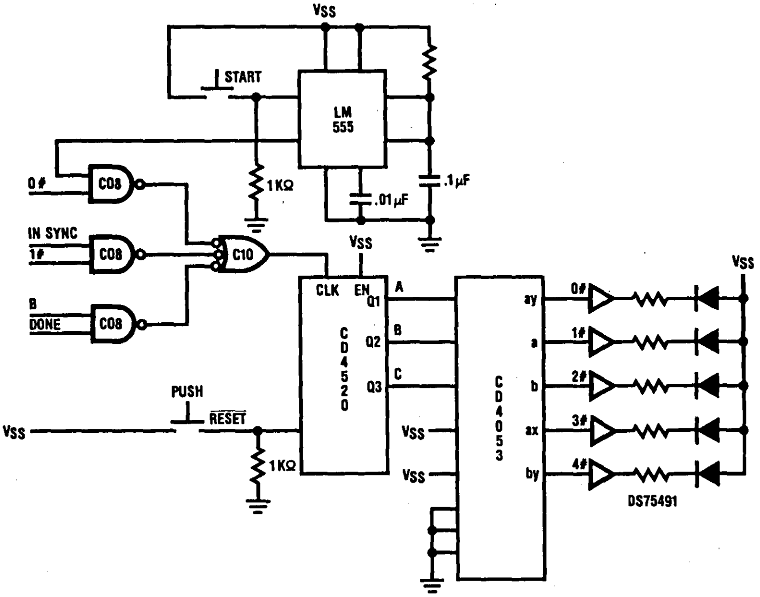
The tester should at this point force instructions into the L port. These instructions will be executed as if they were from the ROM. The sequence of the instructions is not particularly critical. Table I gives an example sequence. The main steps are to be able to detect an output change (OGI) early to verify connection/operation. It is much better to find a problem before going through the steps of loading RAM and then finding that the chip doesn't work. All instructions should be exercised although certain ones should be postponed. Enabling the Q register to the L port is an example. This would interfere with the insertion of instructions on the L port. Another problem is the SO test which could be set up with an XAS and then released from the test mode to check proper data output.

Certain commands will require more effort than others. To check the program counter during JMP's and sub-routine operation will require that known info at the new address be available. One should execute a JSRP at some known address and release the test mode to see that the operation in the subroutine (e.g., SC) is done and that a return is made to N + 1. At this point test mode can be re-established to continue the test. The main point to remember is to provide a positive indication of the success of that specific test.



TL/DD/6940-1

FIGURE 1. Tester Clock Generation and Synchronization Circuit



TL/DD/6940-2

FIGURE 2. Tester Mode Sequencer

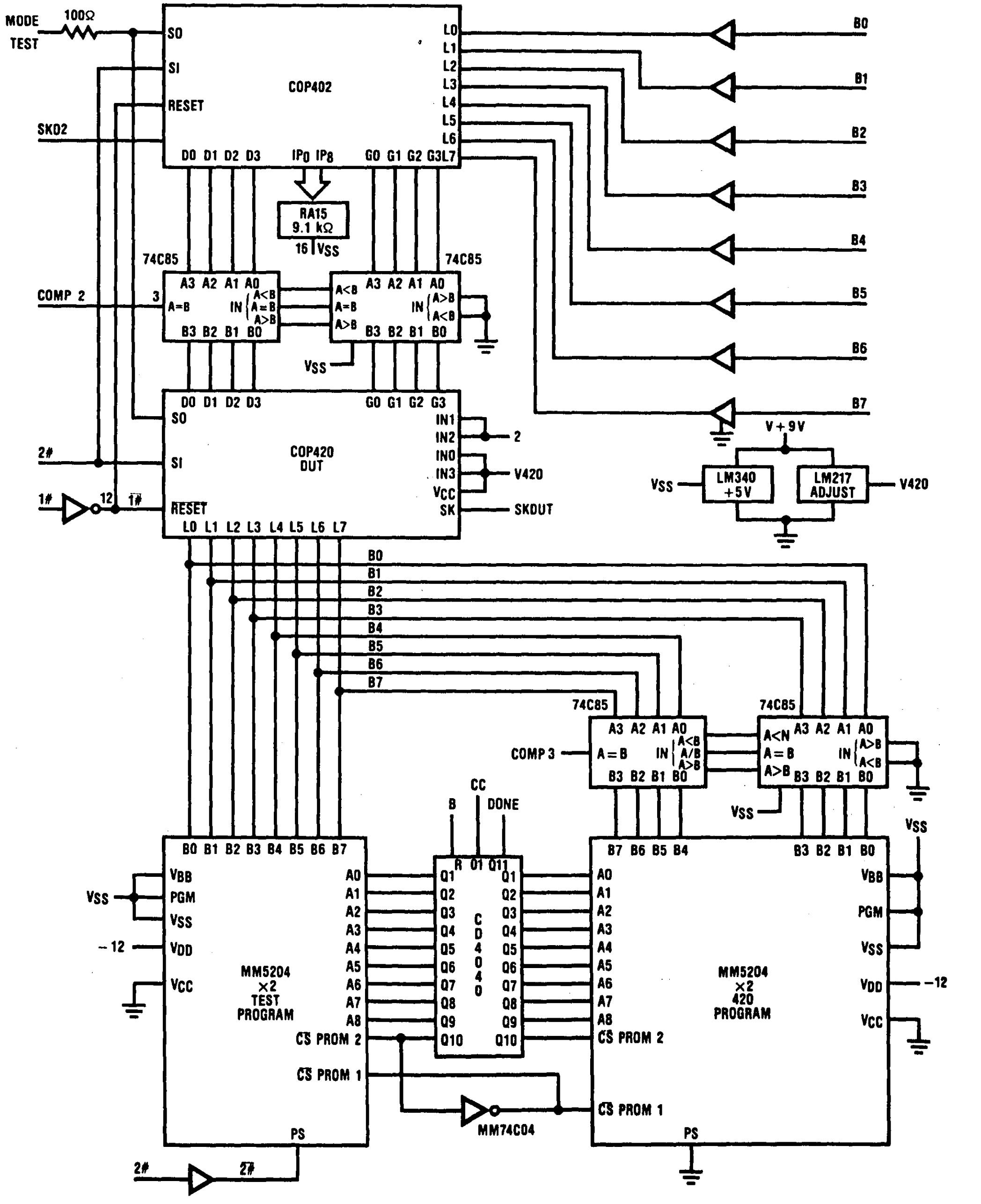
3.3 RAM Test

The verification of RAM is a part of the internal logic test, but is treated separately here. One must check both the RAM and its address register to find all faults. An example of this testing would be to load RAM with a string of STII commands. By then going back and reading this data to the outside (through an OMG instruction in a loop) the tester could verify both RAM and address were functional. One could then load RAM with all 6's and 9's (or 5's and 10's) sequentially to insure that all bits were functional and adjacent bits not shorted. Other similar tests could be run at the discretion of the user to do further testing. All of these tests would utilize the output of data via the G ports to validate the data. See the comparator circuit *Figure 3.*

3.4 ROM Dump

Successful operation of the internal logic tests and RAM will lead to the final test phase, ROM comparison. In order to check the ROM contents, the ROM dump mode must be entered. One should force a JMP to an address near the end of the ROM space (3FF for a 420 chip, 1FF for a 410). A desirable point might be 3FA. The program counter will step ahead on each instruction cycle unless a program control is executed. The next step is to load the Q register with a non-conflicting value so that the enabling of the L outputs will not destroy the second byte of the LEI instruction as control is passed into the ROM dump mode. After going to this address, one should execute an enable of the L lines to the output port (LEI 4). Having done this the external buffers should be disabled and the SI pin taken low. This will allow data out and remove potential level conflicts. By letting the PC step ahead to address zero one can then begin the byte by byte comparison of data. In this mode the controller is not executing the code because the skip line is enabled throughout the sequence. By halting a counter on a failure, one could determine the questionable address.

COP NOTE 7



TL/ DD/6940–3

FIGURE 3. Functional Logic and RAM Comparison Circuit

TABLE I. Typical Test Sequence

INSTRUCTION	RESULT	COMMENTS
NOP	NO CHANGE	CHECK NOP & ALLOW TRANSIENT CYCLE FOR MODE
OGI 9	G(0 > 9)	NOT ON 410L/411L
OGI 6	G(9 > 6)	REVERSE ALL G STATES
STII 8		SET UP 0,0 FOR FUTURE
LBI 3,13		B TO NEW POSITION (3, 13)
OBD	D(0 > 13	CHECK D
CLRA		MAKE SURE A = 0
XABR		3 > A; 0 > Br
CAB		MOVE 3 to Bd
OBD	D(13 > 3)	CHECK XABR CAB & D CHANGE
CLRA		!
AISC 2		!FORCE A > 2
CAB		2 > Bd
OBD	D(3 > 2)	VERIFY 2 FROM A > Bd
STII 7		7 > 0.2 & Bd > 3
OBD	D(2 > 3)	STII INCREMENTS Bd
CAB		SEE THAT A STILL THE SAME
OMG	G(6 > 7)	OMB & RAM CHECK
CLRA		
CAB		B(0,0)
OMG	G(7 > 8)	TIE IN RAM, A & G OPERATION
SMB 0		SMB INST. CHECK
OMG	G(8 > 9)	:
SMB 1		:
OMG	G(9 > 11)	:
RMB 0		:
RMB 3		:
X		:0 > 0,0;2 > A
CAB		A = 2 > B
OMG	G(11 > 7)	OUTPUT M(0,2)
LD 1		M(0,2) > A; B > 1,2
XAD 0,0		A(7) < -> M(0,0) 2
AISC 15		AISC CHECK; A = 1
LDD 0,0		CHECK SKIP OF 2 BYTE INST.
X		STORE 1
OMB	G(7 > 1)	VERIFY
LD 0		COPY1,2 BACK TO A
ADT		ADD TEN
XDS		LEAVE 11 IN 1,2;GO 1, 1 WITH 1
XDS		LEAVE 1 IN 1,1;GO 1,0 W ?
OBD	D(2 > 0)	CHECK Bd MOVEMENT
STII 5		5 > 1,0;Bd TO 1,1
CBA		CHECK B > A
AISC 3		AISC CHECK 4 >A
XDS		1 > A; 4 > 1,1
OMG	G(1 > 5)	FROM 1,0
XDS		5 > A; 1 > 1,0; Bd < 15 SKIP
LDD 0,0		SKIPPED !
OBD	D(0 > 15)	
AISC 4		9 > A
X		9 > 15
OMG	G(5> 9)	
CLRA		
COMP		ONES TO A
XOR		FLIP MEMORY
XIS		6 > 1,15; 9 > A; Bd > 1,0
LDD 0,0		SKIP
SKE		
LB 1,2		SKIP 2 WORD LBI (NOT IN 410)
OBD	D(15 > 0)	VERIFY WORD
SKE		11 NOT = 9
LBI 1,0		BACK TO 1,0
SMB 2		:
SKE		:
RMB 2		:
SKE		:CHECK BIT
SMB 3		:MANIPULATIONS
SKE		:
LDD 0,0		:
X 3		Bd > 2,0
XAD 1,1		9 > 1,1; 4 > A
XIS 1		4 > 2,0; Bd > 3,1
ING		INPUT G PORT
X		STORE
CLRA		
ASC		CHECK ADD WITH CARRY
SC		CHECK SET CARRY
SKC		CHECK SKIP ON CARRY
LDD 0,0		
X		STORE A
OMG	G = 9	NO CHANGE
CLRA		
ASC		
X		
OMG	G(9 > 10)	CARRY ADDS ONE TO MEMORY
CAMQ		STORE A & M IN Q; 10,9
XDS		9 > 3,1; 10 > A; Bd > 3,0
X		STORE 9 IN 3,0
OMG	G(10 > 9)	
LD 2		9 > A; Bd > 1,0
OMG	G(9 > 1)	
LD 3		1 > A; Bd > 2,0
OMG	G(1 > 2)	
ADD		ADD WITHOUT CARRY
X		STORE 3 IN 2,0
SC		
LDD 0,0		7 > A
CASC		CHECK CASC
SKC		
X		STORE 12
OMG	G(2 > 12)	
CLRA		:
AISC 3		:
X		:
SC		:CHECK
SKC		:SKC/SC
X		:
OMG	G(12 > 3)	
RC		:
SKC		:CHECK
X		:RC
OMG	G(3 > 12)	:
LBI 0,0		:CHECK
LBI 1,15		;SEQUENTIAL LBI'S
LBI 2,7		ALSO SKIPPED (LBI 2,7 NOT IN 410)
OMG	G(2 > 7)	
CQMA		LOAD CONSTANTS FROM Q
OMG	G(7 > 9)	CHECK
X		:
OMG	G(9 > 10)	:
LEI 1		
XAS		STORE A – > S (9)
CLRA		
AISC 7		:
SKGBZ 0		:
X		:CHECK
OMG		:
SKGBZ 1		:
X		;G BIT
OMG	G (10 > 7)	:
SKGBZ 2		:
X		:
OMG	G(7 > 10)	:TESTS
SKGBZ 3		:
X		:
OMG	G(10 > 7)	:
SKGZ		
X		:CHECK
OMG	G(7 > 10)	:
OGI 0	G(10 > 0)	:G TEST
SKGZ		:
X		:
OMG	G(0 > 10)	:
SKMBZ 0		
X		CHECK MEMORY BIT TESTS
OMG		NO CHANGE
SKMBZ 1		

TABLE I. Typical Test Sequence (Continued)

INSTRUCTION	RESULT	COMMENTS
X		
OMG	G(10 > 7)	NO SKIP
SKMBZ 2		
X		WON'T SKIP
OMG	G(7 > 10)	
INIL		SEE THAT L LATCHES RESET
ININ		ASSUME G − > I
SKE		
X1		Br > 1
OMG		SHOULD BE EQUAL
INIL		:
X		:
SKMBZ 3		:
OBD	D(15 > 0)	:INIL TEST
OGI 1		:
LBI 3,11		:
OGI 0		:
INIL		:
X		:
SKMBZ 0		:
OBD	D(0 > 11)	:
NOP		
XAS		:
X		:XAS TEST
OMG	G(10 > 9)	:

INSTRUCTION	RESULT	COMMENTS
LBI 0,0		LOAD RAM WITH
STII 7		CONSTANTS USING
STII 14		STII
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		
LBI 1,0		
STII 7		
STII 14		
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		
LBI 2,0		
STII 7		
STII 14		
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		

INSTRUCTION	RESULT	COMMENTS
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		
LBI 3,0		
STII 7		
STII 14		
STII 5		
STII 12		
STII 3		
STII 10		
STII 1		
STII 8		
STII 15		
STII 6		
STII 13		
STII 4		
STII 11		
STII 2		
STII 9		
STII 0		

INSTRUCTION	RESULT	COMMENTS
LBI 0,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
LBI 1,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

TABLE I. Typical Test Sequence (Continued)

INSTRUCTION	RESULT	COMMENTS
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
LBI 1,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
LBI 3,0		CHECK FOR RAM DATA
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT
OMG		OUTPUT DATA
LD		:
XIS		:MOVE TO NEXT DIGIT

INSTRUCTION	RESULT	COMMENTS
JMP X	INITIALIZE—SELECT ADDRESS X FOR OGI OR OMG (SELECT LBI FOR KNOWN DATA)	
RELEASE TEST MODE	OBD (SELECT B FOR KNOWN CONDITION) CHECKS JMP	

TABLE I. Typical Test Sequence (Continued)

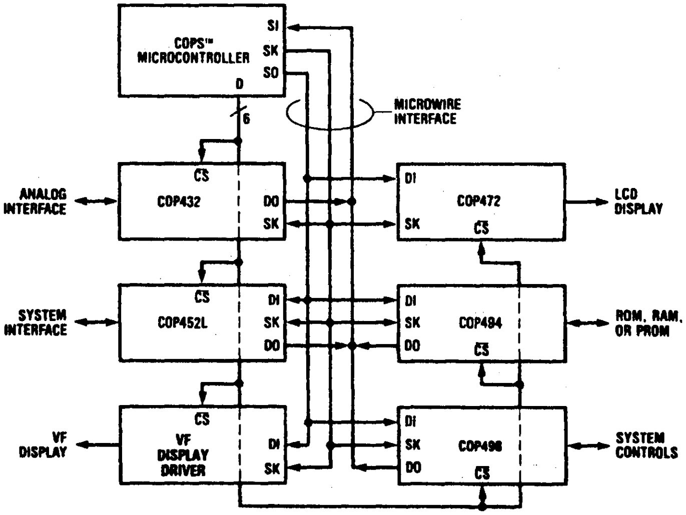
INSTRUCTION	RESULT	COMMENTS
SET TEST MODE		
JP X-2	:	
JSR Y	CHECK JP & JSR	
RELEASE TEST MODE	"Y" SHOULD CHANGE THE OUTPUT	
	CONDITIONS OF "X"	
EXECUTE CODE (Y)	IF AT ALL POSSIBLE	
SET TEST MODE		
RET		
RELEASE TEST MODE		
EXECUTE "X" AGAIN	VERIFIES RET	
SET TEST MODE		
JP X-2		
JSRP Z	CHECK JSRP & RETSK	
RELEASE TEST MODE		
EXECUTE CODE	"Z" SHOULD CHANGE "X"	
	OUTPUT CONDITIONS	
SET TEST MODE		
RETSK	DON'T CHANGE Z CONDITIONS —	
	RETSK	
RELEASE TEST MODE		
EXECUTE	" "	
SET TEST MODE		
LOAD A & M TO	FIND VALUE OF ADDRESS IN BLOCK	
	(4 PAGES)	
VALUE OF ADDRESS	AT OR JUST BEFORE AN OUTPUT	
TO GO TO	CHANGE SET A & M TO ADDRESS	
OUTPUT CHANGE	OF "VALUE"	
JID	CHECKS JID	
RELEASE TEST MODE		
EXECUTE OUTPUT		
SET TEST MODE	LOAD A & M WITH A UNIQUE ADDRESS	
LOAD A & M	SUCH THAT CONTENTS OF THAT	
	ADDRESS WILL BE SEEN ON G	
LQID		
X064	;OR USE THIS CAUSE THE DATA COMES	
	;FROM YOUR TESTER ANYWAY	
CQMA		
OMG	LQUID & CQMA CHECKED	
X		
OMG	"	
INL		:
OMG	G − > 2	INL TEST (COPY OF 2nd BYTE)
X		
OMG	G − > E	:

This test sequence is not to be taken as a recommended test routine and is only shown as an example of what might be done to test various COPS parts. It is also advisable to approach measurements in the test mode with some caution. As stated earlier, one can force a large current into the SO node to place the chip in the test mode. Not only can this current do damage if unlimited, but it can also cause local current overloading such that some I/O conditions may be adversely affected. Obviously this will be more pronounced at higher V_{CC} voltages. A specific example is that the L output current sink test should only be tested at a V_{OUT} of 0.4V and 0.36 mA as the more stringent tests can exceed power limits when combined with the SO current.

MICROWIRE™

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and/or may permit the COPS controller to be packaged in a smaller (and even lower cost) package. (MICROWIRE peripherals may also be used with non-COPS controllers). For further applications information, refer to COPS Briefs 8 and 9. MICROWIRE makes sense.

The example below illustrates the power and versatility of MICROWIRE via an extreme example—using one of each type of peripheral with a single controller.



TL/DD/6940-4

COP431 SERIES, 8-BIT A/D CONVERTERS

The COP431 series is an 8-bit successive approximation A/D converter with a serial I/O and configurable input multiplexer with up to 8 channels. The serial I/O is configured to comply with the NSC MICROWIRE serial data exchange standard for easy interface to the COPS family of processors, and can interface with standard shift registers or other µPs.

The 2, 4 or 8 channel multiplexers are software configured for single-ended or differential inputs as well as channel assignment.

The differential analog voltage input allows increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

COP452L FREQUENCY/COUNTER PERIPHERAL

The COP452L contains 2 independent 16-bit counter/register pairs, and is well suited to a wide variety of tasks involving the measurement and/or generation of times and/or frequencies. Included are multiple tones, precise duty cycles, event counting, waveform measurement, "white noise" generation, and A-D/D-A conversions. An on-chip zero-crossing detector can trigger a pulse with a programmed delay and duration.

COP472-3 LIQUID CRYSTAL DISPLAY CONTROLLER

The COP472-3 Liquid Crystal Display (LCD) Controller drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472-3 contains an on-chip oscillator and generates all the multilevel waveforms for backplanes and segment outputs on a triplex display. One COP472-3 can drive 36 segments multiplexed as 3 × 12 (4½ digit display). Two COP472-3 devices can be used together to drive 72 segments (3 × 24) which could be an 8½ digit display.

COP494 256-BIT SERIAL ELECTRICALLY ERASABLE PROGRAMMABLE MEMORY

The COP494 is a 256-bit non-volatile memory. The device contains 256 bits of read/write memory divided into 16 registers of 16 bits each. Each register is serially read or written by the COP400 Family Controller. Written information is stored in a floating gate cell with at least 10 years of retention.

COP498/COP499 LOW POWER CMOS RAM AND TIMER

The COP498 low power CMOS Random-Access Memory and Timer is an external memory and timer chip with the simple MICROWIRE serial interface. The device contains 256 bits of read/write memory divided into 4 registers of 64 bits each. The COP498 also contains a crystal-based timer for timekeeping purposes, and can provide a "wake-up" signal to turn on a COPS controller.

The COP498 can be used for low power standby memory and can also be used for low power operation by turning the controller off and on, on a duty cycle basis.

The COP499 Low Power CMOS Random-Access Memory is an external memory and switch chip with the simple MICROWIRE serial interface. The device contains 256 bits of read/write memory divided into 4 registers of 64 bits each. The COP499 also contains circuitry that enables the user to turn a controller on and off while maintaining the integrity of the memory.

Current Consumption in NMOS COPS™ Microcontrollers

National Semiconductor
Application Brief 3
Len Distaso

Current consumption in the N-channel COPS microcontrollers is a function of manufacturing process variation and three operating condition parameters: temperature, voltage, and frequency. The aforementioned process variation swamps all other variations. Of the operating condition parameters, temperature is by far the most significant. This application brief is intended to provide the user with a guide to approximate the worst-case current consumption of the NMOS COPS microcontroller at a given set of operating conditions and to approximate the current variation with respect to temperature, voltage, and frequency.

Note that this is a guide only. Some approximations in the equations have been made. Only the current values found in the various device data sheets are guaranteed. Values derived by the techniques described here are neither guaranteed nor tested.

PROCESS VARIATION

If a user were to measure the current in two identical COPS microcontrollers under identical operating conditions (i.e., same temperature, voltage, and frequency), the results would probably be different. The reason for this difference is variation in the manufacturing process within its valid range. This variation can be quite substantial; a range of about 3 to 1 can be expected. This variation is essentially a device-to-device variation and basically not related to the operating conditions of the device. The three operating condition parameters (temperature, voltage, and frequency) affect current in the manner described below.

The values for current consumption in the various device data sheets are worst-case maximum values and assume that the processing parameters are at the end of the valid range which will produce maximum current consumption in the device.

THE EFFECT OF FREQUENCY

The frequency effect on current consumption is primarily a device design consideration. The higher the intended operating frequency, the higher the maximum current. However, once the device is designed in this process for a given maximum frequency, there is little variation with operating frequency. To be sure, there is some variation. As might be expected, current consumption is greater at higher frequencies. The variation is, however, slight—typically less than 5%.

THE EFFECT OF VOLTAGE

The operating voltage of the microcontroller has a slightly greater effect on current consumption than the operating current. Current consumption increases with increasing operating voltage. On examining the MOS device equations, one finds that the device current is proportional to the square of a voltage term:

$I \alpha (V_{GS} - V_T)^2$

where:

I = device current

V_{GS} = device gate to source voltage

V_T = device threshold voltage.

In the N-channel COPS devices, current is consumed primarily by the load devices. Most of these devices, though not all, are depletion mode devices with the gate and source tied together. Thus, V_{GS} is 0. Therefore, the primary mechanism for current consumption as related to voltage is variation in V_T. The depletion mode load devices in the COPS NMOS microcontrollers have geometries (length is much greater than width) which tend to minimize variations in threshold voltage. There are additional second order effects related to operating voltage, such as effective channel lengths shortening due to increased voltage, which affect current consumption. These effects, however, do not have a major impact on current consumption. Note also that the threshold voltage is affected by process variation. This is one of the areas where the process variation contributes to the device-to-device variation in current consumption. The user can typically expect to see a 5% to 10% variation in current due to operating voltage with the maximum current consumption occurring at maximum operating voltage.

THE EFFECT OF TEMPERATURE

Of the three operating parameters affecting current consumption in the NMOS COPS microcontrollers, temperature has by far the greatest impact. The relationship is given by the following simplified, empirical equation:

$I(T) = I_O(T/T_O)^{-3/2}$

where:

T_O = reference junction temperature in °K

T = device junction temperature in °K

I_O = device current at temperature T_O

$I(T)$ = device current at temperature T.

Although this equation is for a single transistor, it can be applied to the entire microcontroller since all the devices are made with the same process and will exhibit the same characteristics. It should also be noted that the temperatures involved are device junction temperatures. The junction temperature is essentially a function of two items:

$T_j = F(T_A, \theta_{jA})$

where:

T_j = junction temperature

T_A = ambient temperature

θ_{jA} = package thermal characteristic.

The preceding relationship indicates that the package for the device will affect current because the package affects junction temperature. This should not come as a surprise. One need only consider the differences between ceramic and plastic packages to find support for this claim.

For purposes of discussion, it will be assumed that junction temperature is given by the following:

$T_j = T_A + 25°K$

where T_j and T_A are as defined previously. Note that this is an approximation. It is not necessarily true for all packages, or any package. The relationship between junction temperature and ambient temperature is also not necessarily linear. However, the approximation is reasonable and provides a workable framework.

Substituting the junction temperature relationship into the current equation, the following equation results:

$$I(T_A) \cong I_O \left(\frac{T_A + 25}{T_{AO} + 25}\right)^{-3/2}$$

where:

T_{AO} = reference ambient temperature, °K

T_A = ambient temperature, °K

I_O = current at ambient temperature T_{AO}

$I(T_A)$ = current at ambient temperature T_A.

AN EXAMPLE

The COP320L has a specified maximum current of 10 mA. In this process, maximum current occurs at minimum temperature, which is −40°C in this case. It is desired to find the maximum current at 25°C. Therefore,

$T_{AO} = -40°C = 233°K$

$T_A = 25°C = 298°K$

$I_O = 10\text{ mA}$

$I(T_A)$ to be determined

$$I(T_A) \cong I_O \left(\frac{T_A + 25}{T_{AO} + 25}\right)^{-3/2}$$

$$\cong 10\text{ mA } (323/258)$$

$$\cong 7.14\text{ mA}.$$

Thus the maximum current for the COP320L at 25°C is approximately 7 mA.

CONCLUSION

A means is provided to the user to approximate the current variation of the NMOS COPS microcontroller over its valid operating range. A given device will consume its maximum current at maximum operating voltage, maximum operating frequency, and minimum operating ambient temperature. Conversely, minimum current will be consumed at minimum operating voltage, minimum operating frequency, and maximum operating ambient temperature.

The user should remember that this document is intended as a guide only. The values produced here are reasonable but they are approximations and are not guaranteed values. The user should also remember that the equations and methods discussed here do not involve process variation. The numbers calculated approximate the worst-case maximum current values at a given set of operating conditions. The user should be prepared to see a wide range of values over the course of volume production.

Further Information on Testing of COPS™ Microcontrollers

National Semiconductor
Application Brief 4
Len Distaso

COP Note 7 describes the basic approach and philosophy for testing COPS microcontrollers. This application brief is intended to complement and expand COP Note 7. It is assumed that the reader is familiar with and has access to COP Note 7.

TEST MODE

On COPS microcontrollers, test mode is entered by forcing the SO output to a logic "1" when it should otherwise be a logic "0". The easiest way to do this is to hold the COPS device in reset, hold the $\overline{\text{RESET}}$ pin low, and pull SO up to a logic "1" level. WARNING: *Do not force more than 3.0V on SO, as damage to the device may occur.* SO should be forced to approximately 2.5V to guarantee entry into test mode and to protect the device from damage.

Once the device is in test mode, the state of the SI input controls the type of test. SI at a logic "1" (high level) conditions the device to accept instructions from an external source via the L port. In test mode, when SI is high, the internal ROM is disabled. SI at a logic "0" (low level) forces the device to dump the internal ROM to the L port where the user can read and verify the ROM contents.

INSTRUCTION INPUT

With the device in test mode and SI at a logic "1", the microcontroller will read the data at the L port as instructions. The instructions must be presented at the beginning of each cycle time and must remain valid during the whole cycle time. The chip SK output is the instruction cycle clock in test mode and can be used as the timing reference. *Figure 1* indicates the timing for instruction input using the chip's SK output as the reference. A new instruction must be valid at the L inputs within approximately 200 ns of the rising edge of SK. The user should make every effort to make this time (t2 in *Figure 1*) as short as possible.

It is possible to create an external SK signal which more closely duplicates the internal SK. This requires building a divider from CKI and synchronizing the resultant signal with the device under test. This is significant because it is the internal version of the SK signal which is the master timing signal for the microcontroller. The short time from the rising edge of the SK output to instruction valid is necessary because the actual objective is to provide new instructions at the rising edge, or close to it, of the internal timing signal. If the user creates the external timing signal, the 200 ns time is not applicable. A new instruction, or ROM word, would be presented at each rising edge of the external signal. A method for generating and using this external SK is described in COP Note 7.

ROM DUMP

With SI at logic "0" in test mode, the microcontroller will dump the ROM to the L port. ROM will be dumped sequentially, one word at a time, starting at whatever value the program counter contains. A new ROM word appears at the L lines every falling edge of the chip SK signal. The output timing (t1 in *Figure 1*) is the L output timing as found in the various device data sheets. The device will remain in ROM dump mode as long as SI is at logic "0" in test mode. The program counter will wrap around from the maximum address to 000 and ROM dump will continue.

To get a ROM dump, the user cannot simply enter test mode and force SI to logic "0". Some conditioning of the device is necessary. This requires that the user first go into instruction input mode and set up the device. The suggested sequence is as follows:

1. Enter test mode—pull $\overline{\text{RESET}}$ low, force SO to about 2.5V.
2. Force SI to logic "1" and force 0s on L lines—RESET still low.
3. Force RESET high and input the following sequence to the device:

```
CLRA
JMP    3FC   (modify for ROM size)
LQID
O44H
LEI    4
NOP
```

4. During the NOP, change SI from high to low as shown in *Figure 2.* The ROM dump should start at address 000H at the time shown in *Figure 2.*

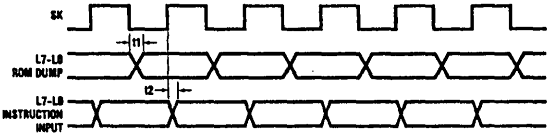
Figure 3 presents a general timing diagram for the entire sequence above. The jump instruction (JMP 3FC) in the sequence is used merely to position the program counter so that the ROM dump will begin at a specified location. That jump will be modified to reflect different ROM sizes or different desired starting locations for the ROM dump.

CHANGING BETWEEN INSTRUCTION INPUT AND ROM DUMP

The change from instruction input to ROM dump is accomplished according to the timing in *Figure 2.* It is necessary to do this to perform a valid ROM dump. However, it is not recommended to go the other direction, from ROM dump to instruction input, "on the fly". The instruction input mode should only be entered while the device is reset, $\overline{\text{RESET}}$ line low, to guarantee proper timing.

CONCLUSION

With COP Note 7 and this application brief, the user should be able to create a workable functional test for his COPS microcontroller. The relative timing is presented here and general techniques and sequences are provided in COP Note 7.

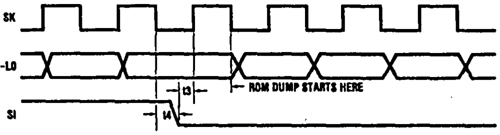


t1 = L output timing (t_{PD1}, t_{PD0}) as found in data sheet

t2 ~ 200 ns max

TL/DD/5146-1

FIGURE 1. Basic Test Mode Timing

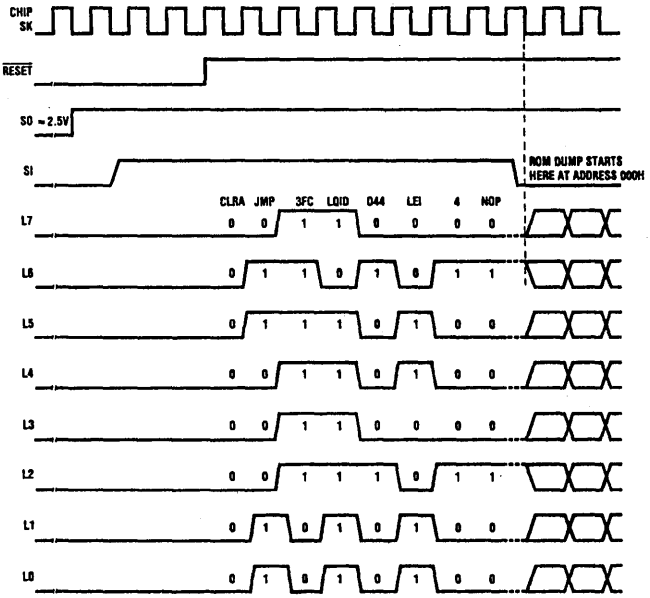


t4 ≈ t3 ~ 1 μs min for 4 μs devices

t4 ≈ t3 ~ 4 μs min for 16 μs devices

TL/DD/5146-2

FIGURE 2. Timing for Changing from Instruction Input to ROM Dump—Test Mode



TL/DD/5146-3

FIGURE 3. Relative Timing for Suggested Sequence to Generate ROM Dump

COPS™ Interrupts

National Semiconductor
Application Brief 6
Jim Murashige

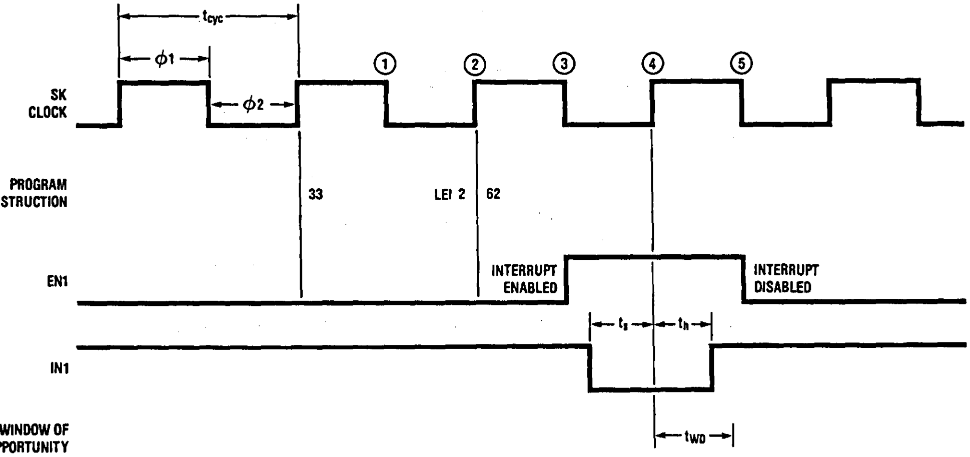
This brief describes in detail the timing requirements pertinent to COPS interrupts. *Figure 1* shows a typical enable-interrupt sequence in relation to the SK (Instruction Cycle) Clock. The SK clock is actually derived afrom the ϕ1 clock which is 180° out of phase with the ϕ2 clock. It is the ϕ1 and ϕ2 clocks to which all operation is referenced but for our purposes the SK will suffice. Program instructions are read on a rising ϕ1 edge and executed during the ϕ1, ϕ2 cycle time. Here we see the EN register interrupt enable bit EN2 being set with an LEI instruction. Interrupts are actually enabled on the ϕ2 leading edge of the second byte of the instruction point ③. Timing for an INTERRUPT DISABLE is essentially the same.

The interrupt line is sampled on the leading edge of ϕ1 as shown and interrupts are recognized if the minimum setup and hold times shown are satisfied. Note that the guaranteed times are longer than the typicals. The interrupt signal conditioning circuitry contains a falling edge detection circuit (a one shot) which requires that in addition to meeting the setup and hold times, the enable interrupt bit EN1 must have been turned on sometime before the end of the WINDOW of OPPORTUNITY shown. If not, the interrupt will be missed and another high to low IN1 transition will be required. EN1 is automatically disabled upon interrupt recognition at point ⑤. Note that although the interrupt is recognized at point ④ it will not be acted upon until all successive transfer of control instructions are executed as defined in the data sheets.

Because of gate delays it is doubtful that if an interrupt had been generated in time to meet the leading ϕ1 edge at point ② that the EN1 enable bit would have been on in time to meet the WINDOW of OPPORTUNITY.

By doing a worst case analysis one can see that in order to guarantee reception of an asynchronous interrupt IN1 must remain low for at least 2 instruction cycles. The analysis is as follows. Assuming that interrupts had been enabled prior to point ①, if the interrupt arrives a little after point ① it will not satisfy the minimum setup requirements bringing us up to a point ⑤ our total elapsed time becomes ⑤ − ① = 2 t_{CYC}.

In a dual COPS the interrupt sequence is the same except that now an instruction cycle time is made up of both a Processor X and a Processor Y instruction execution cycle. With one ϕ1 and ϕ2 clock per processor execution cycle itne instruction cycle time is made up of 2 ϕ1's and ϕ2's. Therefore 1 instruction cycle time in a dual COPS is equivalent to 2 instruction cycle times in a single COPS as far as ϕ1's, ϕ2's and interrupts are concerned.



TL/DD/5180-1

FIGURE 1. COP Interrupt Diagram

Parameter	Min	Typ	Max
t_s	½ t_{CYC}	200 ns	
t_n	½ t_{CYC}	200 ns	
t_{WO}	$-\infty$	½ t_{CYC} − 600 ns	0

Protecting Data in Serial EEPROMs

National Semiconductor
Application Brief 15
Paul Lubeck

National offers a broad line of serial interface EEPROMs which share a common set of features:

- Low cost
- Single supply in all modes (+5V ± 10%)
- TTL compatible interface
- MICROWIRE™ compatible interface
- Read-Only mode or read-write mode

This Application Brief will address protecting data in any of National's Serial Interface EEPROMs by using read-only mode.

Whereas EEPROM is non-volatile and does not require V_{CC} to retain data, the problem exists that stored data can be destroyed during power transitions. This is due to either uncontrolled interface signals during power transitions or noise on the power supply lines. There are various hardware design considerations which can help eliminate the problem although the simplest most effective method may be the following programming method.

All National Serial EEPROMs, when initially powered up are in the Program Disable Mode*. In this mode it will abort any requested Erase or Write cycles. Prior to Erasing or Writing it is necessary to place the device in the Program Enable Mode†. Following placing the device in the Program Enable Mode, Erase and Write will remain enabled until either executing the Disable instruction or removing V_{CC}. Having V_{CC} unexpectedly removed often results in uncontrolled interface signals which could result in the EEPROM interpreting a programming instruction causing data to be destroyed.

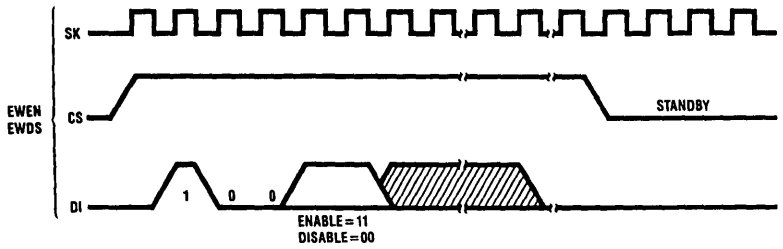
Upon power up the EEPROM will automatically enter the Program Disable Mode. Subsequently the design should incorporate the following to achieve protection of stored data.

1) The device powers up in the read-only mode. However, as a backup, the EWDS instruction should be executed as soon as possible after V_{CC} to the EEPROM is powered up to ensure that it is in the read-only mode.

2) Immediately preceding a programming instruction (ERASE, WRITE, ERAL or WRAL), the EWEN instruction should be executed to enable the device for programming; the EWDS instruction should be executed immediately following the programming instruction to return

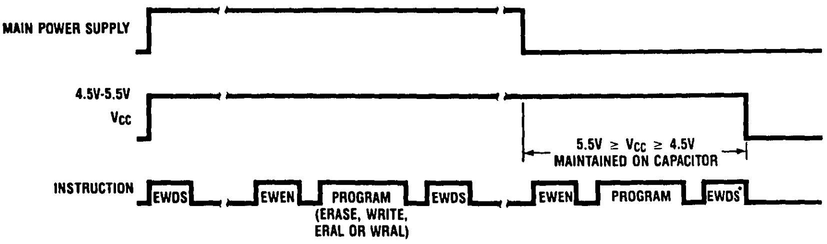
*EWDS or WDS, depending on exact device.

†EWEN or WEN, depending on exact device.



TL/D/7085-1

FIGURE 1. EWEN, EWDS Instruction Timing



TL/D/7085-2

*EWDS must be executed before V_{CC} drops below 4.5V to prevent accidental data loss during subsequent power down and/or power up transients.

FIGURE 2. Typical Instruction Flow for Maximum Data Protection

the device to the read-only mode and protect the stored data from accidental disturb during subsequent power transients or noise.

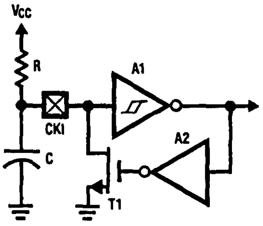
3) Special care must be taken in designs in which programming instructions are initiated to store data in the EEPROM after the main power supply has gone down. This is usually accomplished by maintaining V_{CC} for the EEPROM and its controller on a capacitor for a sufficient amount of time (approximately 50 ms, depending on the clock rate) to complete these operations. This capacitor must be large enough to maintain V_{CC} between 4.5 and 5.5 volts for the total duration of the store operation, INCLUDING the execution of the EWDS instruction immediately following the last programming instruction. FAILURE TO EXECUTE THE LAST EWDS INSTRUCTION BEFORE V_{CC} DROPS BELOW 4.5 VOLTS MAY CAUSE INADVERTENT DATA DISTURB DURING SUBSEQUENT POWER DOWN AND/OR POWER UP TRANSIENTS.

A Users Guide to COPS™ Oscillator Operation

National Semiconductor
Application Note 326
Jim Murashige

The following discussion is an overview of the COPS oscillator circuits meant to give the reader a working knowledge of the circuits. Although the descriptions are very general and light on detail; a background in complex frequency analysis is necessary. For additional information the references cited should be consulted as well as the many works on oscillator theory.

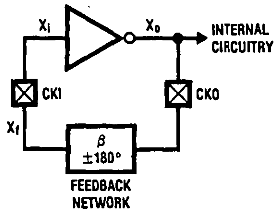
There are 2 basic circuits from which all of the COPS oscillator options are provided. (See option lists in individual data sheets.) The first and simplest in description is the astable one shot of *Figure 1* which gives us our RC oscillator option. A1 and A2 are inverters with A1 possessing a Schmitt trigger input. T1 is a large N channel enhancement MOS FET. Operation with the external R-C shown is as follows. Assuming C is initially discharged the CKI pin is low forcing T1 off. As C charges through R the trigger point of A1 is eventually reached at which time T1 is turned on discharging C and beginning a new cycle. Although almost any combination of R-C could be chosen, we would ideally like to have as short a discharge time as possible thereby eliminating the high variability in T1 drain current from device to device as a timing factor. For this reason R is chosen very large and C very small. This choice also leads to minimum R-C power dissipation. For the CKI Schmitt trigger clock input option the T1 MOS FET is merely mask disabled from the oscillator circuit.



TL/DD/5139-1

FIGURE 1. R-C Oscillator

The second oscillator circuit is the classic phase shift oscillator depicted in *Figure 2*. Found not only on COPS but on most other microprocessor circuits it is the simplest oscillator in terms of component complexity but the most difficult to analyze.



TL/DD/5139-2

FIGURE 2. Phase Shift Oscillator

The conditions under which the circuit will oscillate are described by the Barkhausen Criterion which states that oscillation will occur at the frequency for which the total loop phase shift from x_i to x_f is 0° or a multiple of 360° (i. e., x_f is identical to x_i). In addition the total loop gain must be > 1 to insure self propagation. The inverting amplifier shown between x_i and x_o provides 180° of phase shift thus leaving the feedback network to supply the other ±180°. The feedback network can be comprised of active or passive components but highly effective oscillators are possible using only passive reactive components and the general configuration of *Figure 3*.

If you work out the feedback loop equations for *Figure 3* it can be shown that in order to achieve ±180° phase shift:

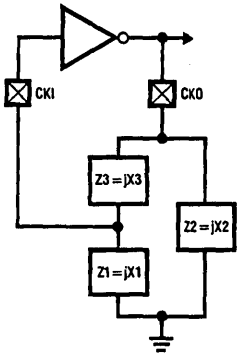
$$X1 + X2 + X3 = 0 \qquad (1)$$

X1 and X2 must both be inductors or capacitors (2)

therefore X3 is inductive if X1 is capacitive and vice versa

if X1 and X2 are capacitors it is a Colpitts Oscillator

X1 and X2 are inductors it is a Hartley Oscillator



TL/DD/5139-3

FIGURE 3. Typical Feedback Configuration

The Colpitts configuration is commonly shown in microprocessor oscillator circuits (*Figure 5*) with the inductive X3 replaced by a crystal for reasons we shall soon see. The equivalent electrical model of a crystal is shown in *Figure 4b* and a plot of its Reactance versus Frequency shown in *Figure 4c*. R-L-C represent the electro-mechanical properties of the crystal and C_0 the electrode capacitance. There are 2 important points on the reactance curve labeled f_a and f_b.

At $f_a = \frac{1}{2\pi}\sqrt{\frac{1}{LC}}$

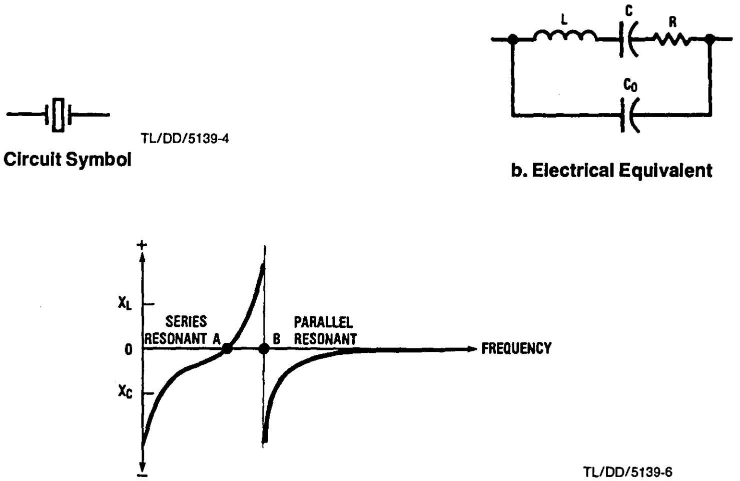
the crystal is at series resonance with L and C canceling each other out leaving only a nonreactive R for 0 phase shift. This mode of operation is important in oscillator circuits where a non-inverting amplifier is used and 0° phase shift must be preserved.

At $f_b = \frac{1}{2\pi}\sqrt{\frac{1}{LC} + \frac{1}{LC_C}}$

which is just a little higher than f_a the crystal is at parallel resonance and appears very inductive or capacitive. Note that the cyrstal will only appear inductive between f_a and f_b and that it becomes highly inductive very quickly. In addition f_b is only a fraction of a percent higher than f_a. Therefore the only time that the crystal will satisfy the $X3 = -(X1 + X2)$ condition in the Colpitts configuration of *Figure 5* is when the circuit is oscillating between f_a and f_b. The exact frequency will be the one which gives an inductive reactance large enough to cancel out:

$$X1 + X2 = \frac{1}{\omega C1} + \frac{1}{\omega C2} = \frac{1}{\omega}\left[\frac{1}{C1} + \frac{1}{C2}\right] = \frac{1}{2\pi f}\left[\frac{1}{C_L}\right]$$

Therefore by varying C1 or C2 we can trim slightly the oscillator frequency.



TL/DD/5139-4

a. Circuit Symbol

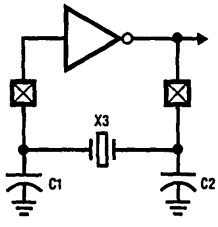
TL/DD/5139-5

b. Electrical Equivalent

TL/DD/5139-6

c. Reactance Versus Frequency

FIGURE 4. Quartz Crystal

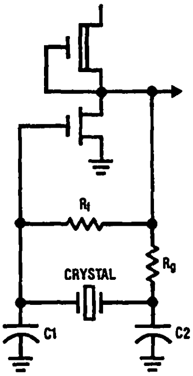


TL/DD/5139-7

FIGURE 5. Colpitts Oscillator

The Q of a circuit is often bounced around in comparing different circuits and can be viewed graphically here as the slope of the reactance curve between f_a and f_b. Obviously the steeper the curve the smaller the variation in f necessary to restore the Barkhausen Phase Shift Criterion. In addition a lower Q (more R) means that the reactance curve won't peak as high at f_b, necessitating a smaller X1 + X2. When selecting crystals the user should be aware that the frequency stamped on the cans are for either parallel or series resonance, which, although very close, may matter significantly in the particular application.

An actual MOS circuit implementation of *Figure 5* is shown in *Figure 6*. It consists of a MOS inverter with depletion load and the crystal π network just presented. External to the COPS chips are the R_f and R_g resistors. R_f provides bias to the MOS inverter gate $V_g = V_o$. Since the gate draws no current R_f can be very large (MΩ) and should be, since we do not wish it to interact with the crystal network. R_g increases the output resistance of the inverter and keeps the crystal from being over driven.

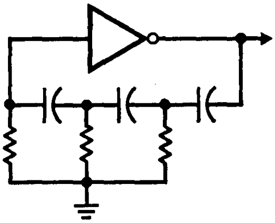


TL/DD/5139-8

FIGURE 6. MOS Oscillator

Of course the feedback network doesn't have to have the configuration of *Figure 3* and can be anything so long as the Barkhausen Phase Shift Criterion is satisfied. One popular configuration is shown in *Figure 7* where the phase shift will be 180°

$$\text{at } f = \frac{1}{(2\pi RC\sqrt{6})}$$



TL/DD/5139-9

FIGURE 7. R-C Phase Shift Oscillator

REFERENCES

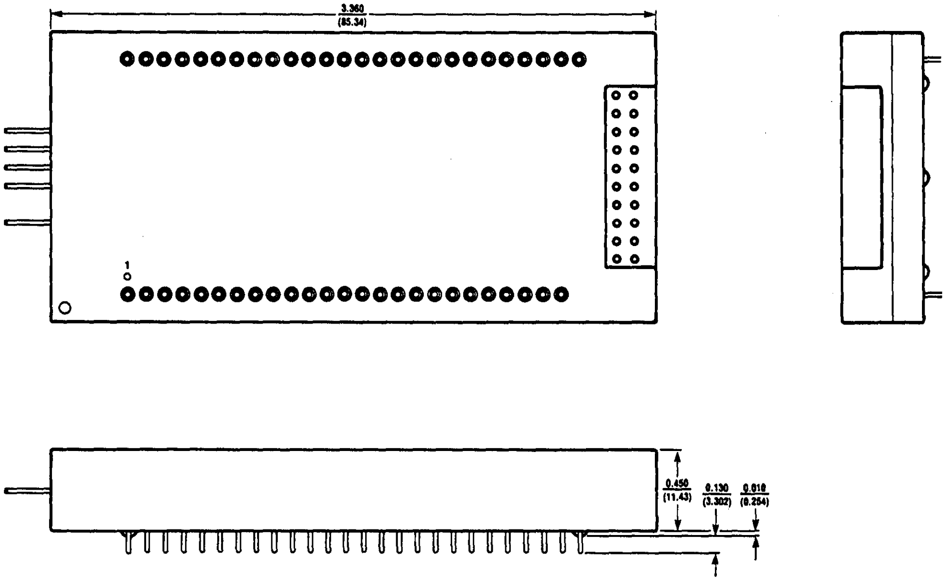
1. Crystal/INS8048 Oscillator, AN-296, March 1982, National Semiconductor

2. Oscillator Characteristics of COPS Microcontrollers, CN-5, Feb. 1981, National Semiconductor

3. Integrated Electronics, Chapter 14, Millman and Halkias 1972

4. Handbook of Electronics Calculations, Chapter 9, Kaufman and Seidman 1979

5. 1982 COPS Microcontroller Databook, National Semiconductor



TL/DD/5139-10

Implementing an 8-Bit Buffer in COPS™

National Semiconductor
Application Note 329
David Pointer

Sometimes a COP microcontroller must input and/or output 8-bit data; for instance, when handling ASCII data. In some applications, the processor must also provide temporary storage for 8-bit data before it is output. The COP instruction set and RAM structure lend themselves very nicely to providing a 32 digit, 8-bit buffer for a solution to these applications.

Such a large buffer is possible using a COP440 or a COP444L. The other members of the COP400 family with half as much RAM as these two would provide a 16 digit 8-bit buffer using the techniques described in this example.

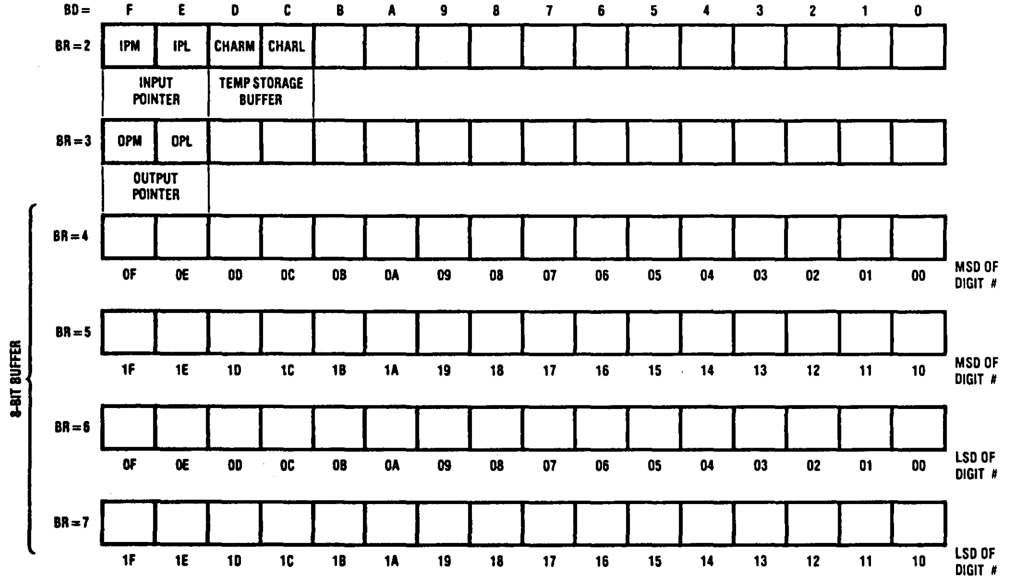
Four adjacent RAM registers (16 digits each) are required. Referring to *Figure 1*, registers 4, 5, 6, and 7 are used for the buffer. Each RAM location contains 4 bits, so 2 locations will be used to store a byte of data. But these RAM locations are not adjacent to each other. You will note that the MSD of digit number 0A hex is in RAM location (4, A) while the LSD of the same digit is in RAM location (6, A).

The 2 RAM locations CHARM and CHARL are used for temporary storage of an 8-bit value.

In addition, 4 RAM locations are used for buffer pointers: those labelled IPM and IPL are the MSD and LSD of the input pointer, and those labelled OPM and OPL are the MSD and LSD of the output pointer. Each pointer's function is to store an 8-bit counter whose value ranges from 00 hex thru 1F hex. The input pointer's value is used for storing the temporary storage buffer contents into the digit with the same number. For example, if the input pointer equals 14 hex, then the contents of CHARM would be stored in RAM location (5, 4) and the contents of CHARL would be stored in RAM location (7, 4). The output pointer's value is used for retrieving a digit from the buffer and putting it in CHARM and CHARL. For instance, if the output pointer equals 05 hex, then the contents of RAM location (4, 5) would be transferred to CHARM and the contents of RAM location (6, 5) would be transferred to CHARL.

A simple example of one possible application of the buffer is flowcharted in *Figure 2.* In this example, data is input to CHARM and CHARL, then stored in the buffer. An output device (a printer) is checked to see if it is ready to receive data. If it is, data is brought out of the buffer and put in CHARM and CHARL for output to the printer.

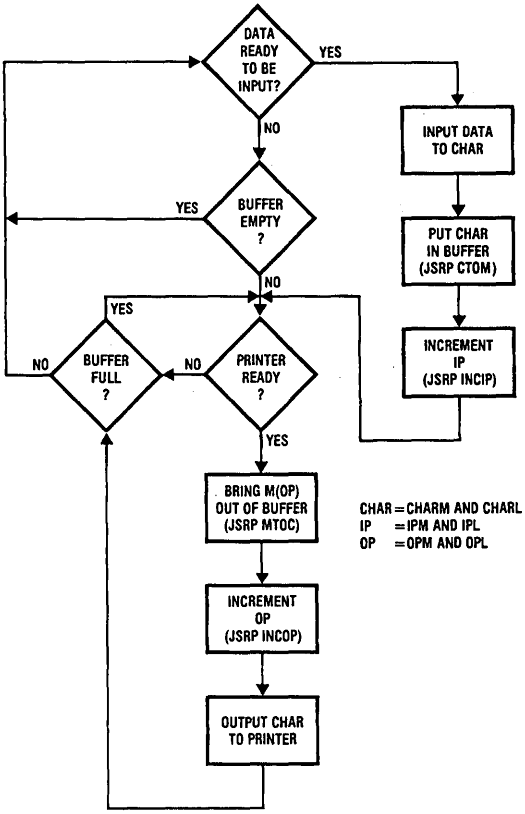
Pages 3 and 4 contain a listing of the subroutines needed to perform the data transfers in the 32-digit, 8-bit buffer.



BD=	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
BR=2	IPM	IPL	CHARM	CHARL													
	INPUT POINTER		TEMP STORAGE BUFFER														
BR=3	OPM	OPL															
	OUTPUT POINTER																
BR=4																	
	0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00	MSD OF DIGIT #
BR=5																	
	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	MSD OF DIGIT #
BR=6																	
	0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	00	LSD OF DIGIT #
BR=7																	
	1F	1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	LSD OF DIGIT #

TL/DD/5181-1

FIGURE 1. 8-Bit Buffer RAM Map



TL/DD/5181-2

FIGURE 2. Buffer Example Flowchart

```
COP CROSS ASSEMBLER    PAGE:  1
BUFFER

             ;*************************************
             ;***                               ***
             ;***  8-BIT RAM BUFFER SUBROUTINES  ***
             ;***                               ***
             ;*************************************
             ;THESE ARE SUBROUTINES FOR IMPLEMENTING A 32 BYTE
             ;BUFFER IN A COP440 OR COP444L RAM  9/3/82
    01BC     .CHIP 444
             .TITLE BUFFER
    002D     CHARM   =       2,13            ;TEMPORARY STORAGE BUFFER MSD
    002C     CHARL   =       2,12            ;TEMPORARY STORAGE BUFFER LSD
    002F     IPM     =       2,15            ;INPUT POINTER MSD
    002E     IPL     =       2,14            ;INPUT POINTER LSD
    003F     OPM     =       3,15            ;OUTPUT POINTER MSD
    003E     OPL     =       3,14            ;OUTPUT POINTER LSD
000 00               CLRA
    0080     .PAGE 2
             ;MTOC IS A SUBROUTINE THAT TRANSFERS M(OPM) AND M(OPL) TO
             ;CHARM AND CHARL
080 233E     MTOC:   LDD     OPL             ;LOAD LSD OUTPUT POINTER
082 50               CAB                     ;WHICH IS BD
083 233F             LDD     OPM              ;LOAD MSB OUTPUT POINTER FOR B
085 54               AISC    4                ;MAKE BR EQUAL 4 OR 5
086 12               XABR
087 25               LD      2               ;LOAD M(OPM), MAKE BR = 6 OR 7
088 23AD             XAD     CHARM           ;M(OPM) TO CHARM
08A 05               LD                       ;LOAD M(OPL)
08B 23AC             XAD     CHARL           ;M(OPL) TO CHARL
08D 48               RET
             ;
             ;
             ;CTOM IS A SUBROUTINE THAT TRANSFERS CHARM AND CHARL TO
             ;M(IPM) AND M(IPL)
08E 232E     CTOM:   LDD     IPL             ;LOAD LSD INPUT POINTER
090 50               CAB                     ;WHICH IS BD
091 232F             LDD     IPM             ;LOAD MSD INPUT POINTER FOR BR
093 54               AISC    4               ;MAKE BR = 4 OR 5
094 12               XABR
095 232D             LDD     CHARM           ;LOAD MSD TEMP STORAGE
097 26               X       2               ;TO M(OPM), MAKE BR = 6 OR 7
098 232C             LDD     CHARL           ;LOAD LSD TEMP STORAGE
09A 06               X                       ;TO M(OPL)
09B 48               RET
             ;
             ;
```

```
COP CROSS ASSEMBLER    PAGE:  2
BUFFER

   46            .FORM
   47            ;INCREMENTS INPUT POINT OR OUTPUT POINTER, ROLLS OVER
   48            ;AT 1F HEX
   49 09C 2D     INCIP:  LBI     IPL              ;POINT TO LSD OF POINTER
   50 09D 3D     INCOP:  LBI     OPL
   51 09E 22             SC                       ;C=1 FOR INCREMENT
   52 09F 00             CLRA
   53 0A0 30             ASC                      ;INCREMENT RAM VALUE
   54 0A1 44             NOP                      ;NEGATES SKIP CONDITION
   55 0A2 04             XIS                      ;STORE AND POINT TO (X,F)
   56 0A3 00             CLRA
   57 0A4 30             ASC                      ;PROPAGATE CARRY, IF ANY, TO MS
   58 0A5 44             NOP
   59 0A6 06             X                        ;STORE
   60 0A7 45             RMB     1                ;ROLL OVER AT X'1F
   61 0A8 48             RET
   62            ;
   63            ;
   64            .END


COP CROSS ASSEMBLER    PAGE:  3
BUFFER

CHARL  002C     CHARM  002D     CTOM   008E *   INCIP  009C *
INCOP  009D *   IPL    002E     IPM    002F     MTOC   0080 *
OPL    003E     OPM    003F
NO ERROR LINES
   42  ROM WORDS USED
COP   444  ASSEMBLY
SOURCE CHECKSUM = C6A5
INPUT FILE    6:RBUFFC. SRC  VN:    5
```

Designing with the NMC9306/COP494 a Versatile Simple to Use E² PROM

National Semiconductor
Application Note 338
Masood Alavi

This application note outlines various methods of interfacing an NMC9306/COP494 with the COPS™ family of microcontrollers and other microprocessors. *Figures 1–6* show pin connections involved in such interfaces. *Figure 7* shows how parallel data can be converted into a serial format to be inputted to the NMC9306; as well as how serial data outputted from an NMC9306 can be converted to a parallel-format.

The second part of the application note summarizes the key points covering the critical electrical specifications to be kept in mind when using the NMC9306/COP494.

The third part of the application note shows a list of various applications that can use a NMC9306/COP494.

GENERIC CONSIDERATIONS

A typical application should meet the following generic criteria:

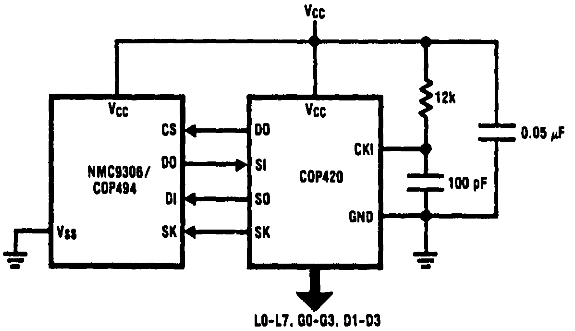
1. Allow for no more than 10,000 E/W cycles for optimum and reliable performance.
2. Allow for any number of read cycles.
3. Allow for an erase or write cycle that operates in the 10–30 ms range, and not in the tens or hundreds of ns range as used in writing RAMs. (Read vs write speeds are distinctly different by orders of magnitude in E²PROM, not so in RAMs.)
4. No battery back-up required for data-retention, which is fully non-volatile for at least 10 years at room-ambient.

SYSTEM CONSIDERATIONS

When the control processor is turned on and off, power supply transitions between ground and operating voltage may cause undesired pulses to occur on data, address and control lines. By using WEEN and WEDS instructions in conjunction with a LO-HI transition on CS, accidental erasing or writing into the memory is prevented.

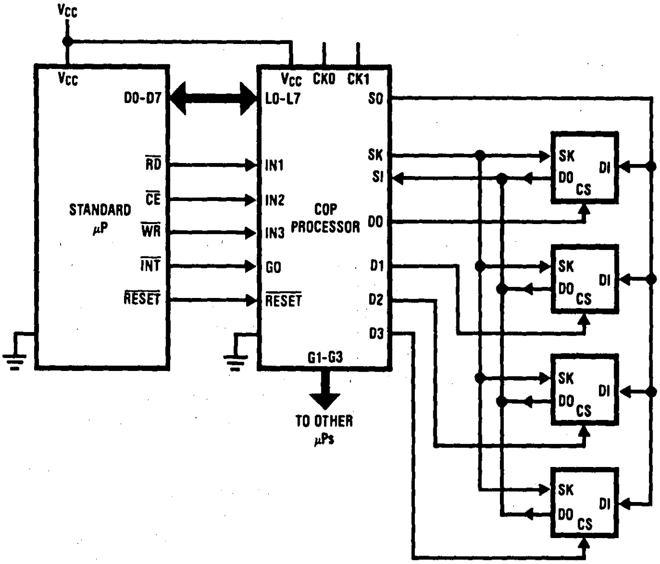
The duty cycle in conjunction with the maximum frequency translates into having a minimum Hi-time on the SK clock. If the minimum SK clock high time is greater than 1 µs, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max. On the low side no limit exists on the minimum frequency. This makes it superior to the COP499 CMOS-RAM. The rise and fall times on the SK clock can also be slow enough not to require termination up to reasonable cable-lengths.

Since the device operates off of a simple 5V supply, the signal levels on the inputs are non-critical and may be operated anywhere within the specified input range.



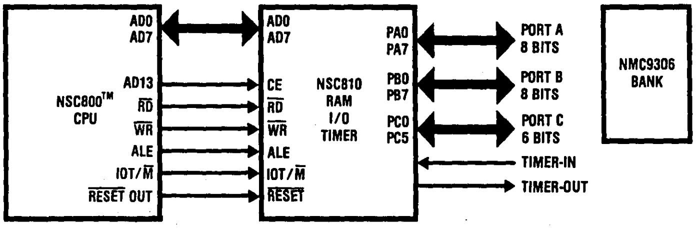
TL/D/5286-1

FIGURE 1. NMC9306/COP494 — COP420 Interface



TL/D/5286-2

FIGURE 2. NMC9306 — Standard μP Interface Via COP Processor



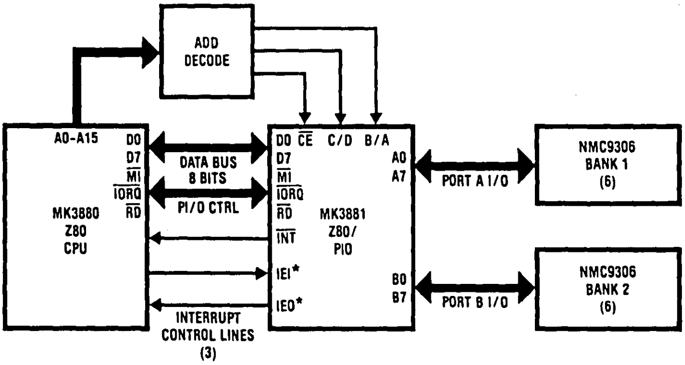
TL/D/5286-3

PA0 → SK } Common to all 9306's
PA1 → DI/DO }
PA2-7 → 6CS for 6- 9306's

* SK is generated on port pins by bit-set and bit-clear operations in software. A symmetrical duty cycle is not critical.

* CS is set in software. To generate 10-30 ms write/erase the timer/counter is used. During write/erase. SK may be turned off.

FIGURE 3. NSC800™ to NMC9306 Interface (also Valid for 8085/8085A and 8156)



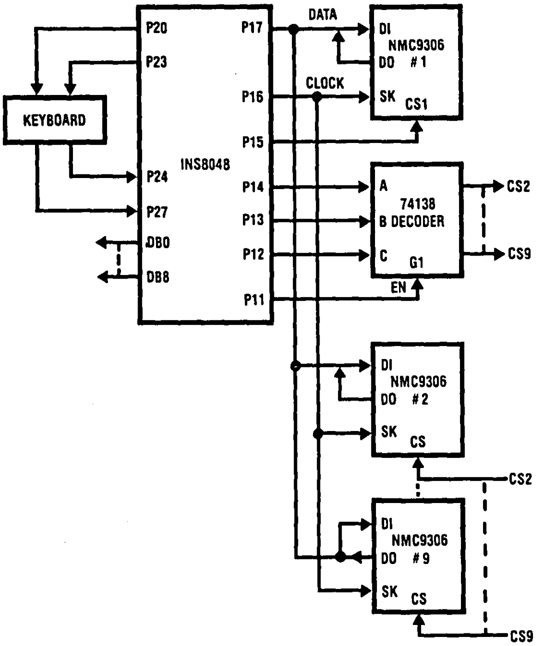
TL/D/5286-4

Z80-P10	9306	
A0	SK	Common to all 9306's (Bank 1)
A1	DI/DO	
A2-A7	CS1-CS6	

* Only used if priority interrupt daisy chain is desired

* Identical connection for Port B

FIGURE 4. Z80 — NMC9306 Interface Using Z80-PIO Chip

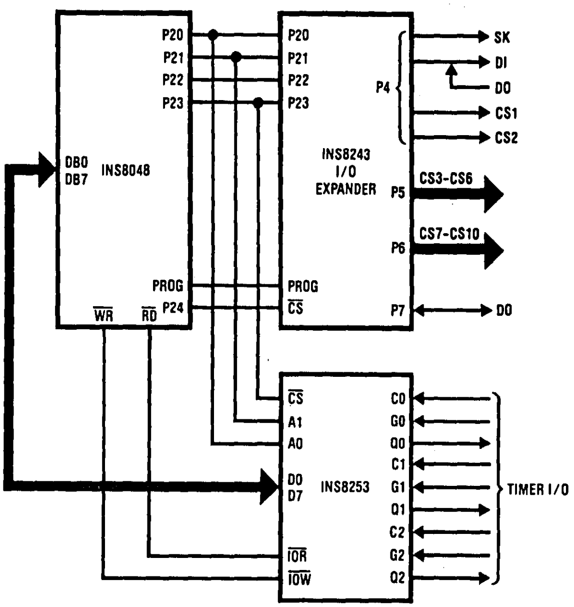


TL/D/5286-5

* SK and DI are generated by software. It should be noted that at 2.72 μs/instruction. The minimum SK period achievable will be 10.88 μs or 92 kHz, well within the NMC9306 frequency range.

* DO may be brought out on a separate port pin if desired.

FIGURE 5. 48 Series μP — NMC9306 Interface

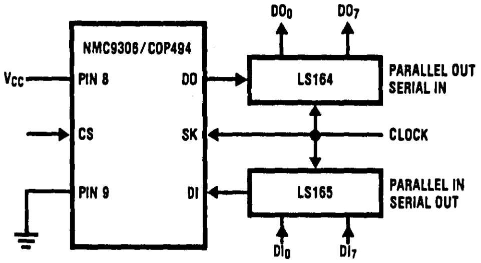


TL/D/5286-6

Expander outputs

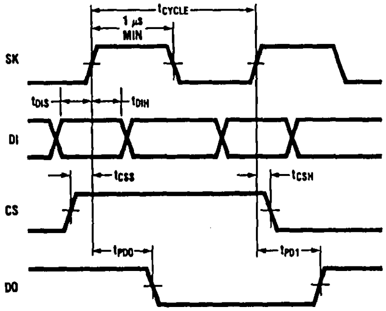
	DI SK	(COMMON)
Port 4	CS1 CS2	
Port 5-6	CS3–CS10	
Port 7	DO (COMMON)	

FIGURE 6. 8048 I/O Expansion



TL/D/5286-7

FIGURE 7. Converting Parallel Data into Serial Input for NMC9306/COP494



Min	Max
t_{CYCLE} 0	250 kHz
t_{DIS} 400	ns
t_{D1H} 400	ns
t_{CSS} 200	ns
t_{CSH} 0	ns
t_{PD0}	2 μs
t_{PD1}	2 μs

TL/D/5286-8

FIGURE 8. NMC9306/COP494 Timing

THE NMC9306/COP494

Extremely simple to interface with any μP or hardware logic. The device has six pins for the following functions:

Pin 1	CS*	HI enabled
Pin 2	SK	Serial Clock input
Pin 3	DI	For instruction or data input
Pin 4	DO**	For data read, TRI-STATE® otherwise
Pin 5	GND	
Pin 8	V_{CC}	For 5V power
Pins 6–7	No Connect	No termination required

*Following an E/W instruction feed, CS is also toggled low for 10 ms (typical) for an E/W operation. This internally turns the VPP generator on (HI-LO on CS) and off (LO-HI on CS).

**DI and DO can be on a common line since DO is TRI-STATED when unselected DO is only on in the read mode.

USING THE NMC9306/COP494

The following points are worth noting:

1. SK clock frequency should be in the 0–250 kHz range. With most μPs this is easily achieved when implemented in software by bit-set and bit-clear instructions, which take 4 instructions to execute a clock or a frequency in the 100 kHz range for standard μP speeds. Symmetrical duty cycle is irrelevant if SK HI time is ≥ 2 μs.
2. CS low period following an E/W instruction must not exceed the 30 ms max. It should best be set at typical or minimum spec of 10 ms. This is easily done by timer or a software connect. The reason is that it minimizes the 'on time' for the high V_{PP} internal voltage, and so maximizes endurance. SK-clock during this period may be turned off if desired.
3. All E/W instructions must be preceded by EWEN and should be followed by an EWDS. This is to secure the stored data and avoid inadvertent erase or write.
4. A continuously 'on' SK clock does not hurt the stored data. Proper sequencing of instructions and data on DI is essential to proper operation.
5. Stored data is fully non-volatile for a minimum of ten years independent of V_{CC}, which may be on or off. Read cycles have no adverse effects on data retention.
6. Up to 10,000 E/W cycles/register are possible. Under typical conditions, this number may actually approach 1 million. For applications requiring a large number of cycles, redundant use of internal registers beyond 10,000 cycles is recommended.
7. Data shows a fairly constant E/W Programming behavior over temperature. In this sense E^{2}PROMs supersede EPROMs which are restricted to room temperature programming.
8. As shown in the timing diagrams, the start bit on DI must be set by a ZERO - ONE transition following a CS enable (ZERO - ONE), when executing any instruction. ONE CS enable transition can only execute ONE instruction.
9. In the read mode, following an instruction and data train, the DI can be a don't care, while the data is being outputted i.e., for next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
10. The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI till another CS LO-HI transition starts a new instruction cycle.
11. When a common line is used for DI and DO, a probable overlap occurs between the last bit on DI and start bit on DO.
12. After a read cycle, the CS must be brought low for 1 SK clock cycle before another instruction cycle can start.

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ — After read command is shifted in
DI becomes don't care and data can
be read out on data out, starting
with dummy bit zero.

WRITE — Write command shifted in followed by
data in (16 bits) then CS pulsed low
for 10 ms minimum.

INSTRUCTION SET

Instruction	SB	Opcode	Address	Data	Comments
READ	01	10xx	A3A2A1A0		Read Register A3A2A1A0
WRITE	01	01xx	A3A2A1A0	D15–D0	Write Register A3A2A1A0
ERASE	01	11xx	A3A2A1A0		Erase Register A3A2A1A0
EWEN	01	0011	XXXX		Erase/Write Enable
EWDS	01	0000	XXXX		Erase/Write Disable
ERAL	01	0010	XXXX		Erase All Registers
WRAL	01	0001	XXXX	D15–D0	Write All Registers

NMC9306 has 7 instructions as shown. Note that MSB of any given instruction is a "1" and is viewed as a start bit in the interface sequence. The next 8 bits carry the op code and the 4-bit address for 1 of 16, 16-bit registers.

X is a don't care state.

The following is a list of various systems that could use a NMC9306/COP494

A. Airline terminal
Alarm system
Analog switch network
Auto calibration system
Automobile odometer
Auto engine control
Avionics fire control
B. Bathroom scale
Blood analyzer
Bus interface
C. Cable T.V. tuner
CAD graphics
Calibration device
Calculator—user programmable
Camera system
Code identifier
Communications controller
Computer terminal
Control panel
Crystal oscillator
D. Data acquisition system
Data terminal
E. Electronic circuit breaker
Electronic DIP switch
Electronic potentiometer
Emissions analyzer
Encryption system
Energy management system
F. Flow computer
Frequency synthesizer
Fuel computer
G. Gas analyzer
Gasoline pump
H. Home energy management
Hotel lock
I. Industrial control
Instrumentation
J. Joulemeter
K. Keyboard -softkey
L. Laser machine tool
M. Machine control
Machine process control
Medical imaging
Memory bank selection
Message center control
Mobile telephone
Modem
Motion picture projector
N. Navigation receiver
Network system
Number comparison
O. Oilfield equipment
P. PABX
Patient monitoring
Plasma display driver
Postal scale
Process control
Programmable communications
Protocol converter
Q. Quiescent current meter
R. Radio tuner
Radar dectector
Refinery controller
Repeater
Repertory dialer
S. Secure communications system
Self diagnostic test equipment
Sona-Bouy
Spectral scanner
Spectrum analyzer
T. Telecommunications switching system
Teleconferencing system
Telephone dialing system
T.V. tuner
Terminal
Test equipment
Test system
TouchTone dialers
Traffic signal controller
U. Ultrasound diagnostics
Utility telemetering
V. Video games
Video tape system
Voice/data phone switch
W. Winchester disk controller
X. X-ray machine
Xenon lamp system
Y. YAG—laser controller
Z. Zone/perimeter alarm
system

A Study of the Crystal Oscillator for CMOS-COPS™

National Semiconductor
Application Note 400
Abdul Aleaf

INTRODUCTION

The most important characteristic of CMOS-COPS is its low power consumption. This low power feature does not exist in TTL and NMOS systems which require the selection of low power IC's and external components to reduce power consumption.

The optimization of external components helps decrease the power consumption of CMOS-COPS based systems even more.

A major contributor to power consumption is the crystal oscillator circuitry.

Table I presents experimentally observed data which compares the current drain of a crystal oscillator vs. an external squarewave clock source.

The main purpose of this application note is to provide experimentally observed phenomena and discuss the selection of suitable oscillator circuits that cover the frequency range of the CMOS-COPS.

Table I clearly shows that an unoptimized crystal oscillator draws more current than an external squarewave clock. An RC oscillator draws even more current because of the slow rising signal at the CKI input.

Although there are few components involved in the design of the oscillator, several effects must be considered. If the requirement is only for a circuit at a standard frequency which starts up reliably regardless of precise frequency stability, power dissipation and etc., then the user could directly consult the data book and select a suitable circuit with proper components. If power consumption is a major requirement, then reading this application note might be helpful.

WHICH IS THE BEST OSCILLATOR CIRCUIT?

The Pierce Oscillator has many desirable characteristics. It provides a large output signal and drives the crystal at a low power level. The low power level leads to low power dissipation, especially at higher frequencies. The circuit has good short-term stability, good waveforms at the crystal, a frequency which is independent of power supply and temperature changes, low cost and usable at any frequency. As compared with other oscillator circuits, this circuit is not disturbed very much by connecting a scope probe at any point in the circuit, because it is a stable circuit and has low impedance. This makes it easier to monitor the circuit without any major disturbance. The Pierce oscillator has one disadvantage. The amplifier used in the circuit must have high gain to compensate for high gain losses in the circuitry surrounding the crystal.

TABLE I

A. Crystal oscillator vs. external squarewave COP410C change in current consumption as a function of frequency and voltage, chip held in reset, CKI is ÷4.

I = total power supply current drain (at V_{CC}).

Crystal

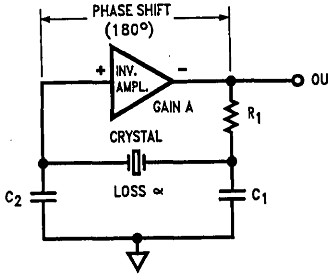
V_{CC}	f_{CKI}	Inst. cyc. time	I μA
2.4V	32 kHz	125 μs	8.5
5.0V	32 kHz	125 μs	83
2.4V	1 MHz	4 μs	199
5.0V	1 MHz	4 μs	360

External Squarewave

V_{CC}	f_{CKI}	Inst. cyc. time	I
2.4V	32 kHz	125 μs	4.4 μA
5.0V	32 kHz	125 μs	10 μA
2.4V	1 MHz	4 μs	127 μA
5.0V	1 MHz	4 μs	283 μA

WHAT IS A PIERCE OSCILLATOR?

The Pierce is a series resonant circuit, and its basic configuration is shown below.



TL/DD/8439-1

FIGURE 1

For oscillation to occur, the Barkhausen criteria must be met: (1) The loop gain must be greater than one. (2) The phase shift around the loop must be 360°.

Ideally, the inverting amplifier provides 180°, the R_1C_1 integration network provides a 90° phase lag, and the crystal's impedance which is a pure resistance at series resonance together with C_2 acts as a second integration network which provides another 90° phase lag. The time constants of the two RC phase shifting networks should be made as big as possible. This makes their phase shifts independent of any changes in resistance or capacitance values. However, big RC values introduce large gain losses and the selected amplifier should provide sufficient gain to satisfy gain requirement. CMOS inverters or discrete transistors can be used as amplifiers. An experimental evaluation of crystal oscillators using either type of amplifier is given within this report.

CRYSTAL OSCILLATORS USING CMOS-IC

The use of CMOS-IC's in crystal oscillators is quite popular. However, they are not perfect and could cause problems. The input characteristics of such IC's are good, but they are limited in their output drive capability.

The other disadvantage is the longer time delay in a CMOS-inverter as compared to a discrete transistor. The longer this time delay the more power will be dissipated. This time delay is also different among different manufacturers.

As a characteristic of most CMOS-IC's the frequency sensitivity to power supply voltage changes is high. As a group, IC's do not perform very well when compared with discrete transistor circuits.

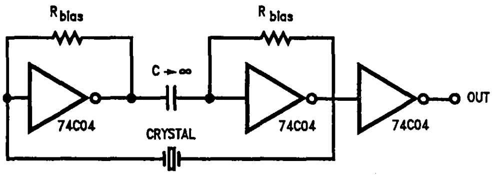
But let us not be discouraged. Low component count which leads to low cost is one good feature of IC oscillators.

As a rule, IC's work best at the low end of their frequency range and poorest at the high end.

Several types of crystal oscillators using CMOS-IC's have been found to work satisfactorily in some applications.

CMOS—TWO INVERTER OSCILLATOR

The two inverter circuit shown in *Figure 2* is a popular one. The circuit is series resonant and uses two cascaded inverters for an amplifier.



TL/DD/8439–2

FIGURE 2

Each inverter has a DC biasing resistor which biases the inverter halfway between the logic "1" and "0" states. This will help the inverters to amplify when the power is applied and the crystal will start oscillation.

The 74C family works better as compared with other CMOS-IC's. Will oscillate at a higher frequency and is less sensitive to temperature changes. The CMOS-COPS data sheet states that a crystal oscillator will typically draw 100 μA more than an external clock source. However, the crystal oscillator described above will draw approximately as much current as an external squarewave clock. The experimental data presented below shows the comparison:

Chip held in Reset, $V_{CC} = +5.0V$

f = 455 kHz, COP444C, CKI is ÷8

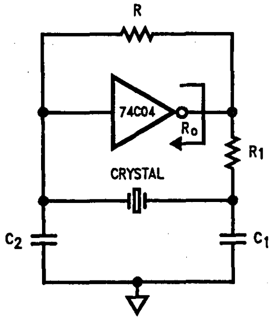
Instruction cycle time = 17.5 μs

I = total power supply (V_{CC}) current drain

Oscillator Type	I (current drain)
Crystal Osc. (data sheet)	950 μA
Crystal Osc. (two inverter)	810 μA
Ext. Clock	790 μA

PIERCE IC OSCILLATOR

Figure 3 shows a Pierce oscillator using CMOS inverter as an amplifier.



TL/DD/8439–3

FIGURE 3

The gain of CMOS inverter is low, so the resistor R_1 should be made small. This reduces gain losses. The output resistance of the inverter (Ro) can be the integrating resistor for the RoC_1 phase lag network.

Omitting R_1 or with a small value of R_1, the crystal will be driven at a much higher voltage level. This will increase power dissipation.

For lower frequencies (i.e., 32 kHz), R_1 must be large enough so that the inverter won't overdrive the crystal. Also, if R_1 is too large we won't get an adequate signal back at the inverter's input to maintain oscillation. With large values of R_1 the inverter will remain in its linear region longer and will cause more power dissipation. Typically for 32 kHz, R_1 should be constrained by the relation.

$$\frac{1}{2\pi R_1C_1} \leqslant 32\text{ kHz}$$

At higher frequencies, selection of R_1 is again critical. In order to drive a heavy load at high frequency, the amplifier output impedance must be low. In order to isolate the oscillator output from C_1 so it can drive the following logic stages, then R_1 should be large. But again, R_1 must not be too large, otherwise it will reduce the loop gain.

The value of R_1 is chosen to be roughly equal to the capacitive reactance of C_1 at the frequency of operation, or the value of load impedance Z_L.

$$\text{Where } Z_L = \frac{X_{c1}^2}{R_L}$$

$$R_L = R_S = \text{series resistance of crystal}$$

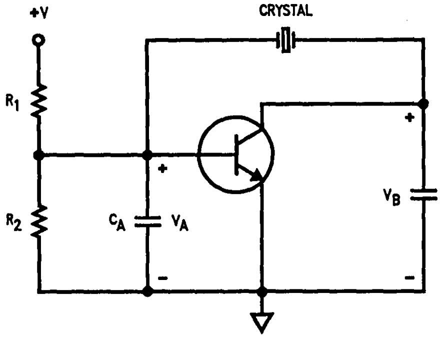
The small values of C_1 and C_2 will help minimize the gain reduction they introduce.

$$\text{typically: } C_1 = C_2 = 220\text{ pF at 1 MHz}$$
$$C_1 = C_2 = 330\text{ pF at 2 MHz}$$

DISCRETE TRANSISTOR OSCILLATOR

As mentioned earlier, a discrete transistor circuit performs better than an IC circuit. The reason for this is that in a discrete transistor circuit it is easier to control the crystal's source and load resistances, the gain and signal amplitude.

A discrete transistor circuit has shorter time delay, because it uses one or two transistors. This time delay should always be minimized, since it causes more power dissipation and shifts frequency with temperature changes. *Figure 4* shows a basic Pierce oscillator using a transistor as an amplifier.



TL/DD/8439-4

FIGURE 4

The basic phase shift network consists of C_{A_1}, C_{B_2} and the crystal which looks inductive and is series resonant with C_{A_1} and C_{B_1}. The phase shift through the transistor is 180° and the total phase shift around the loop is 360°. The condition of a unity loop gain must also be satisfied.

$$\frac{V_A}{V_B} = -\left(\frac{C_B}{C_A}\right)$$

$$\frac{V_A}{V_B} = -\left(\frac{X_{CA}}{X_{CB}}\right)$$

For oscillation to occur, the transistor gain must satisfy the relation

$$G\left(\frac{V_A}{V_B}\right) \geq, 1$$

where $G = -g_{fe}Z_L$

g_{fe} is the transconductance of the transistor

Z_L is the load seen by the collector

$$Z_L = \frac{X_B^2}{Re}, \quad X_B = -\frac{1}{WC_B}$$

Re is the crystal's effective series resistance.

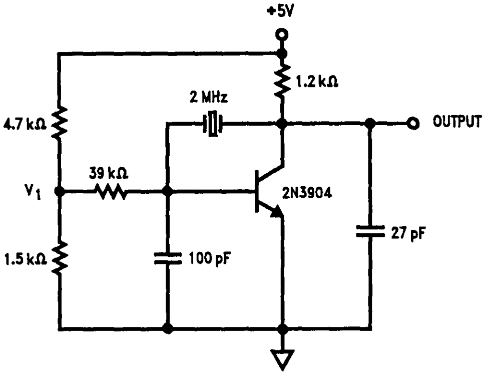
The crystal's drive level

$$P_d = \frac{V_B 2_{Re}}{X_B 2}$$

This drive level should not exceed the manufacturer's spec.

Certain biasing conditions might cause collector saturation. Collector saturation increases oscillator's dependence on the supply voltage and should be avoided.

The circuit of *Figure 5* has been tested and has a very good performance.



TL/DD/8439-5

FIGURE 5

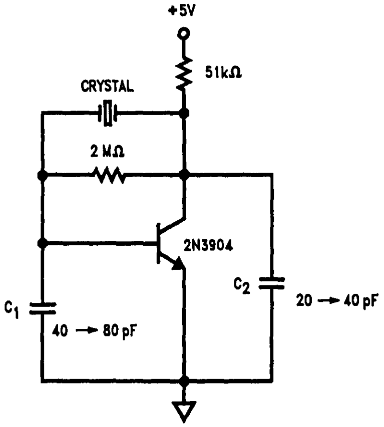
This circuit will oscillate over a wide range of frequencies 2–20 MHz.

$$\text{Voltage } (V_1) = \frac{(5)\,(1.5)}{1.5 + 4.7} = 1.21\text{V}$$

$$\text{Base Current} = \frac{1.21\text{-}V_{BE}}{39\text{k}} = 15.6\ \mu\text{A}$$

At Saturation ($V_{CE} = 0$)

$$I_{C\,(SAT)} = \frac{5}{1.2} = 4.2\text{ mA}$$



TL/DD/8439-6

FIGURE 6

Having 15.6 μA of base current, for saturation to occur

$$h_{FE} = \frac{4.2\ \text{mA}}{15.6\ \mu\text{A}} = 269$$

The DC beta for 3904 at 1 mA is 70 to 210, so no problem with saturation, even at lower supply voltages.

The current consumption (power supply V_{CC} current drain) of COP444C using the above oscillation circuit is around 267 μA.

The circuit of *Figure 6* is another configuration of discrete transistor oscillator.

The performance of above circuit is also good. The only drawback is that it does not provide larger output signal.

CONCLUSION

As discussed within this report, a discrete transistor circuit gives better performance than an IC circuit. However, oscillators using discrete transistors are more expensive than those using IC's when assembly labor costs are included. So, the selection of either circuit is a trade-off between better performance and cost.

The data and circuits presented here are intended to be used only as a guide for the designer. The networks described are generally simple and inexpensive and have all been observed to be functional. They only provide greater flexibility in the oscillator selection for CMOS-COPS systems.

Selecting Input/Output Options On COPS™ Microcontrollers

National Semiconductor
Application Note 401
Abdul Aleaf

INTRODUCTION

There are a variety of user selectable input and output options available on COPS when the ROM is masked. These options are available to help the user tailor the I/O characteristics of the Microcontroller to the application. This application note is intended to provide the user a guide to the options: What are they? When and how to use which ones? The paper is generally written without reference to a specific device except when examples are given. It must be remembered that any given generic COPS Microcontroller has a subset of all the possible options available and that a given pin might not have all possible options. A reference to the device data sheet will determine which options are available for a specific device and a specific pin of that device.

INPUT/OUTPUT OPTIONS

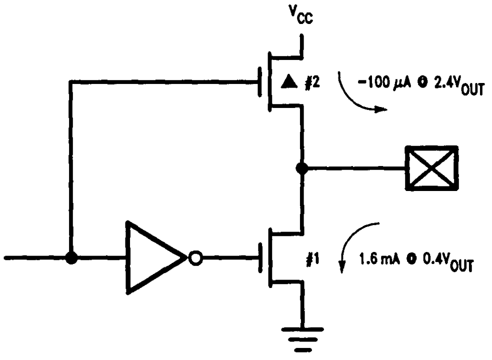
Table I summarizes the I/O capability of NMOS-COPS, in general. However, some of the options have different configuration in CMOS-COPS. Data sheets provide information on the I/O options associated with the CMOS-COPS.

I. OUTPUTS

The following discussion provides detailed information on the capabilities of the mask-programmable output options available on COPS.

A. STANDARD OUTPUT

This option is a simple, straightforward, logic compatible output used for simple logic interface. It is available on SO, SK and all D and G outputs, It is recommended to be used as a default option for all but SO, SK outputs.

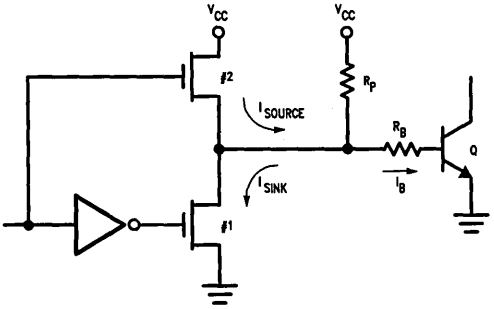


TL/DD/8440-1

FIGURE 1. Standard Output

Figure 1 shows the standard output configuration. The enhancement mode device to ground is good at sinking current (sinks 1–2 mA) and is compatible with the sinking requirement of 1 TTL load (1.6 mA at 0.4V). It will meet the "low" voltage requirement of CMOS logic. All output options use this device (device #1) for current sinking. On the other hand, the relatively high impedance depletion-mode device (device #2) to V_{CC} provides low current sourcing capability (100 μA at 2.4V). This pullup is sufficient to provide the source current for a TTL high level and will go to V_{CC} to meet the "high" voltage requirements of CMOS logic. An external resistor to V_{CC} may be required to interface to other external devices requiring higher sourcing capability.

An interface example to a common emitter NPN transistor is given below:



TL/DD/8440-2

FIGURE 2

R_B is needed to limit transistor's base current if $I_{source} > I_{B(max)}$.

R_p helps generate base drive if the I_{source} is not sufficient. The disadvantage of R_p is the introduction of more power dissipation. The temperature effects on the reverse saturation current I_{CBO} causes I_C to shift. I_{CBO} approximately doubles for every 10°C temperature rise. The effect of changes in I_{CBO} reduces off state margin and increases power dissipation in the off state.

However, in a typical device, the current supplied by R_p will swamp out any effects on I_{CBO}. Another parameter found to be decreasing linearly with temperature is V_{BE}:

$$\Delta V_{BE} = V_{BE_2} - V_{BE_1} = -k(T_2 - T_1)$$

where $k \approx 2$ mV/°C, T in °C.

Now let's consider a practical example:

LOW SOURCE CURRENT OUTPUT:

Standard output, COP420, device #2.

The selected transistor is 2N3904.

DESIGN CONSIDERATIONS:

a. Q is in saturation during ON-state.

b. Q's collector current I_C = 100 mA

TABLE I

	Default	Standard	Push-Pull	High Sink	Very High Sink	LED	Hi-Current LED	TRI-STATE® Push-Pull	Hi Current TRI-STATE Push-Pull	Open Drain
SO	Push-Pull	Logic Compatible; Non MICROWIRE™	MICROWIRE Higher Drive, Faster X'sition							External Pull Up
SK	Push-Pull	Logic Compatible; Non MICROWIRE	MICROWIRE Higher Drive Faster Transition							External Pull Up
D	Standard	Logic Compatible		L Parts Only 15 mA	L Parts Only 30 mA					External Pull Up, Standard, Hi Sink or V.H.S. Pull Down
G	Standard	Logic Compatible; Inputs		L Parts Only 15 mA	L Parts Only 30 mA					External Pull-Up, Standard, Hi Sink or V.H.S. Pull Down
L	Standard	Logic Compatible; Inputs, TRI-LEVEL				Hi Source 1.5 mA TRI-LEVEL	L Parts Only Higher Source 3 mA TRI-LEVEL	MICROBUS™ Meets TRI-STATE Spec. TRI-LEVEL	L Parts Only Meets TRI-STATE Spec. TRI-LEVEL	External Pull Up TRI-LEVEL
H	Standard	Logic Compatible Inputs								External Pull Up
R	Standard	Logic Compatible; Inputs, TRI-LEVEL	Higher Drive Faster Transition TRI-LEVEL					Meets TRI-STATE Spec TRI-LEVEL		External Pull Up TRI-LEVEL

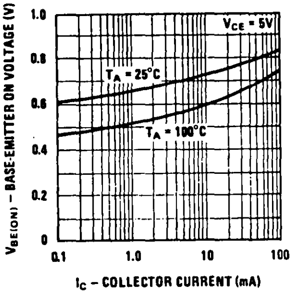
c. Assuming a "forced" of 10 for Q. This is a standard value for β to insure saturation.

For an $I_C = 100$ mA, $\beta = 10$, we have $I_B \geq 10$ mA. The low current standard output certainly cannot provide $I_B \geq 10$ mA. Therefore, a pullup resistor (R_p) is required.

d. Now we need to select the minimum allowed value for R_p. The sinking ability of COPS output will determine R_p. We must sink the pullup current to a $V_{OUT} < V_{BE}$ in order to hold Q off. Also, note that

$$\frac{\Delta V_{BE}}{\Delta T} = -2\ \text{mV/°C}.$$

e. Assuming the worst case is at V_{CC} (max) and High-temperature (let $\Delta T = 20°C \Rightarrow \Delta V_{BE} = -40$ mV). From $V_{BE(ON)}$ Vs. I_C curve, *Figure 3*:



TL/DD/8440-3

FIGURE 3. 2N3904 I/V

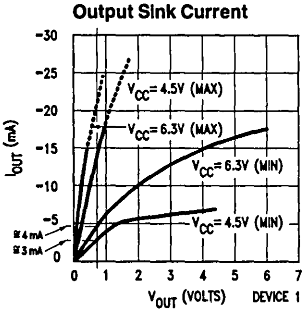
at 100 mA, 25°C, $V_{BE} \cong 0.85$V.

So, our $V_{BE(45°C)} = 0.85 - 0.04 \cong 0.81$V.

There is not margin here for process V_{BE} variations so we can allow 200 mV of slope,

$V_{BE} = 0.61$V (worst case)

f. Having $V_{BE} = 0.61$V, we go to COPS sink graph and draw a vertical line at $V_{OUT} = V_{BE} = 0.61$V. *Figure 4* below:



TL/DD/8440-4

FIGURE 4

This will tell us, at $V_{out} = V_{BE}$, how much current can be sinked to keep Q "OFF". The intersection of $V_{CC} = 6.3$ (MIN) and $V_{BE} = 0.61$V gives us $I_{sink} = 4$ mA.

g. Now calculate R_p.

$$R_p \geq \frac{6.3 - 0.61}{4} k \geq 1.42k$$

$$\text{the actual standard } R_p\ (\pm\ 10\%) = \frac{1.42}{0.9}$$

$$= 1.6k \pm 10\%$$

h. Using the value of R_p, let's calculate the current through R_p at $V_{CC} = 4.5$V(MIN).

$$I_{Rp} = \frac{4.5 - 0.61}{1.42} \text{mA} = 2.74 \text{ mA}$$

Which is less than sink ability of device (3 mA from *Figure 4*) at $V_{CC} = 4.5$V, $V_{out} = 0.61$V.

i. Now calculate the available source current. Here we use $V_{BE(max)}$ which is the worst case, and low temperature.

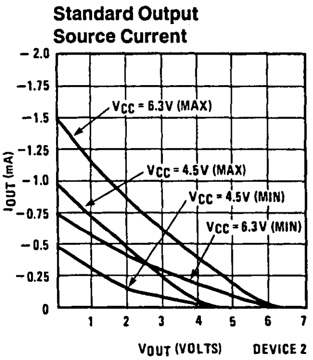
Let T (ambient) = 10°C.

From V_{BE} vs. I_C curve, *Figure 3*:

$V_{BE} \cong 0.83$V at 25°C

$V_{BE} \cong 0.83 + 2$ mv/°C × 15 = 0.86V at 10°C.

Using this value of V_{BE}, we go to COP420 Standard Output source current curve (*Figure 5*), and draw a vertical line at $V_{BE} = 0.86$V. The intersection of this line and $V_{CC} = 4.5$(MIN) gives an $I_{source} = 325\ \mu$A.



TL/DD/8440-5

FIGURE 5

This is low but typical of N-channel low current standard output.

Contribution of R_p

$$I_{Rp} = \frac{4.5 - 0.86}{\underbrace{(1.6)(1.1)}_{R_{p(max)}}} = 2.07 \text{ mA}$$

$I_{B(min)} \cong 2.07 + 0.325 = 2.3$ mA

This is our worst case base drive, but we needed 10 mA.

What can we do to get the base drive we need?

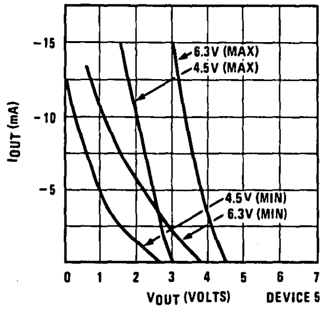
1. We can use above design and allow Q to come out of saturation. The disadvantage is that Q's power dissipation increases.

2. Or use a Darlington configuration (Process 05). In such a configuration only first stage of Darlington can be saturated (not output stage). This will introduce a slightly higher power dissipation. Note that for a process 05 transistor, the forced β is 1000.

3. Use a high source type output such as TRI-STATE output. If we draw a vertical line at V_{BE} = 0.86, we get a source current of ≅ 6 mA at V_{CC} = 4.5(MIN) *Figure 6*, which gives us a worst case

$$I_{B(min)} = 8.07 \text{ mA}.$$

TRI-STATE Output Source Current



TL/DD/8440-6

FIGURE 6

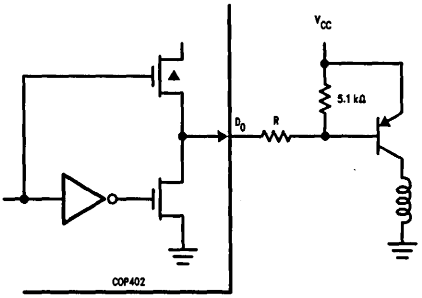
CAUTION On TRI-STATE graph the intersection of $V_{out} = B_{BE}$ = 0.86V and V_{CC} = 6.3V(MAX) curve (*Figure 6*) would result in an $I_{B(Max)}$ = 50–60 mA, which is way too much to handle. In this case there is a need for a series current limiting R_B to kill some of the worst case $I_{B(max)}$.

4. There is a high current Standard-L option on some COPS (i.e., COP4XL, L-port) which provides sufficient source current.

5. N-channel output can generally sink better than source. PNP transistor can be used instead of NPN. The same analysis applies and in general will show better overdirve capabilities.

As shown in *Figure 7*, the D_0 output which has a standard output option, is driving the base of the PNP transistor. Assuming V_{CC} = 4.5V (for COP402), V_{BE} = 1.0V, and a worst case base drive requirement of 3.0 mA. We see that we must supply 200 μA to the base-emitter resistor to turn the transistor on:

$$1.0\text{V}/5.1\text{k} = 200\ \mu\text{A}$$



TL/DD/8440-7

FIGURE 7. PNP Drive

From the output sink current curve on the COP402 data sheet, we find that, at 1.0V the D-line can sink 3.2 mA. To calculate the value of the current limiting resistor,

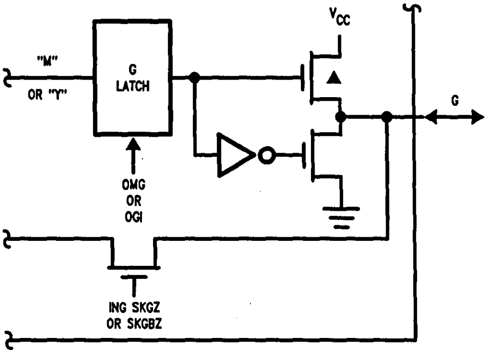
$$R = (V_{CC} - V_{BE} - V_{D0})/I$$

When V_{CC} = 6.3V, the D0 output can sink more than enough current at 0.3V, and if the V_{BE} = 0.7V, we can calculate the maximum D_0 output current:

$$I = (V_{CC} - V_{BE} - V_D)/R$$
$$= (6.3 - 0.7 - 0.3)/780 = 6.3 \text{ mA}.$$

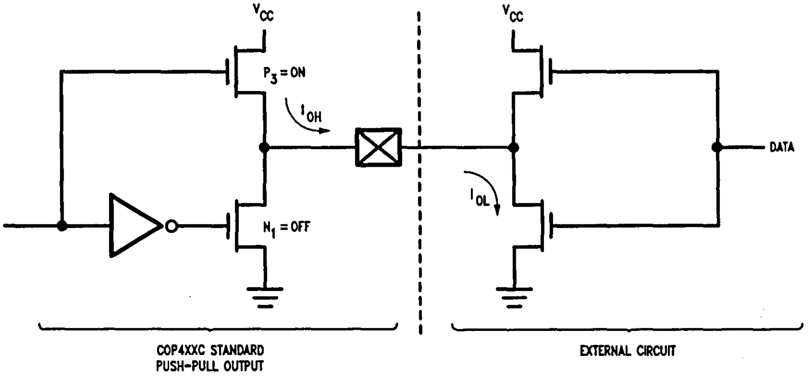
Using the Standard Output Option for Bidirectional I/O (G-port)

The standard output is good at sinking current, but rather weak at sourcing it. Therefore, by using the Standard Drive configuration and outputting 1's to the port, an external source may easily overdrive the port drivers with the added bonus of a built-in pullup. While the depletion-mode device provides sufficient current for a TTL high level, yet can be pulled low by an external source, thus allowing the same pin to be used as an input and output. Data written to the ports is statically latched and remains unchanged until rewritten. As inputs the lines are non-latching (*Figure 8*).



TL/DD/8440-8

FIGURE 8. G Port Characteristics



TL/DD/8440-9

FIGURE 9

When writing a "0" to the port, the enhancement-mode device to ground overcomes the high pullup and provides TTL current sinking capability. While writing a "1" the depletion-mode device behaves as internal pullup maintaining the "1" level indefinitely. In this situation, an input device capable of overriding the small amount of current supplied by the pullup device can be read. This feature provides maximum user flexibility in selecting input/output lines with minimum external components.

In CMOS-COPS the low current push-pull output has even much weaker source current capability and this make it easier to be overriden.

Referring to *Figure 9*.

Note that $I_{OL} > I_{OH}$, otherwise transistors or buffers must be used.

For COP424C/444C, standard push-pull

@ $V_{CC} = 4.5V, V_{out} = 0V, I_{OH(min)} = 30\ \mu A$
$I_{OH(max)} = 330\ \mu A$

@ $V_{CC} = 2.4V, V_{out} = 0V, I_{OH(min)} = 6\ \mu A$
$I_{OH(max)} = 80\ \mu A$

While in NMOS (COP420L), Standard output:

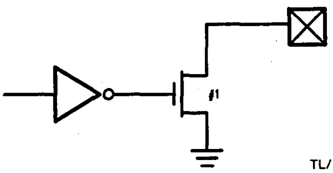
@ $V_{CC} = 4.5V, V_{OH} = 2.0V, I_{OH(min)} = 30\ \mu A$
$I_{OH(max)} = 250\ \mu A$

@ $V_{CC} = 6.3V, V_{OH} = 2.0V, I_{OH(min)} = 75\ \mu A$
$I_{OH(max)} = 480\ \mu A$

As we see, both in CMOS and NMOS it is easier to override I_{OH}. Note that the standard output option is available with standard, high, or very high sink current capability ("L" parts only). The pulldown device is bigger for the high/very high current standard output. The sourcing current is the same. These three choices provide some control over current capability.

B. OPEN-DRAIN OUTPUT

This option uses the same enhancement-mode device to ground as the standard output with the same current sinking capability. It does not contain a load device to V_{CC}, allowing external pullup as required by the user's application. The sinking ability of device #1 determines the minimum allowed external pullup. The analysis discussed earlier for Standard Output options equally applies here. Available on SO, SK, and all D, G, and L outputs.



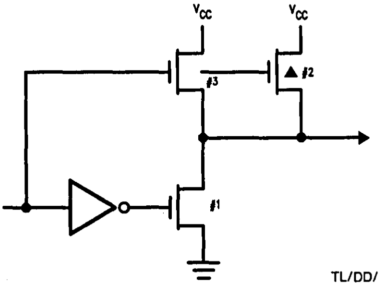
TL/DD/8440-10

FIGURE 10. Open-Drain Output

The open-drain option makes the ports G and L very easy to drive when they are used as inputs. This option is commonly used for high noise margin inputs, unusual logic level inputs as from a diode isolated keyboard, analog channel expansion, and direct capacitive touch-panel interface. Available with standard, high or very high sink capability ("L" parts only).

C. PUSH-PULL OUTPUT

The push-pull output differs from the standard output configuration in having an enhancement-mode device in parallel with the depletion-load device to V_{CC}, providing greater current sourcing capability (better drive) and faster rise and fall times when driving capacitive loads.



TL/DD/8440-11

FIGURE 11. Push-Pull Output

If a push-pull output is interfaced to an external transistor, a current-limiting resistor must be placed in series with the base of the transistor to avoid excessive source current flow out of the push-pull output. This option is generally for MICROWIRE Serial Data exchange.

It is available on SO, SK only and is recommended to be used as a default option for these outputs. A few points must be kept in mind when using SO, SK for MICROWIRE interface.

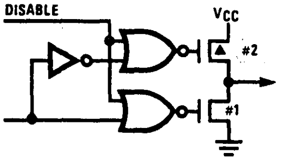
The data sheet specifies the propagation delay for a certain test condition (i.e., $V_{CC} = 5V$, $V_{OH} = 0.4V$, Loading = 50 pF, etc.).

In practice, actual delay varies according to actual input capacitive loading (typical 7–10 pF per IC input), total wire capacitance and PCB stray capacitance connected to the SI input. Thus, if actual total capacitive loading is too large to satisfy the delay time relationships ($t_d = t_{SK} - t_r$; t_d = actual delay time, t_{SK} = the instruction cycle time, t_r = the finite SK rise time), either slow down SK cycle time or add a pullup resistor to speed up SK "0" to "1" transition or use an external buffer to drive the large load. Besides the timing requirement, system supply and fan-out/fan-in requirements have to be considered, too.

If devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Briefly, for devices that have incompatible input levels or source/sink requirements to exchange data, external pullups or buffers are necessary to provide level shifting or driving. Unreliable operation might occur during data transfer, otherwise. For a 100 pF load, a standard COPS Microcontroller may use a 4.7k external resistor, with the output "low" level increased by less than 0.2V. For the same load the low power COPS may use a 22k resistor; with the SO, SK output "low" level increased by less than 0.1V.

D. STANDARD L OUTPUT

Same as Standard Output, but may be disabled. Available on L-outputs only.



TL/DD/8440-12

FIGURE 12. Standard L Output

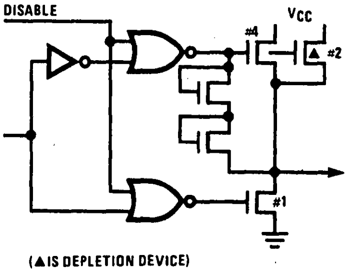
When this option is implemented on the L-port and the L-drivers are disabled to use the L lines as inputs, the disabled depletion-mode device cannot be relied on to source sufficient current to pull an input to a logic high. There are two ways to use L lines as inputs (having standard L option):

The first method requires that the drivers be disabled. In this case the lines are floating in an undefined state. The external circuitry must provide good logic levels both high and low to the input pins. The inputs are then read by the INL instruction. The second method is similar to the technique used for the G-port. The drivers are enabled and a"1" must be written to the Q register.

The external circuitry will then be required only to pull the lines low to a logic "0". The line will pull up to a "1" itself. The INL instruction is used as before to read the lines.

E. LED DIRECT DRIVE OUTPUT

In this configuration, the depletion-load device to V_{CC} is paralleled by an enhancement-mode device to V_{CC} to allow for the greater current sourcing capacity required by the segments of an LED display. Source current is clamped to prevent excessive source current flow.



(▲IS DEPLETION DEVICE)

TL/DD/8440-13

FIGURE 13. LED (L output) NMOS-COPS

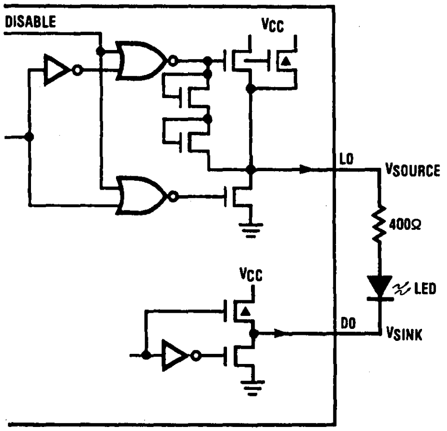
This configuration can be disabled under program control by resetting bit 2 (EN^2) of the enable register to provide simplified display segment blanking.

However, while both enhancement-mode devices are turned off in the disabled mode, the depletion-load device to V_{CC} will still source up to 0.125 mA. As in the case of Standard L output, again this current is not sufficient to pull an input to a logic "1".

The drivers must be disabled and the lines must be pulled high and low externally, whenever they are used as inputs.

Example #1:

When COPS outputs are used to drive loads directly, the power consumed in the outputs must be considered in the maximum power dissipation of the package. *Figure 14* shows an LED segment obtaining its source current from L_0 output and D_0 sinking the current. In this configuration all the power required to drive the LED with the exception of the portion consumed by the LED itself, is consumed within the chip. Assuming COP404L is the driving device:



TL/DD/8440-14

FIGURE 14. LED Drive

If we assume the V_{source} is not inserted, the device has a V_{CC} of 9.5V, and that the voltage drop across the LED is 2.0V.

We can calculate the power dissipation in these outputs. The minimum current that D_0 can sink at 1.0V is 35 mA (COP404L data sheet). L_0 can source up to 35 mA at 3.0V. Therefore, the power dissipation for the L_0 output could be: (9.5 −3.0) (0.035) = 227 mW. The power in the D_0 output is (1)(0.035) = 35 mW.

Now let us calculate the current limiting resistor. Referring to COP404L L_0–L_7 output source current curves, at V_{CC} =9.5V the minimum current curve peaks at I = 6.0 mA and V_{source} = 4.8V. The current curve is actually very flat between 4.0 and 5.0 volts. For maximum current, we need to set the voltage on the L pin equal to 4.8V at 6.0 mA. The D line will sink this current at 0.4V. Therefore, the resistor and LED must make up the difference:

$$V_I = V_D + IR + V_{LED}$$
$$4.8 = 0.4 + 0.006R + 2.0$$
$$R = 400\Omega$$

At the other end of the curve, when the L line sources the maximum current, assume the LED and the D line will have the same voltage drop.

$$V_I = 0.4 + IR + 2.0$$
$$V_I = 2.4 + IR$$

From the current curve, we see that at 6.4V the L line will source 10 mA. Therefore: V_I = 2.4 + (0.01) (400) = 6.4V.

Example #2:

Let's consider a different configuration.

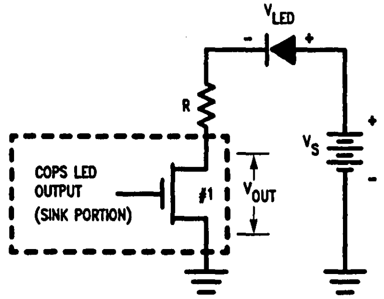


FIGURE 15. LED DRIVE

Now we calculate the series current limiting resistor R. The circuit has two non-linear devices to be considered; the output device and the LED.

The LED in this example is NSC5050. Looking at I/V curve, the device has a threshold 1.6V. Also, note that for V_{LED} > 1.6V the I/V curve is very linear (*Figure 17*). Because of this, the LED characteristic can be modeled as a sharp threshold device with a non-zero source resistance (normally I/V curve is LOG looking).

From ON part of curve,

$$R_S = \frac{1.9 - 1.7}{0.05} = 4\Omega$$

We can neglect R_S as well (only R_S < R). Our model is simply a voltage source for the LED when

$$I = 0 \text{ for } V_{LED} < V_{TH}$$
$$I = \infty \text{ for } V_{LED} > V_{TH}$$

Design Procedure:

1. $I_{LED(min)} = \frac{V_{S(min)} - (V_{LED(max)} + V_{OUT(max)})}{R(max)}$

We need endpoints of the load line.

a. $@V_{out} = 0 \Rightarrow I_{LED(min)} = \frac{V_{S(min)} - V_{LED(max)}}{R(max)}$

b. $@V_{out} + V_{LED(max)} = V_S \Rightarrow I = 0$

$(V_{LED(max)} = 2V)$

2. Plot a and b

Assuming an I_{min} = 7 mA, $V_{S(min)}$ = 4.5V

from 1 $R_{(max)}$ = 357Ω

Draw the load line with slope −1/357 crossing

$V_{out} = V_S - V_{LED(max)} = 4.5 - 2 = 2.5V$.

(*Figure 16*).

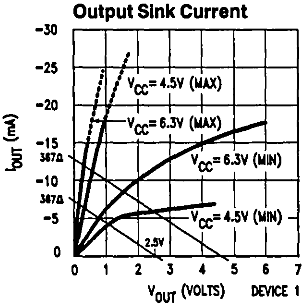
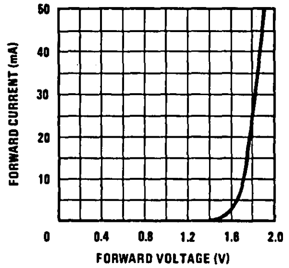


FIGURE 16. COP420

Forward Current (I_F) vs Forward Voltage (V_F)



TL/DD/8440-17

FIGURE 17. LED I/V Characteristic

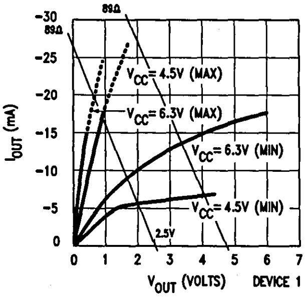
The intersection of this load line and $V_{CC} = 4.5V$ (min) curve, we find an actual value of $I_{(min)} = 4.25$ mA.

To determine I_{max} (at $R = 357\Omega$) we draw a parallel load line intersecting $V_{out} = 6.3 - 2.0 = 4.3V$ and find that @ $V_{CC} = 6.3V$, $I_{(max)} = 13$ mA.

3. From above calculations we observe that our $I_{(min)}$ (actual) is way off. Let's try to rotate our first load line around $V_{out} = 2.5V$ to increase I_{min} and then check I_{max} and R. (*Figure 18*).

Let's go for an I_{min} (actual) = 6 mA. This will give us $R = 89\Omega$ and the max. plot goes off the graph to = 36 mA.

Output Sink Current



TL/DD/8440-18

FIGURE 18. COP420

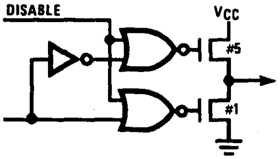
Comments:

1. The design must be a compromise between the two extremes (battery life should also be considered).

2. The lower the LED threshold the better. (The load line moves further up the device curve.)

F. TRI-STATE PUSH-PULL OUTPUT

This option is specifically available to meet the specifications of National's MICROBUS, outputting data over the data bus to a host CPU. It has two enhancement-mode devices to ground and V_{CC}.



TL/DD/8440-19

FIGURE 19. TRI-STATE Push Pull (L output)

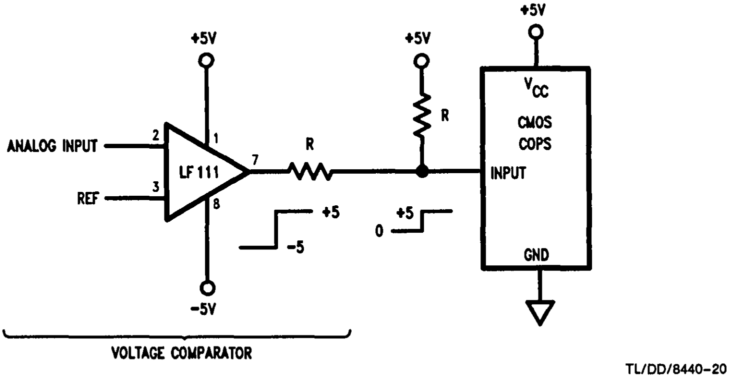
The TRI-STATE logic can disable both enhancement-mode devices to free the MICROBUS data lines for input operation.

CAUTION Never try to pull against the TRI-STATE Output (too much source current) with the drivers enabled and Q register previously loaded with "1". The choices we have are mentioned earlier. Either TRI-STATE L-port or use Standard L output option.

II. INPUTS

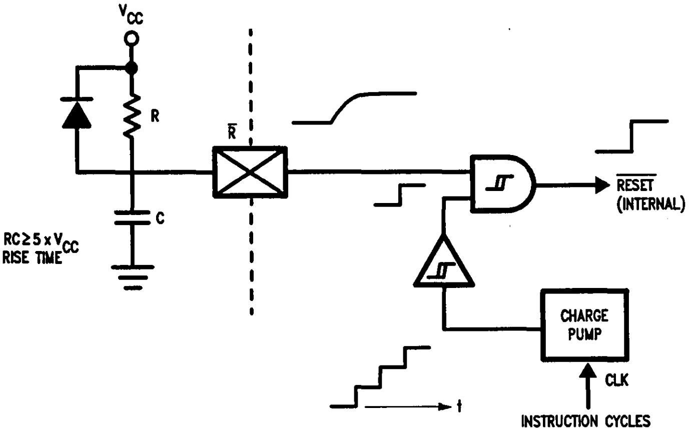
COPS inputs may be programmed either with a depletion load device to V_{CC} or floating (Hi-Z input). All inputs are TTL/CMOS compatible. Hi-Z inputs should not be left floating; they should be connected to the output of a "high" and "low" driving device if active or to V_{CC} and ground if unused. Especially when using CMOS COPS (very high impedance inputs), the open inputs can float to any voltage. This will cause incorrect logic function and more power dissipation. Also, the CMOS inputs are more susceptible to static charge which causes gate oxide rupture and destroys the device. Unlike inputs, the outputs should be left open to allow the output switch without drawing any DC current. Another precaution is powering up the device. Never apply power to inputs or TRI-STATE outputs before both V_{CC} and ground are connected. This will forward bias input protection diodes, causing excessive diode currents. It will also power the device.

Special care must be practiced when interfacing a CMOS-COPS input to an analog IC, powered by different supply voltages. Avoid overvoltage conditions resulting



TL/DD/8440-20

FIGURE 20



TL/DD/8440-21

FIGURE 21

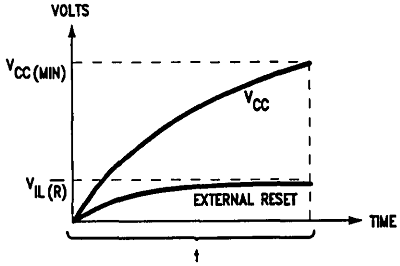
from such situations. As an example, consider the interface of a CMOS-COPS with the LF111 voltage comparator:

When the low level "−5V" appears on the comparator's output, the COPS input is pulled low below "logic low" of "0V". This will cause damage if the comparator sinks enough current. The use of a current-limiting resistor in series with the input is helpful. A better solution is to use a voltage divider as shown in *Figure 20.* Any time a low level appears on the comparator's output, a total voltage drop of 10V will appear across both resistors each dropping 5V, causing the input to sit at 0V. Whenever the output goes high, the resistors will not drop any voltage (no current through the resistors) and a logic high of 5V will appear on the input. To reduce power dissipation introduced by resistors, the resistor value must be high (>100k), because the CMOS inputs have very high input impedance.

$\overline{\text{RESET}}$ INPUT

All COPS Microcontroller have internal reset circuitry. Internally there is an AND gate with one input coming from the $\overline{\text{RESET}}$ input, and the second input connected to a charge pump circuitry. In the Charge pump circuit, a tiny capacitor is being charged upon execution of each internal instruction cycle. When the voltage across this internal capacitor reaches a high logic level, the second input of the AND gate is released.

The Reset logic will initialize (clear) the device upon power-up if the power supply rise time is less than 1 ms and greater than 1 µs. With a slowly rising power supply, the part may start running before V_{CC} is within the guaranteed range. In this case, the user must provide an external RC network and diode shown in *Figure 21* above. The external RC network is there to hold the $\overline{\text{RESET}}$ pin below V_{IL} until V_{CC} reaches at least $V_{CC(min)}$. The desired response is shown in *Figure 22.*



TL/DD/8440-22

FIGURE 22

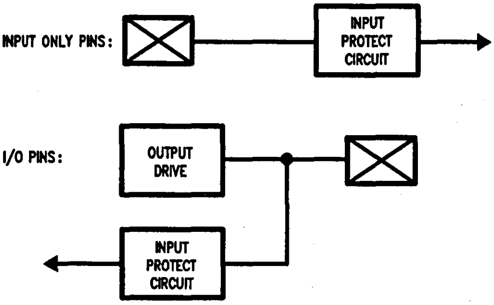
t = 500-600 instruction cycles (8 msec)
for COPxxxL
t = 900–1000 instruction cycles (4 msec)
for COPxxxC

The diode is included in the reset circuitry to cause a "forced Reset" when the power supply goes away and recovers quickly. In such a situation the diode helps discharge the capacitor quickly. Otherwise, if the power failure occurs for a short time, the capacitor will not be fully discharged and the chip will continue operation with incorrect data.

Note that on the CMOS COPS, the internal charge pump circuitry can be disabled when using a very slow clock (<32 kHz) [option 23 = 1]. This is necessary, because one can run from DC to 4 μs instruction cycle time (fully static). In such a situation external RC network discussed earlier must be used.

INPUT PROTECTION DEVICES

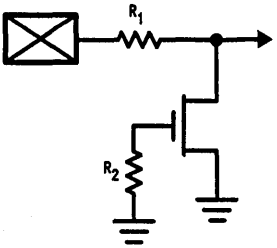
All inputs and I/O pins have input protection circuitry. This circuitry is there regardless of any option selected. It is the first circuitry encountered at the pin.



TL/DD/8440-23

FIGURE 23

For NMOS and XMOS devices, the circuits are of the form:

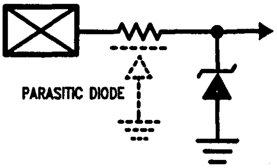


TL/DD/8440-24

FIGURE 24

This is a standard circuit defined for the process. R_1 is on the order of 200Ω. R_2 is around 300Ω (note that the R values are not precise).

This circuit is functionally equivalent to:



TL/DD/8440-25

FIGURE 25

The zener breakdown is around 10–15V; the gate breakdown is 50V.

CONCLUSION

All COPS Microcontrollers have a number of I/O options necessary to implement dedicated control functions in a wide variety of applications. The flexibility to select different options allows the user to tailor within limits, the I/O characteristics of the Microcontroller to the system. Thus, the user can optimize COPS for the system, thereby achieving maximum capability and minimum cost. This application note deals with the basic functionality of COPS I/O characteristics and does not address electrical differences among the various COPS devices.

New CMOS Vacuum Fluorescent Drivers Enable Three Chip System to Provide Intelligent Control of Dot Matrix VF Display

National Semiconductor
Application Note 440
Tom Markman

INTRODUCTION

Vacuum Fluorescent (VF) displays are becoming more and more common in a variety of applications. Manufacturers of everything from Automobiles to Video Recorders have taken advantage of these easy to read displays. VF displays are available in a wide variety of configurations; clock displays, calculator displays, multi-segment, and dot matrix displays are readily available at a low cost. This application note develops and covers in some detail a small CMOS system consisting of a single chip microcontroller and two display drivers which control a 20 character, 5 x 7 dot matrix VF display.

Figure 1 shows the schematic of the system. The microcontroller, a COPS™ 424C, receives a character in ASCII form from the host system, stores the ASCII value of the character in its onboard RAM, converts the ASCII value to a 5 byte data word suitable for the display drivers and displays it on the VF display. The COPS also refreshes the display continuously while performing character update, much like a dumb terminal. Not including the address decoding logic, this application requires only the onboard RAM and ROM of the COPS424C, and National's MM58341 and MM58348 VF display drivers. If a steady message or a scrolling sentence is desired, only small changes in the COPS software are required. In this case the messages could be stored in the ROM of the COPS and the need for a host system would be eliminated.

VF DISPLAY AND VF DISPLAY DRIVER REQUIREMENTS

The display used in this application was an Itron #DC205G2. This 20 segment, 5 x 7 dot matrix, multiplexed display required a filament voltage of 5.7 Vac and a filament current of 37 mAac. The anode and grid voltages were supplied by the display drivers. The voltage and current requirements vary considerably for different displays depending on the size and number of characters, and the configuration (dot matrix, 7 segment, 14 segment, etc.). To determine the voltage requirements for a particular display, a simple calculation can be made. If maximum possible brightness of the display is desired, the following equation must be true:

$E_t \geq E_b + E_k + (I_b)(R_{on})$ where:

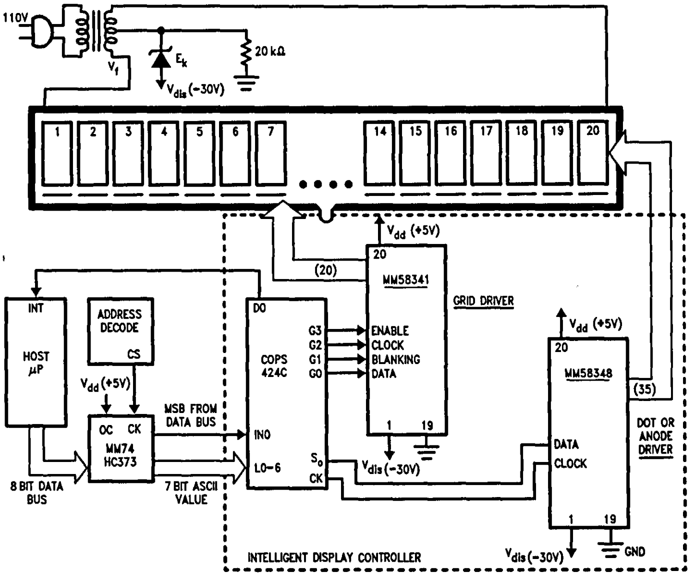
E_t is the total Voltage of the display drvier or $|V_{dis}| + V_{dd}$

E_k is the display Cathode Bias Voltage

$E_b = E_c$ is the typical Anode or Grid Voltage ($V_{p\text{-}p}$)

I_b is the typical anode current (mAp-p)

R_{on} is the display driver output impedance (Ω)



TL/F/8683–1

FIGURE 1. System Diagram Showing the Basic 3-Chip Display Controller and the Interface to a Microprocessor System

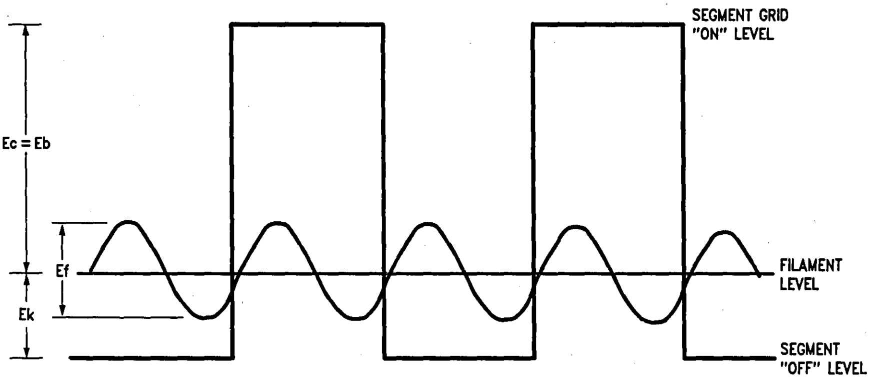
If the maximum brightness is not desired, the following equation can be used: $(E_t)(1.2) \geq E_b + E_k + (I_b)(R_{on})$. In this application, the calculated E_t was 42.25V, however, the display was legible under normal lighting conditions, with an E_t as low as 25V. If your display requires more than the 35V output of the MM58341 and MM58348, pin for pin compatible 60V VF Display Drivers (MM58241, MM58248) are available.

Figure 2 shows the relationship between the required VF display voltages. The cut-off voltage (E_k) is set by the Zener diode on the center tap of the filament transformer. This value is given in the VF display data sheet.

Avoiding Flicker and Pulsing

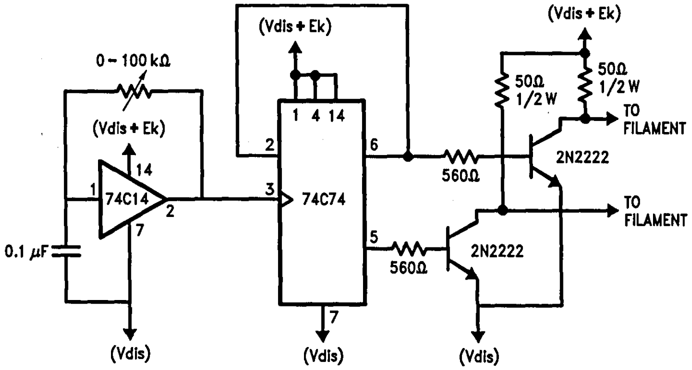
There are two different conditions which may cause the display to appear to flicker. The first is the refresh rate. This is particularly a problem on displays where the micro-controller must up-date more than 25 characters. Since the human eye begins to notice flicker at about 40 Hz, a display with a refresh rate less than that will appear to be flashing on and off.

The second type of flicker occurs when the refresh rate is between 40 Hz and 90 Hz. In this case, the display will appear to be rolling rather than flashing. This condition occurs when the refresh rate and the filament frequency are close together. If a character is only on during the time when the filament voltage is negative, it will appear to be slightly brighter than the character next to it which may only be on during the positive cycle of the filament voltage. If this is the case, as it was in this application, the simplest solution is to increase the frequency of the filament. A DC oscillator circuit, such as the one shown in *Figure 3*, can be used to replace the AC voltage source. The filament frequency can be easily adjusted to eliminate this condition.



TL/F/8683-2

FIGURE 2. Voltage Levels for VF Display



TL/F/8683-3

FIGURE 3. Filament Oscillator Circuit

VF Display Drivers

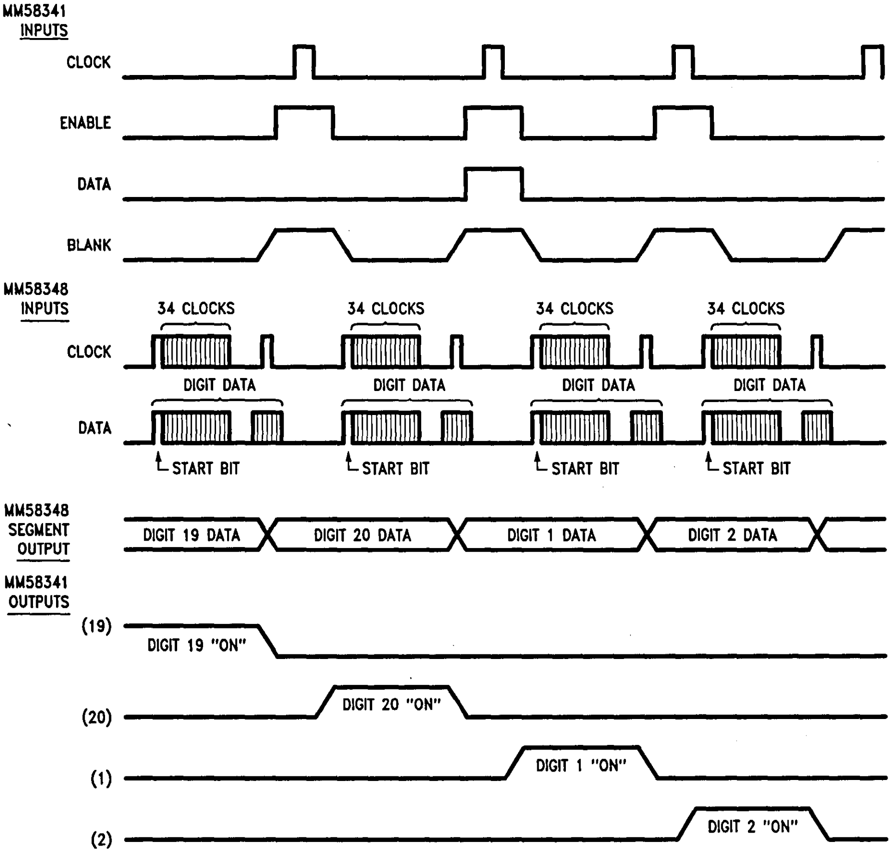
Two high voltage display drivers were needed to control the VF display. A MM58341, was used to control the grids and a MM58348 was used to control the individual pixels or anodes. Both of these drivers receive serial information and output 32 and 35 segments of data respectively.

The MM58341 has three control pins which make it ideal for controlling the grids of a VF display. The blanking control pin will turn off all segments of the display when a logic '1' is applied to this pin. This is particularly important for reducing ghosting, and controlling brightness. Ghosting is a condition where the last characters shadow appears behind the character being displayed. The enable pin acts as an envelope for the input signal. Only while it is at a logic '1' level will the circuit accept clock inputs. When the pin goes low, all the data is latched and displayed. A data out pin is also provided for cascading. If the display has more than 32 grids, a second grid driver can be cascaded by connecting the data out pin to the input data for the second grid driver.

The MM58348 is a 35 bit shift register and latch which is used to control each pixel or dot. When a leading 1, followed by 35 bits of data, is received, the data is latched and displayed. The chip is automatically reset upon power up.

MULTIPLEXED DISPLAY REFRESH TIMING

Considering first the digit driver (MM58341), it becomes clear that the digits must be enabled or refreshed sequentially and that this process must be continuous regardless if the display data has changed. The data for the MM58341 is simply a 1 followed by 19 zeroes where the 1 is shifted through the internal registers of the MM58341. As each digit is enabled, the corresponding segment data is displayed. To insure that no ghosting effects are seen during the transition between digits, the blanking control is activiated just before the data is latched into the dot or anode driver and deactivated just after the data has been latched. During this time when the blanking control is activated, the grid driver is clocked shifting the 1 to the next location. *Figure 4* shows the micro-controller waveforms and the resultant display waveforms for the 20 character display.



TL/F/8683-4

FIGURE 4. Timing Diagram

In between digit strobes, the segment data is updated. The first 34 bits of segment data are set up in the dot driver and the blanking signal is activated to disable all 20 digits. The 35th bit of data is clocked in, updating the segments. Since the MM58348 resets its internal shift register each time the data is latched, it can accept all but the final data bit while still displaying the previous digit. The digit driver is then clocked, shifting the digit strobe to the next position. The enable is then brought low, enabling the next digit. Finally blanking control is deactivated and the data displayed.

During the time which the blanking control is high, the order in which the segments or the digits are updated is not critical. Since this occurs while the display is blank. The digit driver may be clocked first, or the segments could be changed first. In general, the philosophy for the driving this VF multiplexed display is outlined in *Figure 5.*

HOST INTERFACE AND PROGRAMMING

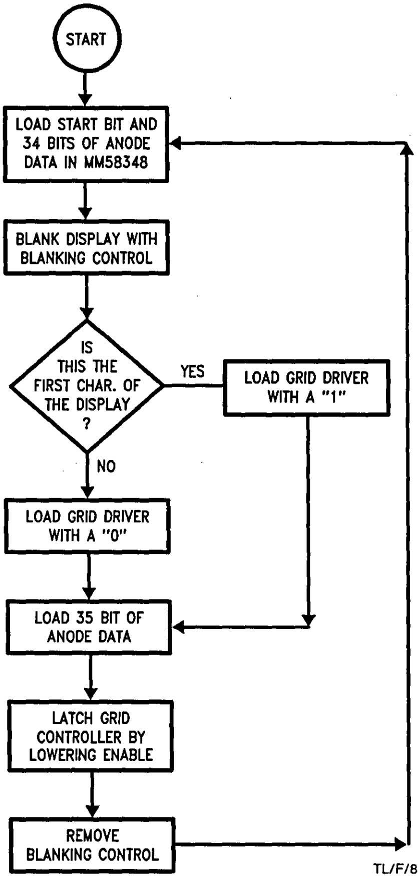
With a minimal amount of address decoding and an eight bit latch, COPS can be interfaced with a common microprocessor bus. When a character has been input into the host to be displayed, the ASCII value of that character is latched into the eight bit latch (MM74HC373) and is read on the L port (L0–6) of the COPS. The MSB of the ASCII value must be a logic 1. This MSB is the signal to the COPS that a new character is being presented. Once the character has been stored, an interrupt is sent from the COPS to the host through the D-0 port. The COPS checks for a new character being input every 200 μs. If a character is being sent, 1 ms is required to store that character in the RAM of the COPS. With the COPS controlling the display, the host micro-processor is not being tied down with character look-up and display refresh. A simple flowchart of the host requirements is shown in *Figure 6.*

COPS SOFTWARE

There are four main sections of the COPS software. The first section, the initialization of the RAM, sets up the RAM as shown in *Figure 7.* A '0' is stored in all of the LSB positions and a '2' is stored in all of the MSB positions. Since the COPS is in a constant display loop, this is necessary to insure a blank display. 20H is the ASCII value of a space. With the RAM set up in this way, a maximum of 28 characters can be stored in RAM. Since the display in this application is only 20 characters long, RAM locations M1,4 to M1,11 and M3,4 to M3,11 are not used. RAM locations 1,12 to 1,15 and 3,12 to 3,15 are used as temporary storage throughout the program and cannot be used for character storage.

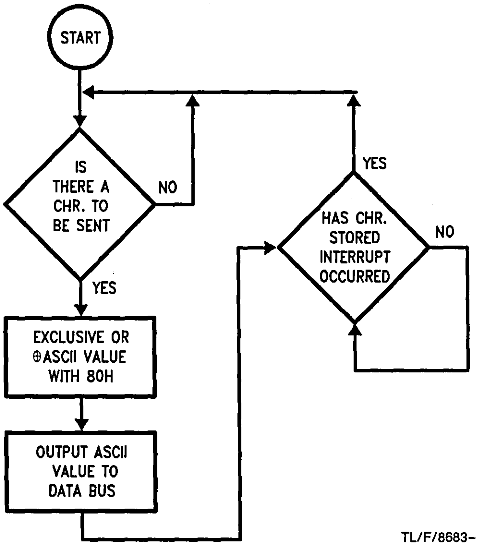
The second part of the program, stores the new characters sent by the host CPU in RAM. Once a character has been sent, this section of the program checks the ASCII value of that character to see if it is a control character or a display character. If it is a display character, the character is stored in RAM and an interrupt is sent to the host. There are three control characters which the COPS program will recognize. Cursor forward (ASCII value 08H) moves the cursor forward without destroying the data, cursor backwards (ASCII value 0CH) moves the cursor backwards without destroying the data, and return (ASCII value 0DH) will clear the display and put the cursor at the beginning of the display. To recognize and store a character, 1 ms is required.

The third part of the program, the display loop, is the heart of the program. Unless a new character has been detected, the program is always in this loop. This section does the


START
LOAD START BIT AND 34 BITS OF ANODE DATA IN MM58348
BLANK DISPLAY WITH BLANKING CONTROL
IS THIS THE FIRST CHAR. OF THE DISPLAY ?
YES
LOAD GRID DRIVER WITH A "1"
NO
LOAD GRID DRIVER WITH A "0"
LOAD 35 BIT OF ANODE DATA
LATCH GRID CONTROLLER BY LOWERING ENABLE
REMOVE BLANKING CONTROL

TL/F/8683–5

FIGURE 5. Flowchart for Display Drivers


START
IS THERE A CHR. TO BE SENT
NO
YES
EXCLUSIVE OR ⊕ASCII VALUE WITH 80H
OUTPUT ASCII VALUE TO DATA BUS
HAS CHR. STORED INTERRUPT OCCURRED
YES
NO

TL/F/8683–6

FIGURE 6. Host System Flowchart

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
LSB Chr 1	LSB Chr 2	LSB Chr 3	LSB Chr 4	LSB Chr 5	LSB Chr 6	LSB Chr 7	LSB Chr 8	LSB Chr 9	LSB Chr 10	LSB Chr 11	LSB Chr 12	LSB Chr 13	LSB Chr 14	LSB Chr 15	LSB Chr 16	M0
MSB Pointer	LSB Pointer	Temp. ASCII STORAGE										LSB Chr 17	LSB Chr 18	LSB Chr 19	LSB Chr 20	M1
MSB Chr 1	MSB Chr 2	MSB Chr 3	MSB Chr 4	MSB Chr 5	MSB Chr 6	MSB Chr 7	MSB Chr 8	MSB Chr 9	MSB Chr 10	MSB Chr 11	MSB Chr 12	MSB Chr 13	MSB Chr 14	MSB Chr 15	MSB Chr 16	M2
Temp. Storage of Pointer												MSB Chr 17	MSB Chr 18	MSB Chr 19	MSB Chr 20	M3

FIGURE 7. COPS RAM Map

Matrix	PAD	Column 1	Column 2	Column 3	Column 4	Column 5	PAD
Binary	0 0 0 1 0 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0						
Hex.	13	EA	24	28	3E		

FIGURE 8

character font look-up, shifts the character data out the COPS serial port to the MM58348, and controls the MM58341 through the four bit parallel port (G0–4). Because the most significant nibble of the program counter is used as part of some COPS instructions, it is important that parts of the program are located at specific locations in ROM.

The final part of the program is the data. Each character is represented by a 5 byte data word. Each byte of the data word is stored at a different location in ROM. Fonts for characters with the ASCII values from 20H–5AH have already been stored in ROM. These characters can be changed or more characters can be added. The only limitation to the number of characters is the amount of available ROM.

CREATING THE 5 BYTE DATA WORD

Any number or combination of pixels or dots can be turned on at a time. To create a new character, it is easiest to first create a binary string which represents the character. A '1' in the binary string will turn on the pixel, a '0' will turn it off. To create this string, start in the upper left corner of the matrix and go down the columns.

The letter 'A' (*Figure 9*) would have a binary string shown in *Figure 8.* The data must be padded to make it an even 5 bytes in length. The pad at the beginning of the data (0001) is used as the leading 1 for the MM58348. The one bit pad at the end of the binary string must be a 0. If a 1 were sent as the pad, it would be used as the start bit for the next character.

The 5 byte data word that would be stored in ROM and represent the letter 'A' would then be 13EA24283E.

STORING THE DATA IN ROM

The 5 bytes of data are stored in 5 different locations in ROM. The first byte of data will be stored, LSB first, at location 200H plus the ASCII value of the character. For example, the ASCII value of the letter 'A' is 41H. The first byte of data for the letter 'A' would be stored, least significant bit first, at 241H. The second byte of data is stored at the location of the first data byte plus 60H or in this case at 2A1H. The location of the third byte is 40H plus the location of the second byte. In this case, the third byte of data would be stored at 2E1H. The fourth byte of data is stored at 300H plus the ASCII value of the character or at 341H for the letter 'A'. The final byte of data is stored 40H from the fourth byte or at 381H. Remember the LSB of each byte is stored first. Table I shows the locations in ROM and the values stored in them for the letter 'A'.

This application shows a VF display controller designed with a minimum number of IC's. If additional information about VF displays or VF display drivers is required, refer to Application Note AN-371 (The MM58348/342/341/248/242/241 direct drive Vacuum Fluorescent (VF) Displays.

TABLE I. Character Data of 'A' and Its Locations in ROM

Address In ROM	Data Stored
0241H	31
02A1H	AE
02E1H	42
0341H	82
0381H	E3

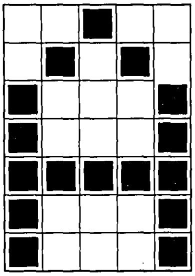


FIGURE 9. 5 x 7 Character as Stored in ROM

Section 1 of COPS Software

```
         .CHIP 424C          ;DEFINES COPS CHIP
;THIS SECTION INITIALIZES THE RAM IN THE COPS BY LOADING A
;2 IN THE MSB AND A 0 IN THE LSB LOCATIONS OF EACH CHARACTER.
;IT ALSO STOPS THE CLOCK AND SETS THE POINTER AT THE FIRST
;CHARACTER OF THE DISPLAY.

RESET:          CLRA
                LBI 3,15        ;LOADS A 2 IN ALL
                JSR CLEAR2      ;MSB LOCATIONS
                LBI 2,15        ;LOADS A 2 IN ALL
                JSR CLEAR2      ;MSB LOCATIONS
                LBI 1,15        ;LOADS A 0 IN ALL
                JSR CLEAR       ;LSB LOCATIONS
                LBI 0,15        ;LOADS A 0 IN ALL
                JSR CLEAR       ;LSB LOCATIONS

                CLRA            ;LOADS POINTER IN RAM
                XAD 1,15        ;MSB IN 1,0F
                CLRA
                AISC 15         ;LSB IN 1,0E
                XAD 1,14

                RC              ;RESETS CARRY TO
                XAS             ;    STOP CLOCK
                JMP START

CLEAR:          CLRA            ;CLEARS REGISTORS
                XDS 0
                JMP CLEAR
                RET

CLEAR2:         CLRA            ;PUTS A 2 IN REGISTORS
                AISC 02
                XDS 0
                JMP CLEAR2
                RET
```

Section 2 of COPS Software

```
;THIS SECTION OF CODE IS ONLY EXECUTED WHEN A NEW
;CHARACTER HAS BEEN ENTERED. IF THE CHARACTER IS
;A CONTROL CHARACTER, THE CURSOR IS MOVED ACCORDINGLY,
;OTHERWISE THE CHARACTER IS STORED IN THE RAM OF THE COPS.

          ;NEW CHARACTER HAS BEEN ENTERED
NEW:           LBI 1,0C        ;DUMMY POINTER
               INL             ;READS ASCII FROM
               XIS  0          ;DATA BUS
               X 0
               LDD 1,0D
               RC              ;CHAR.  MSB=0 THEN YES
               AISC 15         ;MSB<>0 THEN NO
               JMP SPECIAL
               AISC 01
               LDD 1,0E        ;STORE ASCII IN RAM
               CAB
               LDD 1,0F
               XABR
               LDD 1,0C        ;MSB IN 1,0C
               X 2
               LDD 1,0D        ;LSB IN 1, 0D
               X 0
```

Section 2 of COPS Software (Continued)

```
               JSR CURFOR
               LBI 0,01       ;SENDS INTERRUPT TO
               OBD            ; HOST.  CHAR. IS
               LBI 0,0        ; STORED IN RAM
               OBD
               JMP START

          ;SPECIAL CHARS. (CR, LF, CLEAR DISPLAY)

CURFOR:        LDD 1,0E       ;MOVES CURSOR FORWARD ONE
               COMP           ;SPACE.  IF CURSOR IS
               AISC 01        ;MOVED BEYOND THE END OF
               JMP OK         ;DISPLAY, IT WRAPS AROUND
               AISC OF        ;TO THE OTHER END.  DATA IS
               XAD 1,0E       ;NOT DESTROYED BY MOVING
               CLRA           ;CURSOR
               AISC 01
               LBI 1,0F
               XOR
               JMP SKIP
OK:            COMP
SKIP:          LBI 1,0E
               X 0
               RET
CURBAC:        LDD 1,0F       ;MOVES CURSOR BACK ONE
               AISC 01        ;CHARACTER.  DOES NOT
               JMP GOOD       ;DESTROY DATA AS IT IS MOVED
               LBI 1,0E       ;IF MOVED BEYOND THE
               CLRA           ;END OF THE DISPLAY IT
               AISC 01        ;WRAPS AROUND TO THE OTHER
               XOR            ;END
               X 0
               JMP START
GOOD:          XAD 1,0F
               JMP START

SPECIAL:       LDD 1,0C       ;CONTROL CHAR. HAS BEEN
               AISC 03        ;DETECTED
               JMP NOTRET
               JMP RESET      ;RETURN CLEARS DISPLAY,STARTS
                              ;PROGRAM OVER
NOTRET:        AISC 01        ;NOT RETURN, CHECK FOR CURSOR
               JMP CFOR       ;FORWARD
               JMP CURBAC     ;BY DEFAULT, CURSOR BACKWARDS

CFOR:          JSR CURFOR
               JMP START

          ;DISPLAY LOOP
```

Section 3 of COPS Software

```
;THIS IS THE DISPLAY LOOP OF THE PROGRAM. UNLESS A NEW CHARACTER
;HAS BEEN ENTERED AND IS BEING STORED, THE PROGRAM IS ALWAYS IN
;THIS DISPLAY LOOP. IT LOOKS UP THE CHARACTER FONT, SHIFTS THE
;CHARACTER DATA OUT THE SERIAL PORT AND CONTROLS THE GRID DRIVER.

START:         LBI 2,15       ;DISPLAY LOOP POINTER
               JSR HERE       ;GOTO DISPLAY LOOP
               LBI 3,03       ;SECOND DISPLAY LOOP POINTER
               JSR HERE       ;GOTO DISPLAY LOOP
               OGI 09         ;LOADS A 1 IN GRID DRIVER
               OGI OD
               OGI 09

               JMP START

          ;CHECKS FOR NEW CHAR

HERE:          RC
               ININ
               AISC 15
               JMP OLDCHR
               JMP NEW

          ;DISPLAY LOOP FOR OLD CHAR AND
          ;  LOOK UP

OLDCHR:        LD 2           ;LOOKS UP FIRST BYTE OF CHR.FONT
               JSR DATA4      ;    200H+ASCII VALUE
               AISC 06        ;ADDS 06H TO MSB OF ASCII
               JSR DATA2      ;LOOKS UP SECOND BYTE OF CHR FONT
               AISC 0A        ;ADDS 0AH TO MSB OF ASCII
               JSR DATA2      ;LOOKS UP THIRD BYTE OF CHR. FONT
               JSR DATA3      ;LOOKS UP THIRD BYTE OF CHR. FONT
                              ;    AT 300H+ASCII VALUE
               AISC 06        ;ADDS 06H TO MSB OF ASCII VALUE
               OGI 02         ;TURNS ON BLANKING CONTROL
               JSR DATA3      ;LOOKS UP LAST BYTE OF CHR. FONT
          ;CLOCKS A 0 IN GRID DRIVER

               OGI 0A         ;ENABLE,BLANKING CONTROL
               OGI 0E         ;ENABLE,BLANKING CONTROL,CLOCK
               OGI 0A         ;ENABLE,BLANKING CONTROL
               OGI 00         ;A 0 SHIFTED IN

               LD 0
               XDS 2
               JMP HERE
               RET

RIGHT:         LBI 3,15
               CQMA
               JSR SHIFT      ;OUTPUTS A
               X 0            ;NEW DATA
               JSR SHIFT      ;OUTPUTS A
               LEI 01         ;COUNTER MODE
               LDD 3,14       ;1,0 IN A
               XABR           ;A IN BR
               LDD 3,13       ;1,1 IN A
               CAB            ;A IN BD
               LD 2
               RET
```

Section 3 of COPS Software (Continued)

```
POINTER:       LEI 01         ;COUNTER MODE
               XAS            ;A IN SIO
               XABR           ;BR IN A
               AISC 02        ;ADD 2
               XAD 3,14       ;A IN 1,0
               CBA            ;BD IN A
               XAD 3,13       ;A IN 1,1
               LBI 3,15
               XAS            ;SIO IN A
               LEI 08         ;SERIAL MODE
               JMP RIGHT
          ;SHIFTS OUT SERIAL PORT


SHIFT:         LEI 08         ;THIS ROUTINE SHIFTS THE DATA
               SC             ;FROM THE SI/O REGISTER OUT
               XAS            ;THE SERIAL PORT WITH EACH
               NOP            ;CLOCK CYCLE
               NOP
               RC
               XAS
               RET

          .=0200
DATA3:         LQID
               JMP RIGHT
DATA4:         LQID
               JMP POINTER

          .=0300
DATA3:         LQID
               JMP RIGHT
```

Section 4 of COPS Software

```
;THE CHARACTER FONTS FOR THE CHARACTERS WITH ASCII VALUES BETWEEN
20H AND 5AH HAVE BEEN STORED IN THIS SECTION OF THE PROGRAM.

          ;DATA FOR FIRST 2 BYTES OF EACH
          ;    CHAR.

          .=0220
     .WORD 001, 001, 001, 021, 021, 0C1, 061, 001
     .WORD 031, 001, 041, 011, 001, 011, 001, 001
     .WORD 071, 001, 041, 081, 011, 0E1, 031, 081
     .WORD 061, 061, 001, 001, 001, 021, 001, 041
     .WORD 071, 031, 081, 071, 081, 0F1, 0F1, 071
     .WORD 0F1, 081, 081, 0F1, 0F1, 0F1, 0F1, 071
     .WORD 0F1, 071, 0F1, 061, 081, 0F1, 0F1, 0F1
     .WORD 0C1, 0C1, 081
     ;DATA FOR SECOND 2 BYTES OF EACH
     ;    CHAR.

     .=0280
.WORD 000, 000, 0C1, 0F9, 0A4, 095, 02D, 000
.WORD 088, 000, 054, 020, 000, 020, 000, 014
.WORD 01D, 082, 003, 005, 058, 045, 0AC, 001
.WORD 02D, 023, 000, 000, 020, 058, 001, 001
.WORD 00D, 0AE, 0F3, 00D, 0F3, 02F, 02F, 00D
.WORD 02E, 003, 00D, 02E, 00E, 08E, 08E, 00D
.WORD 02F, 00D, 02F, 025, 001, 00C, 008, 00C
.WORD 056, 040, 017

     ;THIRD 2 BYTES OF DATA FOR EACH CHAR.

     .=02C0
.WORD 000, 0E3, 000, 0AC, 0FB, 040, 0A5, 083
.WORD 00A, 002, 0F3, 0F1, 034, 040, 008, 040
.WORD 046, 0F7, 02E, 046, 021, 086, 046, 02E
.WORD 046, 046, 0A0, 0B4, 0A0, 0A0, 015, 022
.WORD 0E6, 042, 04E, 006, 00E, 046, 042, 046
.WORD 040, 0F7, 006, 0A0, 004, 080, 0E0, 006
.WORD 042, 026, 062, 046, 0F3, 004, 008, 034
.WORD 040, 070, 046

     ;FOURTH TWO BYTES OF DATA FOR EACH CHAR.

     .=0320
.WORD 000, 008, 007, 0F7, 0AA, 031, 028, 000
.WORD 008, 02A, 049, 080, 000, 080, 000, 001
.WORD 01D, 018, 09C, 09D, 0F7, 01D, 09C, 084
.WORD 09C, 0AC, 000, 000, 022, 041, 041, 08C
.WORD 0DC, 082, 09C, 01C, 01C, 09C, 084, 09C
.WORD 080, 01C, 0EF, 022, 018, 002, 020, 01C
.WORD 084, 02C, 0A4, 09C, 00C, 018, 028, 010
.WORD 041, 009, 01D

     ;LAST BYTES OF DATA FOR EACH CHAR.

     .=0380
.WORD 000, 000, 000, 082, 084, 064, 0A0, 000
.WORD 000, 083, 044, 001, 000, 001, 000, 004
.WORD 0C7, 020, 026, 0CC, 080, 0C9, 0C8, 00E
.WORD 0C6, 087, 000, 000, 028, 082, 001, 006
.WORD 027, 0E3, 0C6, 044, 0C7, 028, 008, 0C5
.WORD 0EF, 028, 008, 028, 020, 0EF, 0EF, 0C7
.WORD 006, 0A7, 026, 0C4, 008, 0CF, 08F, 0CF
.WORD 06C, 00C, 02C

.END
```

MICROWIRE™ Serial Interface

National Semiconductor
Application Note 452
Abdul Aleaf

INTRODUCTION

MICROWIRE is a simple three-wire serial communications interface. Built into COPS™, this standardized protocol handles serial communications between controller and peripheral devices. In this application note are some clarifications of MICROWIRE logical operation and of hardware and software considerations.

LOGICAL OPERATION

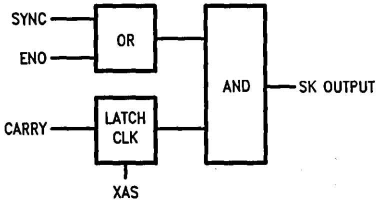
The MICROWIRE interface is essentially the serial I/O port on COPS microcontrollers, the SIO register in the shift register mode. SI is the shift register input, the serial input line to the microcontroller. SO is the shift register output, the serial output line to the peripherals. SK is the serial clock; data is clocked into or out of peripheral devices with this clock.

It is important to examine the logical diagram of the SIO and SK circuitry to fully understand the operation of this I/O port *(Figure 1).*

The output at SK is a function of SYNC, EN0, CARRY, and the XAS instruction. If CARRY had been set and propagated to the SKL latch by the execution of an XAS instruction, SYNC is enabled to SK and can only be overridden by EN0 *(Figure 2).* Trouble could arise if the user changes the state of EN0 without paying close attention to the state of the latch in the SK circuit.

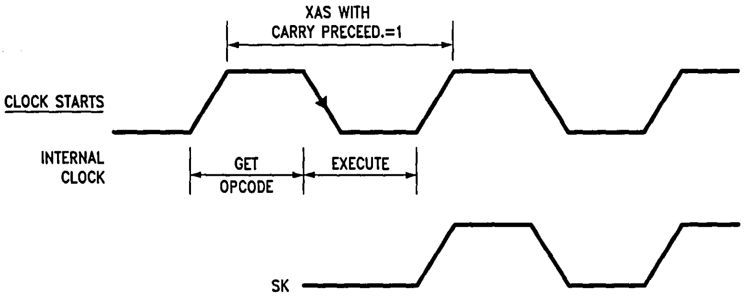
If the latch is set to a logical high and the SIO register enabled as a binary counter, SK is driven high. From this state, if the SIO register is enabled as a serial shift register, SK will output the SYNC pulse immediately, without any intervening XAS instruction.

The SK clock (SYNC pulse) can be terminated by issuing an XAS instruction with CARRY = 0 *(Figure 3).*



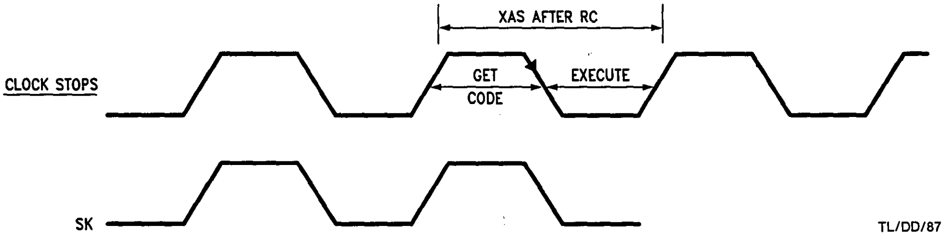
TL/DD/8796-1

FIGURE 1. Logical Diagram of SK Circuit



TL/DD/8796-2

FIGURE 2. SK Clock Starts



TL/DD/8796-3

FIGURE 3. SK Clock Stops

The SIO register can be compared to four master-slave flip-flops shown in *Figure 4*. The masters are clocked by the rising edges of the internal clock. The slaves are clocked by the falling edges of the internal clock. Upon execution of an XAS, the outputs of the masters are exchanged with the contents of the accumulator (read and overdrive) in such a way that the new data are present at the inputs of the four slaves when the falling edge of the internal clock occurs. The content of the accumulator is, therefore, latched respectively in the four slave flip-flops and bit 3 appears directly on SO.

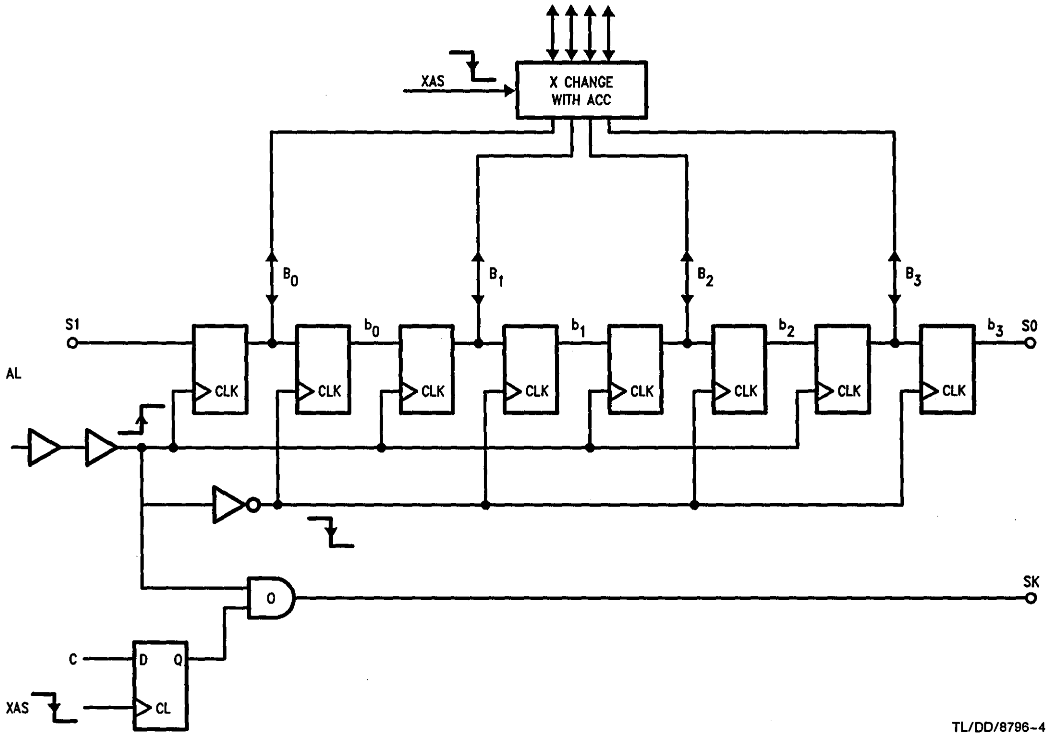
This means that:

a) SO will be shifted out upon the falling edges of SK and will be stable during rising edges of SK.

b) SI will be shifted in upon the rising edge of SK, and will be stable when executing, i.e., an XAS instruction.

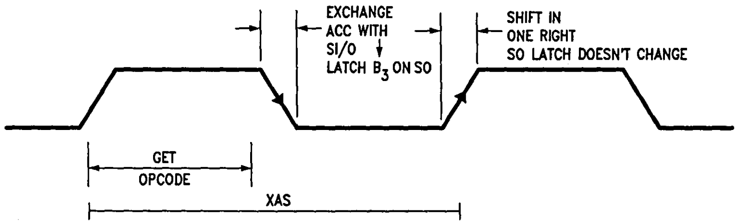
The shifting function is automatically performed on each of the four instruction cycles that follow an XAS instruction *(Figure 5)*.

When the SIO register is in the shift register mode (EN0 = 0), it left shifts its contents once each instruction cycle. The data present on the SI input is shifted into the least significant bit (bit 0) of the serial shift register. SO will output the most significant bit of the SIO register (bit 3) if EN3 = 1. Otherwise, SO is held low. The SK is a logic controlled clock which issues a pulse each instruction cycle. To ensure that the serial data stream is continuous, an XAS instruction must be executed every fourth time. Serial I/O timing is related to instruction cycle timing in the following way:



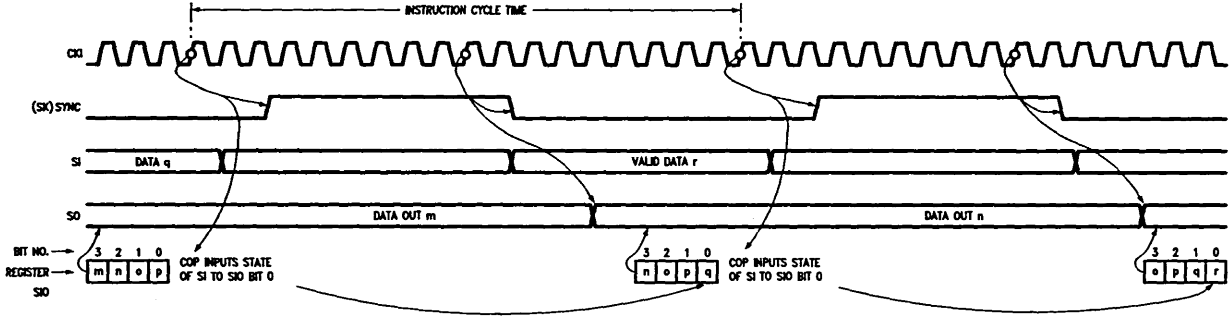
TL/DD/8796-4

FIGURE 4



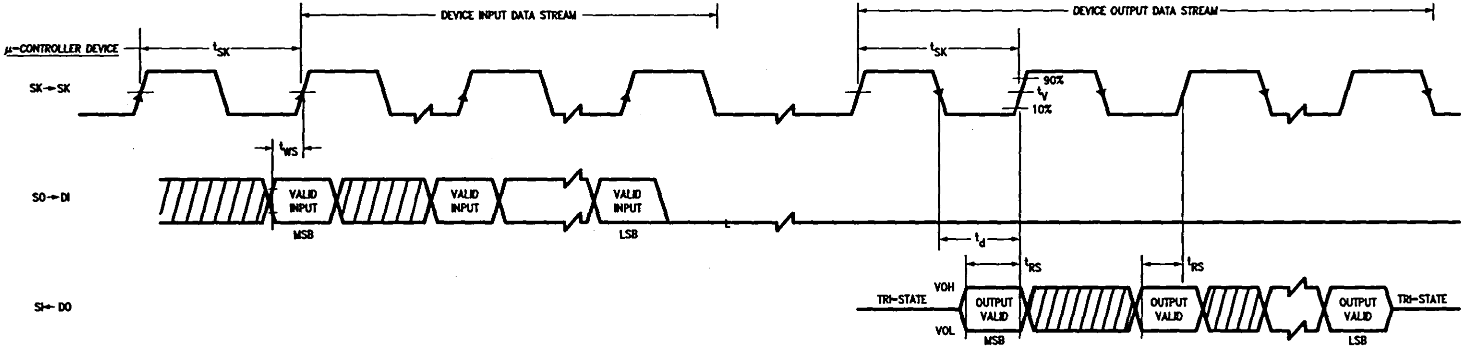
TL/DD/8796-5

FIGURE 5. XAS Sequence



TL/DD/8796-6

FIGURE 6. Serial I/O Timing



TL/DD/8796-7

FIGURE 7. MICROWIRE Serial Data Exchange Timing

To write to device: $t_{ns} > t_{setup}$

To read from device: $t_d < t_{SK} - t_r$; $t_{RS} > t_{SK}/4$

Where: t_{ws} is MICROWIRE write data-in (DI) setup time,

t_{setup} is device data sheet min data setup time to latch in valid data,

t_{SK} is system clock (SK) cycle time (Recommended 50% duty cycle),

t_r is rise time (10% to 70% bout) of system clock (SK),

t_d is device actual delay time before data-out (DO) valid and

t_{RS} is minimum data setup time for controller to shift-in valid data

The first clock rising edge of the instruction cycle triggers the low-to-high transition of SYNC output via SK. At this time, the processor reads the state of SI into SIO bit 0, shifting the current bits 0–2 left. Halfway through the cycle (shown in *Figure 6* as the eight clock rising edge), SK is reset low and the new SIO bit 3 is outputted via SO.

INTERFACING CONSIDERATIONS

To ensure data exchange, two aspects of interfacing have to be considered: 1) serial data exchange timing; 2) fan-out/fan-in requirements. Theoretically, infinite devices can access the same interface and be uniquely enabled sequentially in time. In practice, however, the actual number of devices that can access the same serial interface depends on the following: system data transfer rate, system supply requirement capacitive loading on SK and SO outputs, the fan-in requirements of the logic families or discrete devices to be interfaced.

HARDWARE INTERFACE

Provided an output can switch between a HIGH level and a LOW level, it must do so in a predetermined amount of time for the data transfer to occur. Since the transfer is strictly synchronous, the timing is related to the system clock (SK) *(Figure 7)*. For example, if a COPS controller outputs a value at the falling edge of the clock and is latched in by the peripheral device at the rising edge, then the following relationship has to be satisfied:

$$t_{DELAY} + t_{SETUP} \leq t_{CK}$$

where t_{CK} is the time from data output starts to switch to data being latched into the peripheral chip, t_{SETUP} is the setup time for the peripheral device where the data has to be at a valid level, and t_{DELAY} is the time for the output to read the valid level. t_{CK} is related to the system clock provided by the SK pin of the COPS controller and can be increased by increasing the COPS instruction cycle time.

The maximum t_{SETUP} is specified in the peripheral chip data sheets. The maximum t_{DELAY} allowed may then be derived from the above relationship.

Most of the delay time before the output becomes valid comes from charging the capacitive load connected to the output. Each integrated circuit pin has a maximum load of 7 pF. Other sources come from connecting wires and connection from PC boards. The total capacitive load may then be estimated. The propagation delay values given in data sheets assume particular capacitive loads (e.g. $V_{CC} = 5V$, $V_{OH} = 0.4V$, loading = 50 pF, etc.).

If the calculated load is less than the given load, those values should be used. Otherwise, a conservative estimate is to assume that the delay time is proportional to the capacitive load.

If the capacitive load is too large to satisfy the delay time criterion, then three choices are available. An external buffer may be used to drive the large load. The COPS instruction cycle may be slowed down. An external pullup resistor may be added to speed up the LOW level to HIGH level transition. The resistor will also increase the output LOW level and increase the HIGH level to LOW level transition time, but the increased time is negligible as long as the output LOW level changes by less than 0.3V. For a 100 pF load, the standard COPS controller may use a 4.7k external resistor, with the output LOW level increased by less than 0.2V. For the same load, the low power COPS controller may use a 22k resistor, with the SO and SK LOW levels increased by less than 0.1V.

Besides the timing requirements, system supply and fan-out/fan-in requirements also have to be considered when interfacing with MICROWIRE. For the following discussion, we assume single supply push-pull outputs for system clock (SK) and serial output (SO), high-impedance input for serial input (SI).

To drive multi-devices on the same MICROWIRE, the output drivers of the controller need to source and sink the total maximum leakage current of all the inputs connected to it and keep the signal level within the valid logic "1" or logic "0". However, in general, different logic families have different valid "1" and "0" input voltage levels. Thus, if devices of different types are connected to the same serial interface, the output driver of the controller must satisfy all the input requirements of each device. Similarly, devices with TRI-STATE® outputs, when connected to the SI input, must satisfy the minimum valid input level of the controller and the maximum TRI-STATE leakage current of all outputs.

So, for devices that have incompatible input levels or source/sink requirements, external pull-up resistors or buffers are necessary to provide level-shifting or driving.

SOFTWARE INTERFACE

The existing MICROWIRE protocol is very flexible, basically divided into two groups:

1) 1AAA.....ADDD.....D

where leading 1 is the start bit and leading zeroes are ignored.

AAA.....A is device variable instruction/address word.

DDD.....D is variable data stream between controller and device.

2) No start bit, just bit stream, i.e., bbb.....b

where b is a variable bit stream. Thus, device has to decode various fields within the bit stream by counting exact bit position.

SERIAL I/O ROUTINES

Routines for handling serial I/O are provided below. The routines are written for 16-bit transmissions, but are trivially expandable up to 64-bit transmissions by merely changing the initial LBI instruction. The routines arbitrarily select register 0 as the I/O register. It is assumed that the external device requires a logic low chip select. It is further assumed that chip select is high and SK and SO are low on entry to the routines. The routines exit with chip select high, SK and SO low. GO is arbitrarily chosen as the chip select for the external device.

SERIAL DATA OUTPUT

This routine outputs the data under the conditions specified above. The transmitted data is preserved in the microcontroller.

```
OUT2:  LBI  0,12   ; point to start of
                     data word
       SC
       OGI  14     ; select the external
                     device
```

TABLE I. MICROWIRE Standard Family

Features		Part Number							
		DS3906	MM545X	COP470	COP472	COP430 (ADC83X)	COP498/499	COP452L	COP494 (NMC9306)
GENERAL									
Chip Function		AM/PM PLL	LED Display Driver	VF Display Driver	LCD Display Driver	A/D	RAM & Timer	Frequency Generator	E^2PROM
Process		ECL	NMOS	PMOS	CMOS	CMOS	CMOS	NMOS	NMOS
V_{CC} Range		4.75V–5.25V	4.5V–11V	−9.5V to −4.5V	3.0V–5.5V	4.5V–0.3V	2.4V–5.5V	4.5V–6.3V	4.5V–5.5V
Pinout		20	40	20	20	8/14/20	14/8	14	14
HARDWARE INTERFACE									
Min V_{IH}/Max V_{IL}		2.1V/0.7V	2.2V/0.8V	−1.5V/−4.0V	0.7V_{CC}/0.8V	2.0V/0.8V	0.8V_{CC}/0.4V_{CC}	2.0V/0.8V	2.0V/0.8V
SK Clock Range		0–625 kHz	0–500 kHz	0–250 kHz	4–250 kHz	10–200 kHz	4–250 kHz	25–250 kHz	0–250 kHz
Write Data DI	Setup Min	0.3 μs	0.3 μs	1.0 μs	1 μs	0.2 μs	0.4 μs	800 ns	0.4 μs
	Hold Min	0.8 μs	(3)	50 ns	100 ns (Note 1)	0.2 μs	0.4 μs	1.0 μs	0.4 μs
Read Data Prop Delay		(Note 4)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	2 μs (Note 2)	1 μs (Note 2)	2.0 μs
Chip Enable	Setup	0.3 μs	0.4 μs	1.0 μs Min	1 μs (Note 1)	0.2 μs	0.2 μs (Note 1)	(Note 3)	0.2 μs
	HOLD	0.8 μs	(Note 3)	1.0 μs Min	1 μs (Note 2)	0.2 μs	0 (Note 2)	(Note 3)	0
Max Frequency Range	AM	8 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
	FM	120 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
Max Osc Freq.		(Note 3)	(Note 3)	250 kHz	(Note 3)	(Note 3)	2.1 MHz (−21) 32 kHz (−15)	256–2100 kHz (−4) 64–525 kHz (−2)	(Note 3)
SOFT									
Serial I/O Protocol		11D1...D20	1D1...D35	8 Bits At a Time	b1...b40	1xxx	1yyxxD6...D0 Start Bit	1yxxxx	1AA...DD
Instruction/ Address Word		None	None	None	None	(Note 4)	(Note 4)	(Note 4)	(Note 4)

Note 1: Reference to SK rising edge.

Note 2: Reference to SK falling edge.

Note 3: Not defined.

Note 4: See data sheet for different modes of operation.

```
          LEI  8     ; enable shift
                       register mode
          JP   SEND2
SEND1:    XAS
SEND2:    LD         ; data output loop
          XIS
          JP   SEND1
          XAS        ; send last data
          RC
          CLRA
          NOP
          XAS        ; turn SK clock off
          OGI  15    ; deselect the device
          LEI  0     ; turn SO low
          RET
```

The code for reading serial data is almost the same as the serial output code. This should be expected because of the nature of the SIO register and the XAS instruction.

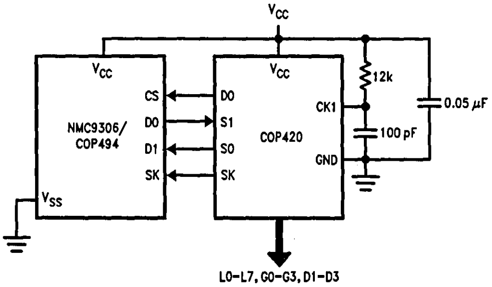
MICROWIRE STANDARD FAMILY

A whole family of off-the-shelf devices exists that is directly compatible with MICROWIRE serial data exchange standard. This allows direct interface with the COPS family of microcontrollers.

Table I provides a summary of the existing devices and their functions and specifications.

TYPICAL APPLICATION

Figure 8 shows pin connection involved in interfacing an NMC9306/COP494 E²PROM with the COP420 microcontroller.

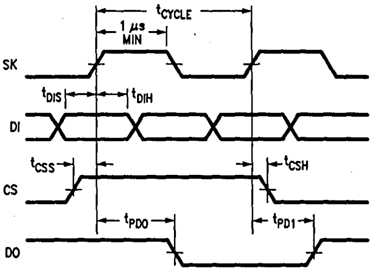


TL/DD/8796-8

FIGURE 8. NMC9306/COP494-COP420 Interface

The following points have to be considered:

1. For COP494 the SK clock frequency should be in the 0 kHz–250 kHz range. This is easily achieved with COP420 running at 4 μs–10 μs instruction cycle time (SK period is the COP420 instruction cycle time). Since the minimum SK clock high time is greater than 1 μs, the duty cycle is not a critical factor as long as the frequency does not exceed the 250 kHz max.
2. CS low period following an E/W instruction must not exceed 30 ms maximum. It should be set at typical or minimum spec of 10 ms. This is easily done in software using the SKT timer on COP420.



TL/DD/8796-9

FIGURE 9. NMC9306/COP494 Timing

3. As shown in WRITE timing diagram, the start bit on DI must be set by a "0" to "1" transition following a CS enable ("0" to "1") when executing any instruction. One CS enable transition can only execute one instruction.
4. In the read mode, following an instruction and data train, the DI can be a "don't care," while the data is being outputted, i.e., for the next 17 bits or clocks. The same is true for other instructions after the instruction and data has been fed in.
5. The data-out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential. If CS is held on after all 16 of the data bits have been outputted, the DO will output the state of DI until another CS LO to HI transition starts a new instruction cycle.
6. After a read cycle, the CS must be brought low for one SK clock cycle before another instruction cycle starts.

INSTRUCTION SET

Commands	Opcode	Comments
READ	10000A3A2A1A0	Read Register 0–15
WRITE	11000A3A2A1A0	Write Register 0–15
ERASE	10100A3A2A1A0	Erase Register 0–15
EWEN	111000 0 0 1	Write/Erase Enable
ENDS	111000 0 1 0	Write/Erase Disable
***WRAL	111000 1 0 0	Write All Registers
ERAL	111000 1 0 1	Erase All Registers

All commands, data in, and data out are shifted in/out on rising edge of SK clock.

Write/erase is then done by pulsing CS low for 10 ms.

All instructions are initiated by a LO-HI transition on CS followed by a LO-HI transition on DI.

READ— After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE— Write command shifted in followed by data in (16 bits) then CS pulsed low for 10 ms minimum.

ERASE

ERASE ALL—Command shifted in followed by CS low.

WRITE ALL—Pulsing CS low for 10 ms.

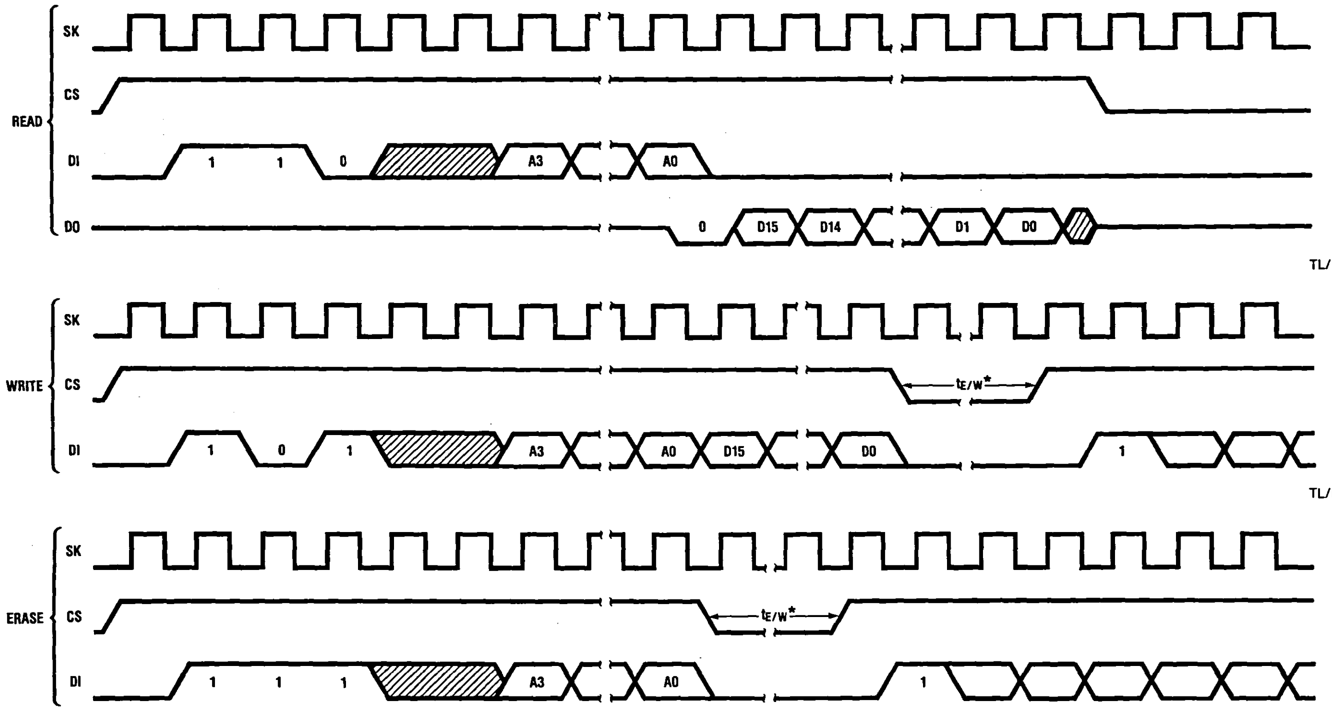
WRITE

ENABLE/DISABLE—Command shifted in.

***(This instruction is not speced on Data sheet.)

TIMING WAVEFORMS

Instruction Timing



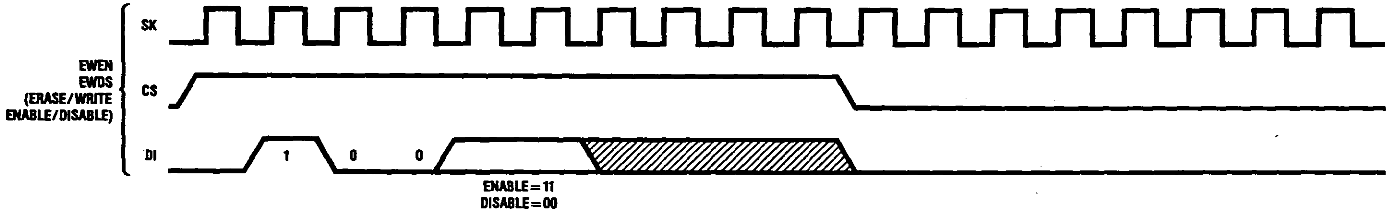
TL/DD/8796-10

TL/DD/8796-11

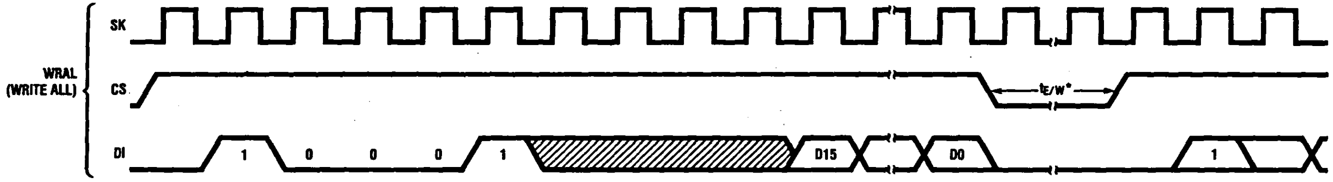
TL/DD/8796-12

*$t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

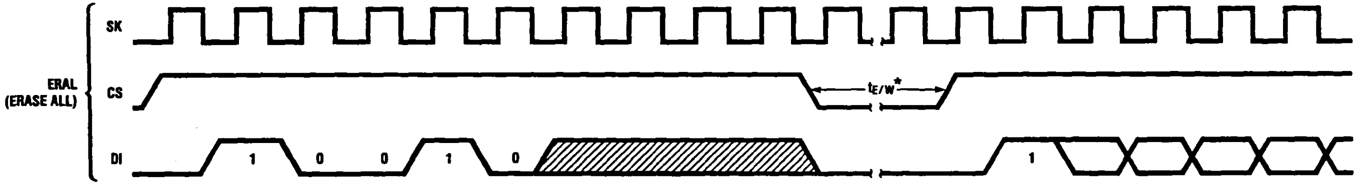
TIMING WAVEFORMS (Continued)



TL/DD/8796-13



TL/DD/8796-14



TL/DD/8796-15

*$t_{E/W}$ measured to rising edge of SK or CS, whichever occurs last.

I/O ROUTINE TO EVALUATE COP494

```
                          .TITLE    E494,  "I/O ROUTINE TO EVALUATE COP494"
    01A4                  .CHIP     420
    0000                  .PAGE     0

                ;THIS IS I/O ROUTINE TO EVALUATE COP494
                ;
                ;

                ;RAM VARIABLES DECLARATIONS:
    000E                  COMMAND = 0, 14           ;494 8BITS INST/ADDR WORD
    001C                  RWDATA  = 1, 12           ;494 16BITS R/W DATA BUFFER

000 00          PON:      CLRA                      ;POWER-ON INIT
001 32                    RC                        ;RESET SK CLOCK
002 4F                    XAS
003 3F          CLRAM:    LBI      3, 0             ;CLEAR RAM FROM 7, 0 TO 0, 15
004 00          CLR:      CLRA                      ;
005 04                    XIS                       ;
006 C4                    JP       CLR              ;CONTI CLEAR REG
007 12                    XABR                      ;(A) TO BR
008 5F                    AISC     15               ;REG 0 CLEARED?
009 600F        DONE:     JMP      C494DR           ;Y, DONE CLEAR RAM, CALL 494 D
00B 12                    XABR                      ;N, DEC BR
00C C4                    JP       CLR              ;CONTI CLEAR REG TILL DONE
00D 44                    NOP
00E 44                    NOP

                ;***      START 494 DRIVER SAMPLE CALLING SEQUENCE   ***

                C494DR:                             ;INIT CALLING SEQUENCE
00F 3350                  OGI      0                ;GO=L TO DESELECT 494
011 3368                  LEI      8                ;ENABLE SIO AS S.R.
                ERASE:
013 0D                    LBI      COMMAND          ;PRELOAD 494 ERASE REG A3-A0
014 7C                    STII     0C               ;PRELOAD 494 ERASE INST
015 70                    STII     0                ;SELECT REG A3-A0
016 690E                  JSR      WI4P4            ;SEND IT
                WEEN:
01B 0D                    LBI      COMMAND          ;LOAD 494 WHEN REG A3-A0
019 73                    STII     3                ;PRELOAD 494 WREN INST
01A 70                    STII     0                ;SELECT REG A3-A0
01B 690E                  JSR      WI494            ;SEND IT
                WRITE:
01D 0D                    LBI      COMMAND          ;PRELOAD WR REG A3-A0
01E 74                    STII     4                ;PRELOAD 494 WRITE INST
01F 70                    STII     0                ;SELECT REG A3-A0
020 1B                    LBI      RWDATA           ;PRELOAD 494 SAMPLE WRITE DATA
021 75                    STII     5
022 7A                    STII     ON
023 75                    STII     5
```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```
024 7A              STII    0A
025 6900            JSR     WD494           ;SEND THEM TO 494
          READ:
027 0D              LBI     COMMAND         ;PRELOAD READ REG A3-A0
028 78              STII    8               ;PRELOAD 494 READ INST
029 70              STII    0               ;SELECT REG A3-A0
02A 6908            JSR     RD494           ;READ 494 DATA BACK VIA SI
02C 44              NOP
02D 44              NOP

    0080            .PAGE   2               ;SUBROUTINE PAGE
080 32    SETUP:    RC                      ;RESET SK BEFORE SELECT 494
081 4F              XAS
082 3351            OGI     1               ;G0=1 TO SELECT 494
084 00              CLRA                    ;ENSURE SO=L BEFORE GEN START B
085 22              SC                      ;
086 4F              XAS                     ;TURN ON SK CLOCK
087 00              CLRA                    ;GENERATE 494 START BIT
088 51              AISC    1               ;
089 22              SC                      ;
08A 4F              XAS                     ;SEND IT AS MSB VIA SO
08B 0D              LBI     COMMAND         ;FETCH 1ST INST/ADDR WORD
08C 05              LD
08D 44              NOP                     ;
08E 4F              XAS                     ;SEND IT (MSB OF INST FIRST)
08F 0E              LBI     COMMAND+1       ;FETCH 2ND INST/ADDR NIBBLE
090 05              LD
091 44              NOP
092 4F              XAS                     ;SEND IT
093 1B              LBI     RWDATA          ;POINT TO READ/WRITE DATA BUFFER
094 48              RET                     ;RET OF SETUP

095 00    TWEDLY:   CLRA                    ;VPP WIDTH, TWE>20MS @ 4Us/INST
096 5B    TWECONT:  AISC    11              ;5 SKT LOOPS?
097 99              JP      . + 2           ;N,CONTI
098 48    TWEDONE:  RET                     ;Y,DONE
099 41              SKT                     ;
09A 99              JP      . -1            ;
09B 96              JP      TWECONT         ;CONTI TWE TIME

09C 48    RET;      RET                     ;2 CYCLES DELAY

    0100            .PAGE   4               ;

          ;***      START 494 I/O DRIVER SUBROUTINE    ***

100 80    WD494:    JSRP    SETUP           ;ENTRY TO WRITE 494 REG A3-A0
101 05    RWLOOP:   LD                      ;R/W 494 16 DATA BITS
102 4F              XAS                     ;
103 04              XIS                     ;
```

I/O ROUTINE TO EVALUATE COP494 (Continued)

```
104 C1               JP    RWLOOP
105 3350             OGI   0        ;DESELECT 494 AFTER R/W DATA
107 D1               JP    FINI     ;
108 80      RD494:   JSRP  SETUP    ;ENTRY TO RD 494 REG A3-A0
109 00               CLRA           ;FINISH SEND OUT A3-A0 VIA SO
10A 44               NOP            ;
10B 44               NOP            ;WAIT 1BIT TIME FOR VALID D15
10C 44               NOP
10D C1               JP    RWLOOP     ;
10E 80      WI494:   JSRP  SETUP    ;ENTRY TO WRITE INST TO 494
10F 00               CLRA           ;ENSURE SO = L
110 4F               XAS            ;
111 00      FINI:    CLRA           ;ENSURE SO = L BETWEEN INST
112 3350             OGI   0        ;DESELECT 494 BETWEEN INST WRIT
114 32               RC             ;
115 4F               XAS            ;TURN OFF SK CLOCK
116 95               JSRP  TWEDLY   ;DELAY TWE >20MS TO PULSE VPP=21
117 48               RET            ;RET OF WD494 OR RD494 OR WI494

                     .END
```

SOFTWARE DEBUG OF SERIAL REGISTER FUNCTIONS

In order to understand the method of software debug when dealing with the SIO register, one must first become familiar with the method in which the COPS MOLE™ (Development System) BREAKPOINT and TRACE operations are carried out. Once these operations are explained, the difficulties which could arise when interrogating the status of the SIO register should become apparent.

SERIAL OUT DURING BREAKPOINT

When the MOLE BREAKPOINTs, the COPS user program execution is stopped and execution of a monitor-type program, within the COP device, is started. At no time does the COP part "idle." The monitor program loads the development system with the information contained in the COP registers.

Note also that single-step is simply a BREAKPOINT on every instruction.

If the COP chip is BREAKPOINTed while a serial function is in progress, the contents of the SIO register will be destroyed.

By the time the monitor program dumps the SIO register to the MOLE, the contents of the SIO register will have been written over by clocking in SI. To inspect the SIO register using BREAKPOINT, an XAS must be executed prior to BREAKPOINT; therefore, the SIO register will be saved in the accumulator.

An even more severe consequence is that the monitor program executes an XAS instruction to get the contents of the SIO register to the MOLE. Therefore, the SK latch is dependent on the state of the CARRY prior to the BREAKPOINT. In order to guarantee the integrity of the SIO register, one must carefully choose the position of the BREAKPOINT address.

As can be seen, it is impossible to single-step or BREAKPOINT through a serial operation in the SIO register.

SERIAL OUT DURING TRACE

In the TRACE mode, the user's program execution is never stopped. This mode is a real-time description of the program counter and the external event lines; therefore, the four external event lines can be used as logic analyzers to monitor the state of any input or output on the COPS device. The external event lines must be tied to the I/O which is to be monitored.

The state of these I/O (external event lines) is displayed along with the TRACE information. The safest way to monitor the real-time state of SO is to use the TRACE function in conjunction with the external event lines.

CONCLUSIONS

National's super-sensible MICROWIRE serial data exchange standard allows interfacing to any number of specialized peripherals using an absolute minimum number of valuable I/O pins; this leaves more I/O lines available for system interfacing and may permit the COPS controller to be packaged in a smaller package.

COPS™ Based Automobile Instrument Cluster

National Semiconductor
Application Note 453
Venkata T. Gobburu

ABSTRACT

Dedicated microprocessor systems find increasing applications in automobile instrumentation. Fuel injection systems, digital radio tuners and similar applications employing the microcontroller have become common place. This paper describes a cost effective microcontroller implementation of an automobile instrument cluster by the COPS group of National Semiconductor, Santa Clara. The instrument cluster provides a vacuum fluorescent display of the vehicle speed, engine RPM, odometers, battery voltage, engine oil pressure and the fuel level. A modular design involving a single microcontroller in conjunction with peripherals to aid in data acquisition from the transducers allows the quantities to be computed with high accuracies and displayed on a real time basis. The single microcontroller environment places severe restrictions on the availability of RAM and ROM. Coupled with the requirement of real time operation the application poses a non trivial challenge. A nonvolatile RAM accumulates the mileage covered. Hamming code techniques ensure the integrity of the data contained in the nonvolatile memory. Inclusion of diagnostics allows a rapid and thorough check against improper operation of the microcontroller, peripherals and the nonvolatile memory. This paper describes the implementation with a COP444L containing 128 nybbles of RAM and 2K bytes of ROM. A display updation rate of 16 Hz can be comfortably realized.

Over the microcomputer usage has diversified dramatically in its scope and breadth. Dedicated microprocessor systems find increasing application in automobile instrumentation and control. From its inception the automobile has acquired considerable sophistication. Increasing demands have been made of the car. Fuel efficiency, higher acceleration rates, simplicity of control and improved ride quality rank high in the demands made of the car. In response the automobile engine has evolved into a complex machine. Crude methods to control or monitor its performance no longer suffice. Microprocessor based fuel injection techniques and ignition control are becoming quite ubiquitous.

The automobile instrument cluster monitors the engine and regularly updates a status display for the operator's benefit. Pertinent information includes the vehicle speed, the engine crankshaft rotational speed, oil pressure in the engine cylinders, condition of the battery and the mileage accumulated. The instrument cluster provides a visual feedback link to the operator allowing corrective action to be initiated as the need arises.

THE AUTOMOBILE INSTRUMENT CLUSTER

The heart of the Automobile Instrument Cluster (AIC) lies in obtaining raw data from various transducers and manipulating it to a form suitable for feedback to the human operator. The feedback, normally visual, conveys the vehicle speed, the engine rpm, the engine temperature, oil pressure, the battery voltage and the odometer values. The AIC can be viewed as a collection of either inherently independent or weakly linked subtasks. Each subtask can be further partitioned into three blocks viz. of raw data collection, processing and displaying it. The component subtasks, in spite of their high degree of independence, can be grouped on the basis of signal available from the transducers. Grouping the subtasks modularizes the design. Partitioning the design in this manner highlights two groups, the first requires a frequency to be measured and the second a voltage level. The two major groupings are briefly examined.

Transducers for the vehicle speed monitor the driveshaft rotation. Computing the engine rpm involves measuring the crankshaft revolution rate. The two independent problems can be seen to basically consist of measuring revolution rates. Transducers based on Hall effect phenomena have been used with commendable success. Alternately the fact that mounting magnets around the driveshaft circumference generates a known number of pulses per shaft rotation can be used effectively. A normally open cam operated reed switch with closure to ground creates a simple revolution transducer. In all the cases the transducer generates a frequency proportional to the quantity under consideration. Obviously some signal conditioning is required before using the frequency with digital components. The describing function can be simply stated as

$$V = k \times f \qquad (1)$$

where

V is the quantity under measurement, the vehicle speed or the engine rotational speed

k is a proportionality constant

f is the transducer freqeuncy output

The proportionality constant, k, can be suitably modified to include changes back and forth between British and metric units.

The problem of measuring the transducer output frequency can be restated to be one of measuring the time period. In case of digital frequencies the equation (1) can be rewritten as

$$V = k/(\text{Ton} + \text{Toff}) \qquad (2)$$

where

Ton is the ON time and

Toff is the OFF time

while the remaining symbols retain their definition from the earlier equation.

The remaining quantities such as the engine temperature, oil pressure, battery voltage and available fuel prove to be slow changing ones. The lower dynamics allow them to be transduced as voltage level signals. Equation (3) states the underlying relation and closely resembles the equations stated above.

$$P = k \times v \qquad (3)$$

where

v is the voltage output of the transducer

P is the quantity under measurement

k is the proportionality constant

Evaluating the accumulating mileage depends indirectly upon the vehicle speed subtask. Integrating the signal from the vehicle speed transducer over time allows the mileage to be accumulated. The associated problems of storing the odometer information and ensuring its integrity require error correcting techniques. They are covered in a later section of the paper.

SYSTEM DESCRIPTION

The COPS Group of National Semiconductor, Santa Clara, offers a wide array of microcontrollers and peripherals to suit this application. Judicious selection of peripherals to aid the microcontroller can reinforce the partitioning suggested earlier to considerably simplify the implementation. *Figure 1* presents a functional block diagram of the AIC.

A COP444L four bit microcontroller provides the necessary computing and decision making capability. Equipped with 128 nybbles of RAM space organized in a matrix fashion and 2K ROM space for storage of the control program, the COP444L operating at an instruction cycle rate of 16 microseconds sequentially obtains information from the peripherals and formats the manipulated results to be manageable by the display drivers. Transducers for the vehicle speed and the engine speed provide proportional frequency signals. Two COP452 peripherals, placed in a Waveform Measure Mode, track the ON time and OFF time of the conditioned transducer outputs. Voltage level signals available from the transducers for the engine temperature, oil pressure, battery condition and the fuel tank can be monitored by a COP438, an eight channel A/D converter. An electronically erasable non volatile RAM, the COP494, allows the odometer information to be stored safely under power down conditions.

A combination of LEDs, vacuum fluorescent displays and high intensity lamps comprise the optical elements of the AIC Standard eight segment alphanumeric and bargraph format displays have been used. A 32 segment LED bargraph, controlled by a MM5450 static display driver, displays the engine rpm. Eight segment alphanumeric vacuum fluorescent displays are used for the vehicle speed and the odometer values. Sixteen segment vacuum fluorescent bargraph displays are used for the engine temperature and available fuel quantity. The battery voltage and oil pressure utilize eight segment vacuum fluorescent bargraph displays. Any potentially dangerous situations detected by the COP444L are underlined by high intensity lamps. Five COP470 display drivers multiplex the various displays under the microcontroller's orchestration.

Single pole single throw switches allow the user to select between the British or the metric units, the trip or the accumulated odometer and reset the trip odometer.

SYSTEM DIAGNOSTICS

Diagnostics aid in isolating faulty components within a system. The algorithmic nature of the diagnostic procedure allows it to be implemented via a microprocessor. A great deal of attention has been focused on diagnostics as considerable cost savings can accrue from a microprocessor based scheme minimizing human involvement. Programming the AIC, in addition to its normal functions, with self test capabilities increases its potential for high volume applications. Normally diagnostics imply using independent means to evaluate the system's performance. Attempting to incorporate self test capabilities necessitates adopting an "inside out" strategy. A basic kernel is first evaluated as functioning correctly. Over iterations the kernel expands by establishing correct operation of other modules.

The AIC implementation described in this paper has an extensive repertoire of diagnostics to check the microcontroller and ensure correct operation of the peripherals. The probability of the microcontroller ROM failing proves to be negligibly small compared to a fault developing in the hardware interconnections. Also the idea of encoding in ROM the algorithm to check ROM data proves suspect. Control program stored in the ROM forms the kernel assumed to be functioning correctly. Writing and reading back an alternating pattern of ones and zeros in the microcontroller RAM checks for leakage of data into adjacent locations. Applying a known voltage, derived locally, to one of the four unused channels on the A/D converter allows it to be tested. The architecture of the COP452 peripherals consists of two independent register-counter pairs. The counters count down from the initial value. To test the COP452 both the register counter pairs have to be checked. By placing the two in a Duty Cycle Mode, the counters can be loaded with initial values from the registers and set to count down. The contents of the counters after a predetermined delay can detect incorrect operation of the device. A fault at the level of a register-counter pair can thus be isolated.

The COP494 stores the odometer information. It becomes vital to maintain the integrity of the information stored in the nonvolatile memory. Continuous use of particular locations in the COP494 can result in failures, typically bit dropouts. It is imperative to be capable of recovering from such errors. Requiring a single COP494 unit to last at least the expected lifetime of the vehicle influences the design of the storage scheme. The AIC implementation described in this paper depends upon Hamming encoding techniques to provide single bit error recovery. Subsequent to recovering from a single bit error all data transactions are carried out from a new location. A flashing display sequence alerts the operator of the occurrence of a non-recoverable error. Suspending all normal functions during such conditions can be used to force the vehicle to be taken to an authorized dealer. Breaking up the odometer data into sections allows updating of particular sections as opposed to restoring the whole every time. Such a strategy maximizes the lifetime of the nonvolatile memory.

SOFTWARE DESCRIPTION

The functional objectives of the AIC and the hardware required to realize them have been detailed in earlier sections of the paper. A summary of the software features completes the description and aids in developing a global understanding of the AIC. The AIC software, written in COP microcontroller assembly language, reflects the modular nature of the problem. The finite amount of memory of ROM space available on the COP444L coupled with real time operation requirements makes programming the AIC a non-trivial problem. Each subtask grouping has been organized as a distinct block of code. The microcontroller sequentially processes each subtask. A brief examination of the salient features follows.

It must be borne in mind that the COP452 peripheral captures an instantaneous picture of the frequency. The strength of the magnets, mounted circumferentially on the driveshaft to transduce revolution rate, cannot be precisely controlled. As a result the transducer, although generating a fixed number of pulses per revolution of the driveshaft, produces a pulse train showing both pulse period and duty cycle variations. Directly using the pulse period from the

COP452 leads to erroneous values of the vehicle speed. The computed vehicle speed, under steady vehicle speed conditions, shows excursions on either side of the nominal value. The first AIC implementation studied the application of an essentially single pole filter with different damping constants to exclude the oscillations. Although a sufficiently damped filter can effectively reduce the oscillations the scheme was discarded in lieu of the resulting degradation in response time. The solution lies in basing the vehicle speed computation on pulse period measurements averaged over consecutive pulses. Since the number of pulses per revolution is known, eight in this case, averaging the pulse period over this number minimizes the steady state error and responds fast. The nature of the solution affects the software organization. It falls upon the microcontroller to sample the conditioned output of the transducer and obtain pulse periods for eight consecutive pulses. To achieve this the software adopts a foreground-background organization. Monitoring the transducer output to catch the consecutive pulses forms the background job. The normal functions of the AIC form the foreground job. Additionally a minimal sampling rate has to be maintained to ensure that even at highest attainable vehicle speeds the microcontroller measures consecutive pulses.

The AIC electronically stores the odometer information in the non-volatile memory. Loss of odometer integrity can be disastrous. Consequently the ability to recover from errors in the non-volatile memory becomes very important. The AIC depends on single bit error correcting Hamming coding methods to avoid loss of information. The algorithm processes the odometer nybble fashion and simplifies the related problems of encoding the data prior to storing it and decoding the composite for data retrieval to trivial table lookups. LQID, a powerful member of the microcontroller instruction set, allows an eight bit value to be looked up based on the key value in the addresed RAM location. To minimize ROM space both the encoding and the decoding sections of the algorithm share the same error table and code for table lookups.

The remaining sections of the AIC software, also exhibit a block structure, do not prove to be as subtle. The straight forward code includes routines such as multiplications and divisions to help in the computations and routines allowing the microcontroller to communicate serially over the MICROWIRE™ with the peripherals.

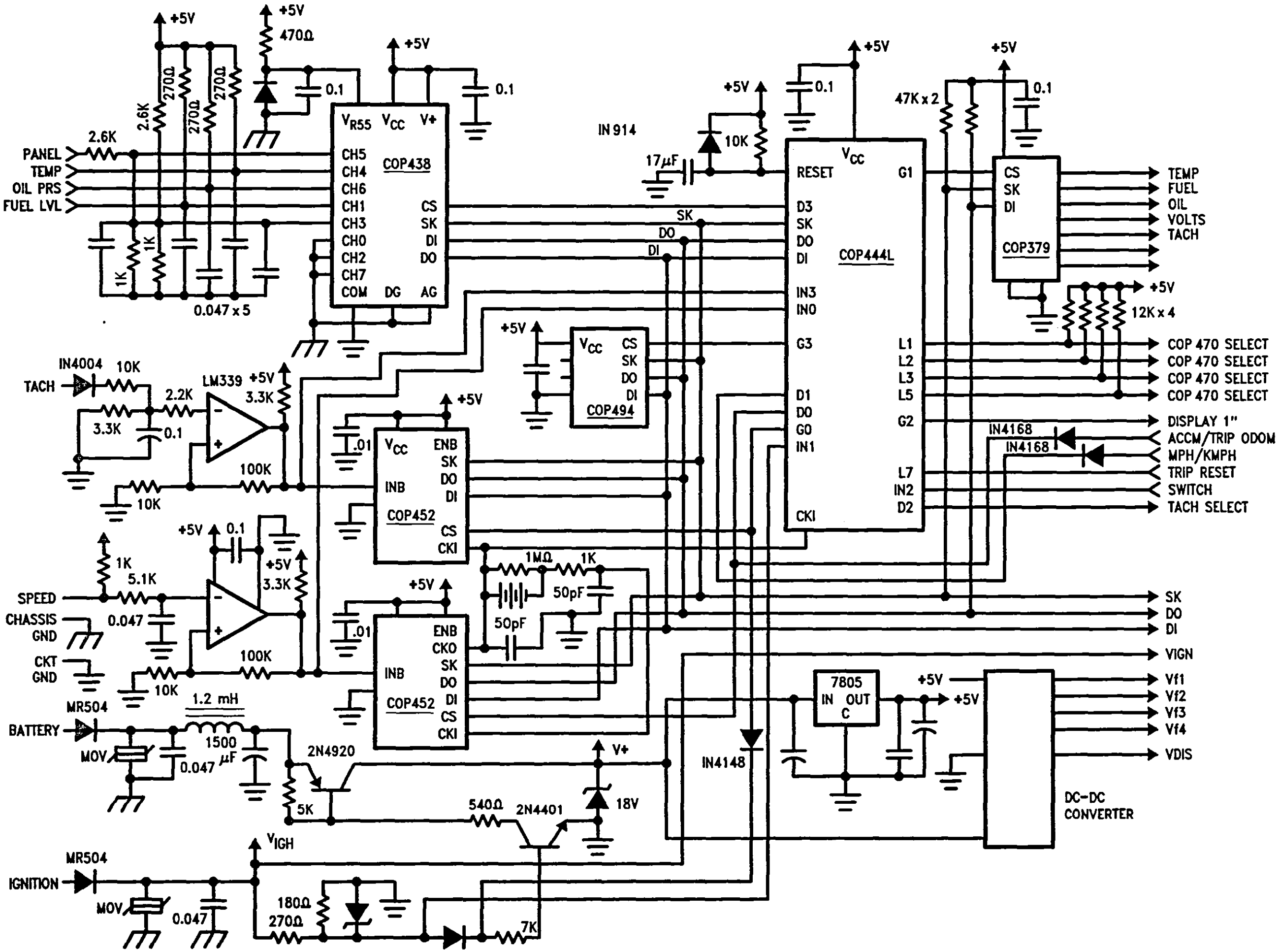
RESULTS AND CONCLUSIONS

The AIC implemented via the COP444L approximately uses 2K of ROM space. The COP444L, running at an instruction cycle time of 16 microseconds, sequences through all the functions in 228 milliseconds. The resulting display updation rate of approximately 4 Hz can be trivially increased to 16 Hz by replacing the COP444L with the equivalently packaged COP440. Table I presents in tabular form the accuracies and speeds at which the different measurements are done. It also shows the proportional speed increases obtainable.

The minimal number of peripherals used combined with the inclusion of diagnostics and error correction emphasize its low cost capabilities. The results serve to validate the feasibility of a cost effective microcontroller based Automobile Instrument Cluster.

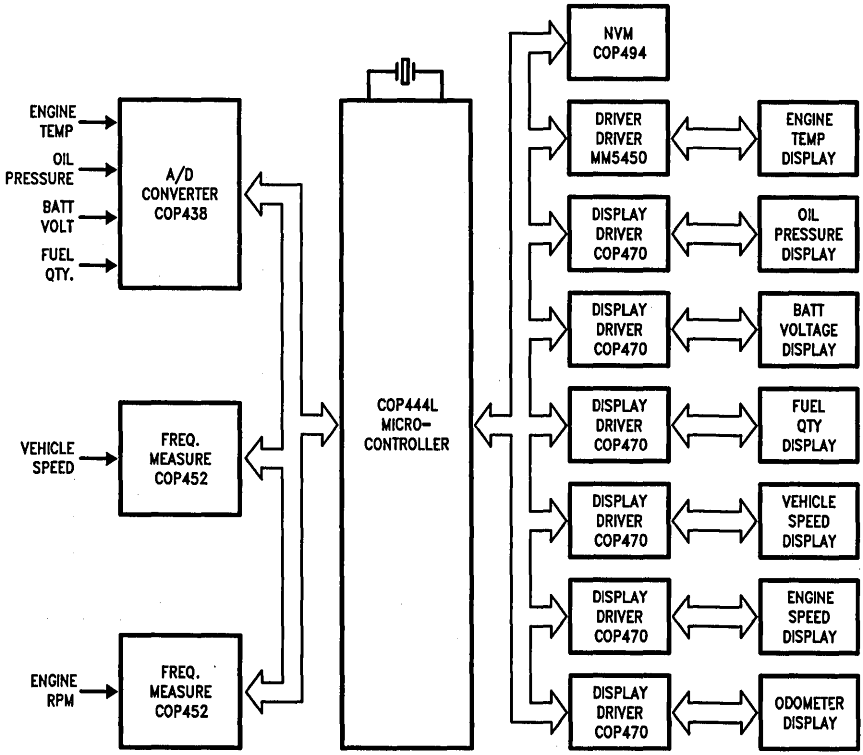
TABLE I. Comparison of Speed and Resolution of Measurements Taken with the COP444L and the COP440

	Measurements with a COP444L		Measurements with a COP440	
	Time Taken µsecs	Resolution Bits	Time Taken µsecs	Resolution Bits
1. Engine rpm	768	17	192	17
2. Vehicle Speed	768	17	192	17
3. Engine Temperature	256	8	64	8
4. Oil Pressure	256	8	64	8
5. Battery Voltage	256	8	64	8
6. Fuel Quantity	256	8	64	8



TL/DD/8798-1

FIGURE 1. Functional Block Diagrams of the AIC



TL/DD/8798-2

FIGURE 2

Automotive Multiplex Wiring

National Semiconductor
Application Note 454
Abdul H. Aleaf

INTRODUCTION

The evolutionary development of vehicle electronic systems has rapidly increased the number of individual wires in the vehicle. The conventional wiring harness will not provide solutions to the problems such as reducing size and weight in addition to meeting cost and reliability objectives. Several approaches have been taken to provide long term solutions. None has succeeded. Miniaturization of cables and wires is one example of a temporary solution.

Multiplexing on the other hand has been regarded as a technique which allows considerable savings to be made in the size and cost of the harness. It can also enhance reliability by reducing the number of electrical connections.

In a multiplex system the control functions will be distributed around the vehicle and complex interconnections between diagnostic terminals, sensors, instruments and switches will not add to the harness complexity. With all its advantages it has not been implemented on a production car yet. The reason has been economical feasibility and lack of suitable semiconductor components for power switching. But, with the rapid technology advances in power FETs and introduction of low cost microcomputers, multiplex wiring can be regarded as a logical successor to conventional wiring systems. Extended development efforts are necessary to introduce a reliable system at reasonable cost.

The Microcontroller Applications Group at National Semiconductor has taken a step towards this goal. A low end multiplex wiring system focusing on asynchronous serial communication in a multi node network has been developed. This paper describes the development of this system on an abstract model which forms the basis for analysis of communication protocol and various node functions.

SYSTEM CONFIGURATION

Figure 1 presents a general view of the system. The system is a centralized single master multiple slave-node scheme. All units are connected together by a balanced twisted pair. The expandable interconnection of different subsystems is achieved with 9600 Baud communication over a standard UART bus. The bus handles the interface between a master controller and the intelligent nodes.

The approach to have a centralized control system offers several advantages as compared against a non-centralized system. It prevents the problem of bus monopolization by a faulty node and is potentially cheaper due to the need for only one complex node (master). The master-slave architecture also prevents bus contention problems.

The master is a COP420L. The COP420L is a 4-bit microcontroller with a software UART that handles asynchronous communication with other processors at speeds up to 9600 Baud.

The use of 4-bit 49¢ microcontrollers (COP413L) at the nodes not only provides intelligence which reduces the required bus bandwidth, it also reduces the incremental cost associated with automotive multiplexing. All standard nodes are identical. One standard program is used. This uniformity contributes to the system flexibility and expandability. External standard nodes may be added to the system to control additional functions. Node types and addresses are selected via external wire jumpers or switches. The slave nodes consist of four remote units to handle functions such as headlamps, tail lamps, etc. These nodes are the front right, front left, rear right and rear left nodes. Incorporated into the system are also a keyboard node, a EIC node and a display node.

The keyboard node may call for a control action at any time. This node is being continuously monitored by the master controller which receives status and processes the command or information.

Overall system intelligence and flexibility is increased by dedicating a node to NS455 the Terminal Management Processor. This node takes the responsibility to display information on a 4″ flat CRT display.

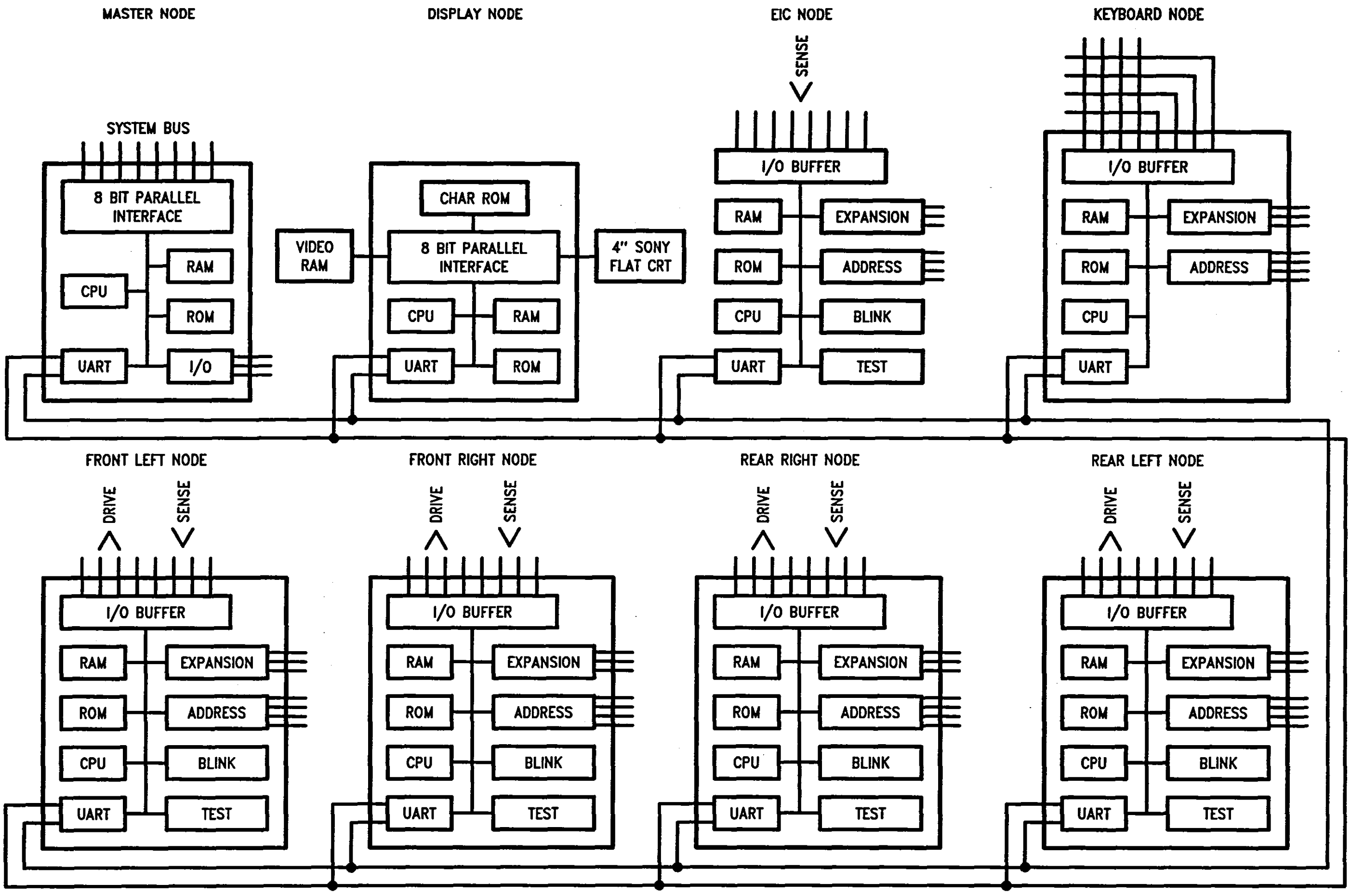
An Electronic Instrument Cluster (EIC) system is a completely independent system. It typically performs all functions associated with the automobile dashboard such as vehicle speed, odometers to accumulate mileage, gauges to display engine temperature, fuel level and so on. It also indicates error conditions such as high engine temperatures, low fuel level etc. The multiplex wiring system uses a standard slave node as a bridge between the two independent systems. The slave node monitors error conditions from the EIC system and passes them to the master node upon request. It becomes relatively simple to allow the master to access all activity in the EIC system via additional commands to the slave node serving the EIC system:

THE COMMUNICATION PROTOCOL

The master unit addresses the remote units sequentially and receives a status reply from each individual node. Data communication is via the standard UART format. It has a start bit, eight data bits, an even parity bit and one stop bit.

Information to be transmitted from the master to a slave node is organized as a frame. Each frame contains the address of destination and command or data. The information in a frame is transmitted as byte format. Address/data differentiation is done by means of a flag. The byte is an address byte if the MSB is set ("1"), otherwise it is a data byte.

Two different types of addressing schemes have been incorporated into the communication protocol; node addressing and class addressing. A class of nodes is formed by grouping together slave nodes with common functions. Commands may be executed either by specific individual nodes or by slave classes. All nodes of the same class execute the command simultaneously. The system implementation at National involved four classes with seven slave nodes per class. So, the total number of nodes possible in this system is 28.



TL/DD/8799-1

FIGURE 1. Block Diagram

The partitioning between the class address and node address reduces the density of bus traffic significantly by eliminating repetative command transmission to individual node class. Lower bus traffic implies that lower transmission bit rate can be used, allowing additional noise immunity. Another advantage of the class addressing is the provision of synchronization for control signals such as HAZARD, LEFT/RIGHT turns.

Error correction is incorporated into the communication protocol. The UART error flags such as PARITY and FRAMING ERRORS protect the system at the physical layer. At the system level, the nodes simply avoid sending an acknowledgement to the master when an error is detected. The master times out and sends the command again.

THE MASTER NODE

The master controller is the heart of the system. Its responsibility is to generate the controlling commands and synchronize the system. It transmits to the remote units and listens to them to get the vehicle status and acts accordingly. Circuit complexity is reduced by implementing extensive software programming in the master controller. This means that the burden is essentially on the master and must be engineered to very high standards of reliability. The device used in the implementation as the master is the COP1430. It is a cost effective 4-bit single chip microcontroller. It features on chip UART which handles asynchronous communications at speeds up to 9600 Baud.

THE SLAVE NODES

The standard slave nodes are based upon the COP413L. The COP413L is a low cost 4-bit microcontroller which may be customized in production. A system such as multiplex wiring requires power consumption to be absolutely minimal. Another basic requirement is that the system should be cost effective. These two facts directed us to use the COP413L at the standard slave node. The COP413L is a low cost (49¢!) low power microcontroller from NSC drawing less than 7 mA at 4.5V to 5.5V. The device contains an 8-bit bidirectional I/O port and a serial expansion port. The CMOS version of COP413L will also be available.

THE DISPLAY NODE

This node can serve as a condition monitoring unit for the vehicle. A considerable quantity of diagnostic information collected from transducers, switches, sensors and various loads are fed to this unit to be displayed on a CRT display. The node is based on a Terminal Management Processor the NS455. The NS455 is a CRT controller on chip. The messages are updated over the serial I/O line by the master controller. The communication format is:

a) The node receives the address.

b) If address matches the local node address, send the copy command

c) Receive new address and execute.

OUTPUT STAGES

The power FETs used for local switching throughout the system are IRF541[4]. These N-channel FETs provide much better drive circuit specification as compared to bipolar output stages. They also feature all of the well established advantages of MOSFET such as voltage control, very fast switching, and very low on state resistance. Another advantage is the lower cost as compared to comparibly rated p-channel devices.

TRANSMISSION MEDIUM

A balanced twisted pair is used for bus medium which provides high noise immunity. The transceiver selected for the bus is DS3695 *(Figure 2)*. This device is a high speed differential TRI-STATE® Bus/line transceiver designed to meet EIA standard for multipoint bus transmission. Bus contention or fault situations that cause excessive power dissipation within the device are handled by a standard thermal shutdown circuit, which forces the driver outputs into the high impedance state.

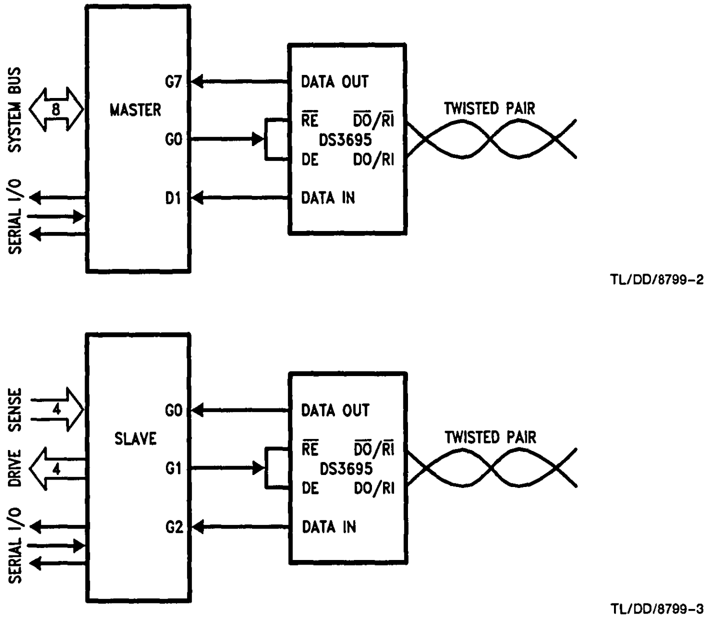


FIGURE 2. Bus Interface

CONCLUSIONS

Multiplex wiring system potentially seems to be a good replacement for conventional wiring system. Reduced complexity, increased flexibility and diagnostic capability could be achieved by incorporating microcontroller devices at nodes within the wiring system. The 4-bit microcontrollers selected are available in a price range, as low as 49¢, that will allow multiplex wiring to compare favorably on a cost-performance basis with the conventional harness.

REFERENCES

1. Michael W. Lowndes and Paul E. V. Phillips, "The Motorcar Multiplex Systems", IEE Conference on Automotive Electronics, 229, England, Nov. 1983.
2. R. F. Robins/W. J. Brittain/M. R. Lunt, "A Car Multiplex Wiring System with Self Coding Control Modules", IEE Conference on Automotive Electronics, 229, Ford Motor Company, UK, Nov. 1983.
3. Booth, J. A., 1983 "Vehicle Interconnection Systems for the Future", IEE Conference on Automotive Electronics, London, Nov. 1983.
4. International Rectifier, HEXFET Databook, 1985.

Dual Tone Multiple Frequency (DTMF)

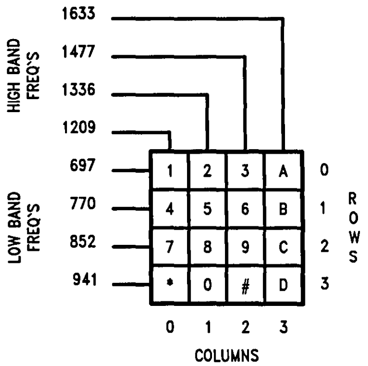
National Semiconductor
Application Note 521
Verne H. Wilson

The DTMF (Dual Tone Multiple Frequency) application is associated with digital telephony, and provides two selected output frequencies (one high band, one low band) for a duration of 100 ms. A benchmark subroutine has been written for the COP820C/840C microcontrollers, and is outlined in detail in this application note. This DTMF subroutine takes 110 bytes of COP820C/840C code, consisting of 78 bytes of program code and 32 bytes of ROM table. The timings in this DTMF subroutine are based on a 20 MHz COP820C/840C clock, giving an instruction cycle time of 1 µs.

The matrix for selecting the high and low band frequencies associated with each key is shown in *Figure 1*. Each key is uniquely referenced by selecting one of the four low band frequencies associated with the matrix rows, coupled with selecting one of the four high band frequencies associated with the matrix columns. The low band frequencies are 697, 770, 852, and 941 Hz, while the high band frequencies are 1209, 1336, 1477, and 1633 Hz. The DTMF subroutine assumes that the key decoding is supplied as a low order hex digit in the accumulator. The COP820C/840C DTMF subroutine will then generate the selected high band and low band frequencies on port G output pins G3 and G2 respectively for a duration of 100 ms.

The COP820C/840C each contain only one timer. The problem is that three different times must be generated to satisfy the DTMF application. These three times are the periods of the two selected frequencies and the 100 ms duration period. Obviously the single timer can be used to generate any one (or possibly two) of the required times, with the program having to generate the other two (or one) times.

The solution to the DTMF problem lies in dividing the 100 ms time duration by the half periods (rounded to the nearest micro second) for each of the eight frequencies, and then examining the respective high band and low band quotients and remainders. The results of these divisions are detailed in Table I. The low band frequency quotients range from 139 to 188, while the high band quotients range from 241 to 326. The observation that only the low band quotients will each fit in a single byte dictates that the high band frequency be produced by the 16 bit (2 byte) COP820C/840C timer running in PWM (Pulse Width Modulation) Mode.



TL/DD/9662-1

FIGURE 1. DTMF Keyboard Matrix

The solution then is to use the program to produce the selected low band frequency as well as keep track of the 100 ms duration. This is achieved by using three programmed register counters R0, R2, and R3, with a backup register R1 to reload the counter R0. These three counters represent the half period, the 100 ms quotient, and the 100 ms remainder associated with each of the four low band frequencies.

The theory of operation in producing the selected low band frequency starts with loading the three counters with values obtained from a ROM table. The half period for the selected frequency is counted out, after which the G2 output bit is toggled. During this half period countout, the quotient counter is decremented. This procedure is repeated until the quotient counter counts out, after which the program branches to the remainder loop. During the remainder loop, the remainder counter counts out to terminate the 100 ms. Following the remainder countout, the G2 and G3 bits are both reset, after which the DTMF subroutine is exited. Great care must be taken in time balancing the half period loop for the selected low band frequency. Furthermore, the toggling of the G2 output bit (achieved with either a set or reset bit instruction) must also be exactly time balanced to maintain the half period time integrity. Local stall loops (consisting of a DRSZ instruction followed by a JP jump back to the DRSZ for a two byte, six instruction cycle loop) are embedded in both the half period and remainder loops. Consequently, the ROM table parameters for the half period and remainder counters are approximately only one sixth of what otherwise might be expected. The program for the half period loop, along with the detailed time balancing of the loop for each of the low band frequencies, is shown in *Figure 2*.

The DTMF subroutine makes use of two 16 byte ROM tables. The first ROM table contains the translation table for the input hex digit into the core vector. The encoding of the hex digit along with the hex digit ROM translation table is shown in Table II. The row and column bits (RR, CC) representing the low band and high band frequencies respectively of the keyboard matrix shown in *Figure 1*, are encoded in

TABLE I. Frequency Half Periods, Quotients, and Remainders

	Freq. Hz	Half Period 0.5P	Half Period in µs	100 ms/0.5P	
				Quotient	Remainder
Low Band Freq.'s	697	717.36	717	139	337
	770	649.35	649	154	54
	852	586.85	587	170	210
	941	531.35	531	188	172
High Band Freq.'s	1209	413.56	414 (256 + 158)	241	226
	1336	374.25	374 (256 + 118)	267	142
	1477	338.52	339 (256 + 83)	294	334
	1633	306.18	306 (256 + 50)	326	244

the two upper and two lower bits of the hex digit respectively. Consequently, the format for the hex digit bits is RRCC, so that the input byte in the accumulator will consist of 0000RRCC. The program changes this value into 1101RRCC before using it in setting up the address for the hex digit ROM translation table.

The core vectors from the hex digit ROM translation table consist of a format of XX00TT00, where the two T (Timer) bits select one of four high band frequencies, while the two X bits select one of four low band frequencies. The core vector is transformed into four different inputs for the second ROM table. This transformation of the core vector is shown in Table III. The core vector transformation produces a timer vector 1100TT00 (T), and three programmed counter vectors for R1, R2, and R3. The formats for the three counter vectors are 1100XX11 (F), 1100XX10 (Q), and 1100XX01 (R) for R1, R2, and R3 respectively. These four vectors produced from the core vector are then used as inputs to the second ROM table. One of these four vectors (the T vector) is a function of the T bits from the core vector, while the other three vectors (F, Q, R) are a function of the X bits. This correlates to only one parameter being needed for the timer (representing the selected high band frequency), while three parameters are needed for the three counters (half period, 100 ms quotient, 100 ms remainder) associated with the low band frequency and 100 ms duration. The frequency parameter ROM translation table, accessed by the T, F, Q, and R vectors, is shown in Table IV.

Program			Bytes/Cycle	Conditional Cycles		Cycles	Total Cycles
	LD	B,#PORTGD	2/3				
	LD	X,#R1	2/3				
LUP1:	LD	A,[X−]	1/3			3	
	IFBIT	2,[B]	1/1			1	
	JP	BYP1	1/3	3	1		
	X	A,[X+]	1/3		3		
	SBIT	2,[B]	1/1		1		
	JP	BYP2	1/3		3		
BYP1:	NOP		1/1	1			
	RBIT	2,[B]	1/1	1			
	X	A,[X+]	1/3	3			
BYP2:	DRSZ	R2	1/3 DECREMENT			3	
	JP	LUP2	1/3 Q COUNT			3	
	JP	FINI	1/3				
LUP2:	DRSZ	R0	1/3 DECREMENT		3	3	
	JP	LUP2	1/3 F COUNT		3	1	
	NOP		1/1			1	
	LD	A,[X]	1/3			3	
	IFEQ	A,#104	2/2			2	
	JP	LUP1	1/3		1	3	31
	NOP		1/1		1		
	IFEQ	A,#93	2/2		2		
BACK:	JP	LUP1	1/3	1	3		35
	JP	BACK	1/3	3			
				3			39

Table IV Frequency	×	Stall Loop	+	Total Cycles	=	Half Period
((114 − 1)		x 6)		+ 39		= 717
((104 − 1)		x 6)		+ 31		= 649
((93 − 1)		x 6)		+ 35		= 587
((83 − 1)		x 6)		+ 39		= 531

FIGURE 2. Time Balancing for Half Period Loop

TABLE II. Hex Digit ROM Translation Table

	0	1	2	3
ROW	697 Hz	770 Hz	852 Hz	941 Hz
COLUMN	1209 Hz	1336 Hz	1477 Hz	1633 Hz

ADDRESS	DATA (HEX)	KEYBOARD
*		
0xD0	000	1
0xD1	004	2
0xD2	008	3
0xD3	00C	A
0xD4	040	4
0xD5	044	5
0xD6	048	6
0xD7	04C	B
0xD8	080	7
0xD9	084	8
0xDA	088	9
0xDB	08C	C
0xDC	0C0	*
0xDD	0C4	0
0xDE	0C8	#
0xDF	0CC	D

```
* HEX DIGIT IS RRCC,
    WHERE R = ROW #
      AND C = COLUMN #
3 - - - EXAMPLE: KEY 3 IS ROW #0,
    COLUMN #2, SO HEX DIGIT
    IS 0010 = 2
       RRCC
```

TABLE III. Core Vector Translation

```
CORE VECTOR  -  XX00TT00 - - - - - - - - - -
                                           *
                                          * *
                                         * * *
```

TIMER VECTOR	TIMER	T	1100TT00
HALF PERIOD VECTOR	R1	F	1100XX11
QUOTIENT VECTOR	R2	Q	1100XX10
REMAINDER VECTOR	R3	R	1100XX01

TABLE IV. Frequency Parameter ROM Translation Table

T - TIMER F - FREQUENCY Q - QUOTIENT R - REMAINDER

ADDRESS	DATA (DEC)	VECTOR
0xC0	158	T
0xC1	53	R
0xC2	140	Q
0xC3	114	F
0xC4	118	T
0xC5	6	R
0xC6	155	Q
0xC7	104	F
0xC8	83	T
0xC9	32	R
0xCA	171	Q
0xCB	93	F
0xCC	50	T
0xCD	25	R
0xCE	189	Q
0xCF	83	F

In summary, the input hex digit selects one of 16 core vectors from the first ROM table. This core vector is then transformed into four other vectors (T, F, Q, R), which in turn are used to select four parameters from the second ROM table. These four parameters are used to load the timer, and the respective half period, quotient, and remainder counters. The first ROM table (representing the hex digit matrix table) is arbitrarily placed starting at ROM location 01D0, and has a reference setup with the ADD A,#0D0 instruction. The second ROM table (representing the frequency parameter table) must be placed starting at ROM location 01C0 (or 0xC0) in order to minimize program size, and has reference setups with the OR A,#0C3 instruction for the F vector and with the OR A,#0C0 instruction for the T vector.

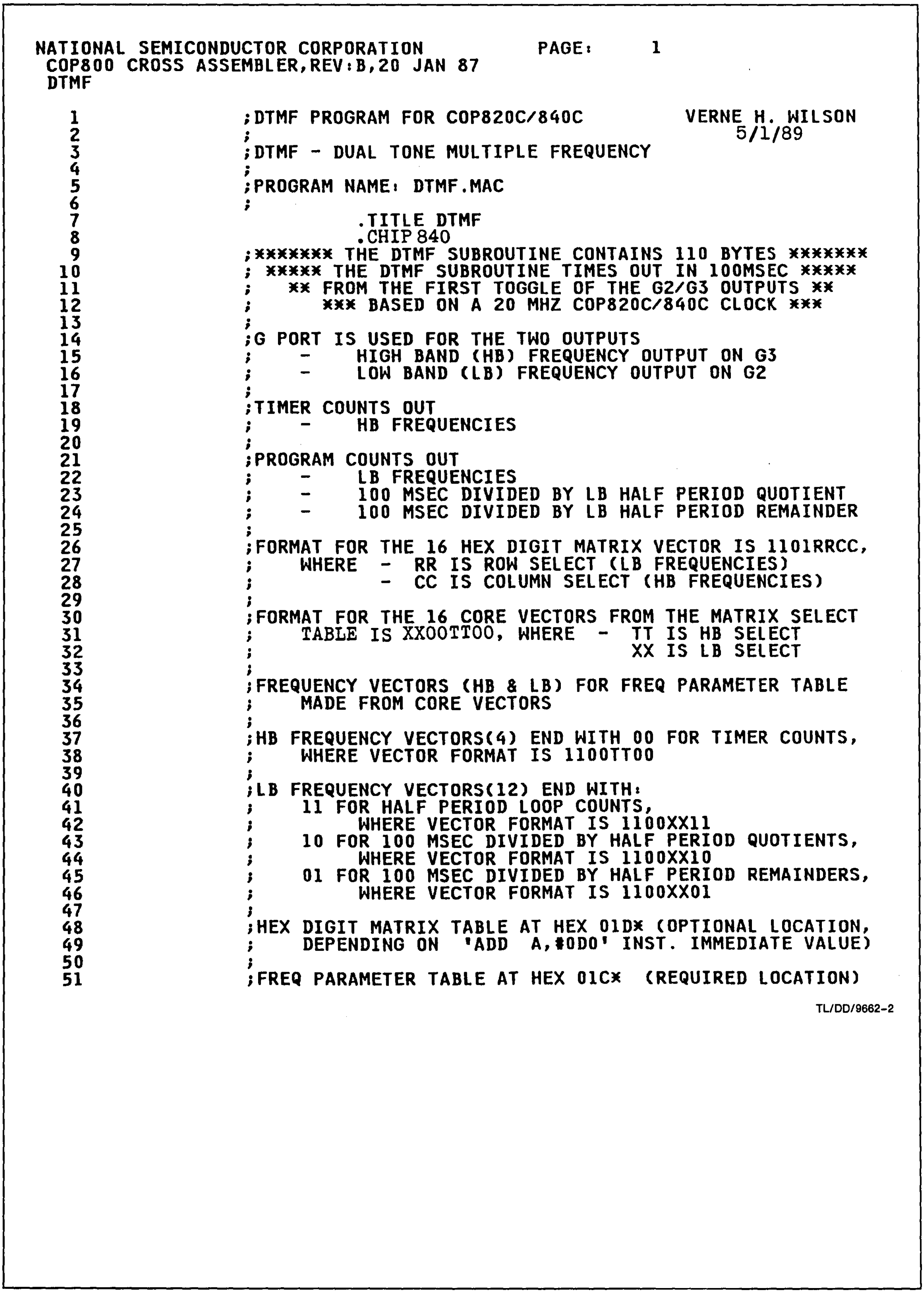
The three parameters associated with the two X bits of the core vector require a multi-level table lookup capability with the LAID instruction. This is achieved with the following section of code in the DTMF subroutine:

```
        LD      B,#R1
        LD      X,#R4
        X       A,[X]
LUP:    LD      A,[X]
        LAID
        X       A,[B+]
        DRSZ    R4
        IFBNE   #4
        JP      LUP
```

This program code loads the F frequency vector into R4, and then decrements the vector each time around the loop. This successive loop decrementation of the R4 vector changes the F vector into the Q vector, and then changes the Q vector into the R vector. This R4 vector is used to access the ROM table with the LAID instruction. The X pointer references the R4 vector, while the B pointer is incremented each time around the loop after it has been used to store away the three selected ROM table parameters (one per loop). These three parameters are stored in sequential RAM locations R1, R2, and R3. The IFBNE test instruction is used to skip out of the loop once the three selected ROM table parameters have been accessed and stored away.

The timer is initialized to a count of 15 so that the first timer underflow and toggling of the G3 output bit (with timer PWM mode and G3 toggle output selected) will occur at the same time as the first toggling of the G2 output bit. The half period counts for the high band frequencies range from 306 to 414, so these values minus 256 are stored in the timer section of the second ROM table. The selected value from this frequency ROM table is then stored in the lower half of the timer autoreload register, while a 1 is stored in the upper half. The timer is selected for PWM output mode and started with the instruction LD [B],#0B0 where the B pointer is selecting the CNTRL register at memory location 0EE.

The DTMF subroutine for the COP820C/840C uses 110 bytes of code, consisting of 78 bytes of program code and 32 bytes of ROM table. A program routine to sequentially call the DTMF subroutine for each of the 16 hex digit inputs is supplied with the listing for the DTMF subroutine.



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:      1
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF

                    ;DTMF PROGRAM FOR COP820C/840C          VERNE H. WILSON
                    ;                                              5/1/89
                    ;DTMF - DUAL TONE MULTIPLE FREQUENCY
                    ;
                    ;PROGRAM NAME: DTMF.MAC
                    ;
                            .TITLE DTMF
                            .CHIP 840
                    ;******* THE DTMF SUBROUTINE CONTAINS 110 BYTES *******
                    ; ***** THE DTMF SUBROUTINE TIMES OUT IN 100MSEC *****
                    ;   ** FROM THE FIRST TOGGLE OF THE G2/G3 OUTPUTS **
                    ;      *** BASED ON A 20 MHZ COP820C/840C CLOCK ***
                    ;
                    ;G PORT IS USED FOR THE TWO OUTPUTS
                    ;    -    HIGH BAND (HB) FREQUENCY OUTPUT ON G3
                    ;    -    LOW BAND (LB) FREQUENCY OUTPUT ON G2
                    ;
                    ;TIMER COUNTS OUT
                    ;    -    HB FREQUENCIES
                    ;
                    ;PROGRAM COUNTS OUT
                    ;    -    LB FREQUENCIES
                    ;    -    100 MSEC DIVIDED BY LB HALF PERIOD QUOTIENT
                    ;    -    100 MSEC DIVIDED BY LB HALF PERIOD REMAINDER
                    ;
                    ;FORMAT FOR THE 16 HEX DIGIT MATRIX VECTOR IS 1101RRCC,
                    ;    WHERE  -  RR IS ROW SELECT (LB FREQUENCIES)
                    ;              -  CC IS COLUMN SELECT (HB FREQUENCIES)
                    ;
                    ;FORMAT FOR THE 16 CORE VECTORS FROM THE MATRIX SELECT
                    ;    TABLE IS XX00TT00, WHERE  -  TT IS HB SELECT
                    ;                                 XX IS LB SELECT
                    ;
                    ;FREQUENCY VECTORS (HB & LB) FOR FREQ PARAMETER TABLE
                    ;    MADE FROM CORE VECTORS
                    ;
                    ;HB FREQUENCY VECTORS(4) END WITH 00 FOR TIMER COUNTS,
                    ;    WHERE VECTOR FORMAT IS 1100TT00
                    ;
                    ;LB FREQUENCY VECTORS(12) END WITH:
                    ;    11 FOR HALF PERIOD LOOP COUNTS,
                    ;          WHERE VECTOR FORMAT IS 1100XX11
                    ;    10 FOR 100 MSEC DIVIDED BY HALF PERIOD QUOTIENTS,
                    ;          WHERE VECTOR FORMAT IS 1100XX10
                    ;    01 FOR 100 MSEC DIVIDED BY HALF PERIOD REMAINDERS,
                    ;          WHERE VECTOR FORMAT IS 1100XX01
                    ;
                    ;HEX DIGIT MATRIX TABLE AT HEX 01D* (OPTIONAL LOCATION,
                    ;    DEPENDING ON  'ADD  A,#0D0' INST. IMMEDIATE VALUE)
                    ;
                    ;FREQ PARAMETER TABLE AT HEX 01C*  (REQUIRED LOCATION)
```

TL/DD/9662-2

```
NATIONAL SEMICONDUCTOR CORPORATION              PAGE:     2
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF

                                 .FORM
                    ;
                    ;MAGIC:         CORE VECTOR
                    ;                 XX00TT00
                    ;
                    ;    TIMER      T     TT00
                    ;    R1         F     XX11
                    ;    R2         Q     XX10
                    ;    R3         R     XX01
                    ;
                    ;DECLARATIONS:
      00D0             PORTLD = 0D0            ; PORTL DATA REG
      00D1             PORTLC = 0D1            ; PORTL CONFIG REG
      00D4             PORTGD = 0D4            ; PORTG DATA REG
      00D5             PORTGC = 0D5            ; PORTG CONFIG REG
      00DC              PORTD = 0DC            ; PORTD REG
      00EA            TIMERLO = 0EA            ; TIMER LOW COUNTER
      00EE              CNTRL = 0EE            ; CONTROL REG
      00EF                PSW = 0EF            ; PROC STATUS WORD
      00F0                 R0 = 0F0            ; LB FREQ LOOP COUNTER
      00F1                 R1 = 0F1            ; LB FREQ LOOP COUNT
      00F2                 R2 = 0F2            ; LB FREQ Q COUNT
      00F3                 R3 = 0F3            ; LB FREQ R COUNT
      00F4                 R4 = 0F4            ; LB FREQ TABLE VECTOR
                    ;
 0000 DD2F          START:     LD        SP,#02F          ; HEX DIGIT MATRIX
 0002 BCD1FF                   LD        PORTLC,#0FF      ; 1  2  3  A
 0005 BCD080                   LD        PORTLD,#080      ; 4  5  6  B
 0008 DEDC                     LD        B,#PORTD         ; 7  8  9  C
 000A 9E00                     LD        [B],#0           ; *  0  #  D
 000C AE            LOOP:      LD        A,[B]            ; DTMF TEST LOOP
 000D 3160                     JSR       DTMF             ; HEX MATRIX DIGIT
 000F DEDC                     LD        B,#PORTD         ; TO SUBROUTINE IS
 0011 AE                       LD        A,[B]            ; OUTPUT TO PORTD
 0012 9405                     ADD       A,#5             ; D0 WILL TOGGLE
 0014 A6                       X         A,[B]            ; FOR EACH CALL OF
 0015 6C                       RBIT      4,[B]            ; DTMF SUBROUTINE
 0016 9DD0                     LD        A,PORTLD         ; PORTL OUTPUTS
 0018 A1                       SC                         ; PROVIDE SYNC
 0019 B0                       RRC       A                ; OUTPUT ORDER IS
 001A 9CD0                     X         A,PORTLD         ; 1,5,9,D,4,8,#,A,
 001C EF                       JP        LOOP             ; 7,0,3,B,*,2,6,C
                    ;
                    ;
```

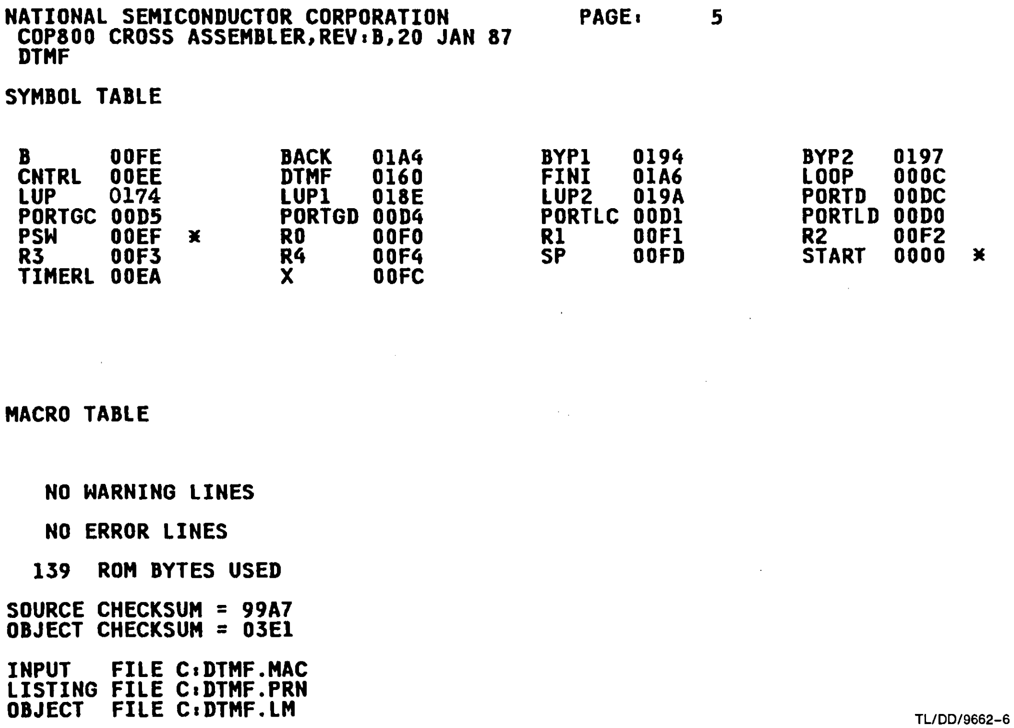
TL/DD/9662-3

```
     0160                  .=0160
                  ;
0160 DED5         DTMF:    LD      B,#PORTGC
0162 9B3F                  LD      [B-],#03F
0164 6B                    RBIT    3,[B]          ; OPTIONAL
0165 6A                    RBIT    2,[B]          ; OPTIONAL
                  ;
0166 94D0                  ADD     A,#0D0
0168 A4                    LAID                   ; DIGIT MATRIX TABLE
                  ;
0169 5F                    LD      B,#0
016A A6                    X       A,[B]
016B AE                    LD      A,[B]
017B 65                    SWAP    A
016C 97C3                  OR      A,#0C3
016E DEF1                  LD      B,#R1
0170 DCF4                  LD      X,#R4
0172 B6                    X       A,[X]
0173 BE           LUP:     LD      A,[X]
0174 A4                    LAID                   ; LB FREQ TABLES
0175 A2                    X       A,[B+]         ;   (3 PARAMETERS)
0176 C4                    DRSZ    R4
0177 44                    IFBNE   #4
0178 FA                    JP      LUP
                  ;
0179 5F                    LD      B,#0
017A AE                    LD      A,[B]
017C 97C0                  OR      A,#0C0
017E A4                    LAID                   ; HB FREQ TABLE
017F DEEA                  LD      B,#TIMERLO     ;   (1 PARAMETER)
0181 9A0F                  LD      [B+],#15
0183 9A00                  LD      [B+],#0
0185 A2                    X       A,[B+]
0186 9A01                  LD      [B+],#1
0188 9EB0                  LD      [B],#0B0       ; START TIMER PWM
                  ;
018A DED4                  LD      B,#PORTGD
018C DCF1                  LD      X,#R1
                  ;
018E BB           LUP1:    LD      A,[X-]
018F 72                    IFBIT   2,[B]          ; TEST LB OUTPUT
0190 03                    JP      BYP1
0191 B2                    X       A,[X+]
0192 7A                    SBIT    2,[B]          ; SET LB OUTPUT
0193 03                    JP      BYP2
0194 B8           BYP1:    NOP
0195 6A                    RBIT    2,[B]          ; RESET LB OUTPUT
0196 B2                    X       A,[X+]
0197 C2           BYP2:    DRSZ    R2             ; DECR. QUOT. COUNT
0198 01                    JP      LUP2
0199 0C                    JP      FINI           ; Q COUNT FINISHED
                  ;
019A C0           LUP2:    DRSZ    R0             ; DECR. F COUNT
019B FE                    JP      LUP2           ; LB (HALF PERIOD)
                  ;
019C B8                    NOP                    ; ************
019D BE                    LD      A,[X]          ; BALANCE
019E 9268                  IFEQ    A,#104         ; LB FREQUENCY
01A0 ED                    JP      LUP1           ; HALF PERIOD
                  ;                               ; RESIDUE
01A1 B8                    NOP                    ; DELAY FOR
01A2 925D                  IFEQ    A,#93          ; EACH OF 4
01A4 E9           BACK:    JP      LUP1           ; LB FREQ'S
01A5 FE                    JP      BACK           ; ************
                  ;
01A6 C3           FINI:    DRSZ    R3             ; DECR. REM. COUNT
01A7 FE                    JP      FINI           ; R CNT NOT FINISHED
                  ;
01A8 BDEE6C                RBIT    4,CNTRL        ; STOP TIMER
01AB 6B                    RBIT    3,[B]          ; CLR HB OUTPUT
01AC 6A                    RBIT    2,[B]          ; CLR LB OUTPUT
                  ;
01AD 8E                    RET
                  ;
```

TL/DD/9662-4

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     4
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF


                                       .FORM
                        ;
                        ;FREQUENCY AND 100MSEC PARAMETER TABLE
      01C0                        .=01C0
                        ;
01C0 9E                           .BYTE     158           ; T
01C1 35                           .BYTE      53           ; R
01C2 8C                           .BYTE     140           ; Q
01C3 72                           .BYTE     114           ; F
01C4 76                           .BYTE     118           ; T
01C5 06                           .BYTE       6           ; R
01C6 9B                           .BYTE     155           ; Q
01C7 68                           .BYTE     104           ; F
01C8 53                           .BYTE      83           ; T
01C9 20                           .BYTE      32           ; R
01CA AB                           .BYTE     171           ; Q
01CB 5D                           .BYTE      93           ; F
01CC 32                           .BYTE      50           ; T
01CD 19                           .BYTE      25           ; R
01CE BD                           .BYTE     189           ; Q
01CF 53                           .BYTE      83           ; F
                        ;
                        ;DIGIT MATRIX TABLE
      01D0                        .=01D0
                        ;                                        ROW  COL
01D0 00                           .BYTE     000           ; 1      0     0
01D1 04                           .BYTE     004           ; 2      0     1
01D2 08                           .BYTE     008           ; 3      0     2
01D3 0C                           .BYTE     00C           ; A      0     3
01D4 40                           .BYTE     040           ; 4      1     0
01D5 44                           .BYTE     044           ; 5      1     1
01D6 48                           .BYTE     048           ; 6      1     2
01D7 4C                           .BYTE     04C           ; B      1     3
01D8 80                           .BYTE     080           ; 7      2     0
01D9 84                           .BYTE     084           ; 8      2     1
01DA 88                           .BYTE     088           ; 9      2     2
01DB 8C                           .BYTE     08C           ; C      2     3
01DC C0                           .BYTE     0C0           ; *      3     0
01DD C4                           .BYTE     0C4           ; 0      3     1
01DE C8                           .BYTE     0C8           ; #      3     2
01DF CC                           .BYTE     0CC           ; D      3     3
                        ;
                                  .END
```

TL/DD/9662-5



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
 COP800 CROSS ASSEMBLER,REV:B,20 JAN 87
 DTMF

SYMBOL TABLE

 B       00FE        BACK    01A4        BYP1    0194        BYP2    0197
 CNTRL   00EE        DTMF    0160        FINI    01A6        LOOP    000C
 LUP     0174        LUP1    018E        LUP2    019A        PORTD   00DC
 PORTGC  00D5        PORTGD  00D4        PORTLC  00D1        PORTLD  00D0
 PSW     00EF   *    R0      00F0        R1      00F1        R2      00F2
 R3      00F3        R4      00F4        SP      00FD        START   0000   *
 TIMERL  00EA        X       00FC



MACRO TABLE


   NO WARNING LINES

   NO ERROR LINES

  139  ROM BYTES USED

SOURCE CHECKSUM = 99A7
OBJECT CHECKSUM = 03E1

INPUT   FILE C:DTMF.MAC
LISTING FILE C:DTMF.PRN
OBJECT  FILE C:DTMF.LM
```

TL/DD/9662-6

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

Kaish Circuit Lockout System

National Semiconductor
Application Note 560
Jerry Leventer
Norman Kaish

I. GENERAL DESCRIPTION

The Kaish Circuit Lockout System (KCL) is a security system to be used in products that employ National Semiconductor 8-bit and 16-bit microcontrollers.

The Kaish Circuit Lockout System offers a high level of security against theft by preventing equipment from operating after it has been tampered with, removed from its installed location, or disconnected from its power source. An optional operating mode utilizes a countdown timer to securely disable the product after a predetermined time period. When the product is re-installed or reconnected to the power source, it remains inoperable until it receives a personal identification (PIN) code that has been previously selected by the owner. After a given number of unsuccessful PIN code entries, the system will enter a Lockout mode which prevents any further access to the product.

An optional encryption reactivation mode allows the manufacturer to remotely re-activate the product.

II. FUNCTIONAL DESCRIPTION

The Kaish Circuit Lockout features are achieved by modifying the software which control the normal functions of the equipment to be protected. It is compatible with National's single chip microcontrollers.

Two software flowcharts are shown in *Figures 1* and *2*. *Figure 1* shows an EEPROM implementation of PIN code storage whereas *Figure 2* shows a battery backup RAM implementation.

A. EEPROM Implementation

The software flowchart of *Figure 1* shows a typical routine for implementing the KCL system in a microcontroller based application using an internal EEPROM or an external EEPROM potted with a microcontroller to form a module to prevent access to the address and data lines. The COP494 (NMC9306) peripheral device which provides 256 bits of external EEPROM can be used in conjunction with National's 8-bit and 16-bit microcontrollers. On the other hand, the COP8640C microcontroller which is currently in design, provides 2 kbytes of ROM, 64 bytes of RAM and 64 bytes of EEPROM in addition to all the standard features of the COP840. This part will provide the best possible security.

Assume that primary power has been removed and then reapplied to the circuit. After hardware and software initialization, the initialization flag word (IFW) is read from non-volatile memory. The IFW is used to indicate 1) whether the product has undergone initial power-up from the factory, and 2) whether the Service/Test mode has been selected. If the IFW matches a pre-determined bit pattern stored in program ROM then the IFW is valid. This condition indicates that this is not the first time the product has been powered up by the user and a valid PIN code has been previously installed. Upon invalid PIN code entry, an error count will be incremented and tested for multiple guesses. Lockout mode will be entered when the count exceeds a given maximum such as five. Upon receiving a valid PIN code, the error count is reset and the user will be prompted to enter a new PIN code if desired.

A1. Lockout Mode

After the user makes five incorrect PIN code tries, the Lockout mode will be entered. For the highest security, the product can be permanently disabled. At the manufacturer's option, the product can be made to resume normal operation by using an encryption algorithm stored in the microcontroller's program memory. The microcontroller will generate a random number and encrypt it. The encrypted number generated by algorithm is transparent to the user. The unencrypted number is displayed and communicated by phone to the manufacturer. Upon proof of ownership, an encrypted number will be given to the user to be entered into the keyboard of the product unit. If the internally generated number matches the entered number, the user will be prompted to assign a new PIN code to the unit and normal operation will resume.

A2. Initial Operation

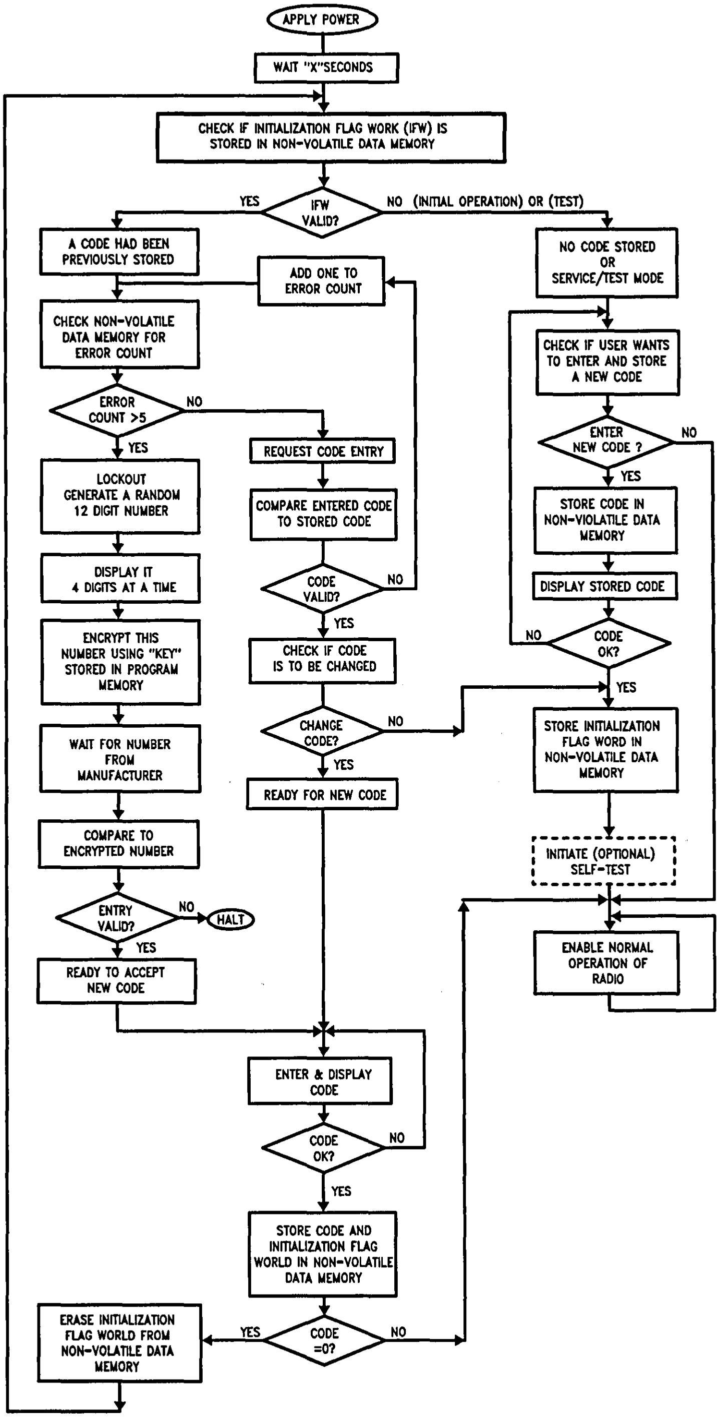
The operation of the program so far has assumed that the Initialization Flag Word (IFW) was correctly stored in nonvolatile memory. This will be true in all but two cases. The first exception is during final assembly when the unit will be powered-up for the first time. The chance for a random match will be very small and only when final assembly is done will the IFW be stored. (The programming of the IFW is done internally by the microcontroller.) Since a PIN code has never been stored, the manufacturer can store a PIN which can later be changed by the user or the user can store the PIN initially. In either case, when a PIN code is finally entered, the microcontroller will store the IFW in nonvolatile memory. This will signify the end of the initial operation of the unit. Subsequent executions of program memory will detect the IFW and require correct entry of the assigned PIN code.

The second exception where the IFW will not be valid is when it has been deliberately erased by the microcontroller program. This occurs when the Service/Test mode is entered. This mode is described below.

A3. Service/Test Mode

The Service/Test mode allows production line testing to proceed without interference from the Kaish Lockout System. The same is true when a unit is brought in to a service center for repair. This mode allows normal operation without the burden of having to repeatedly enter the PIN code.

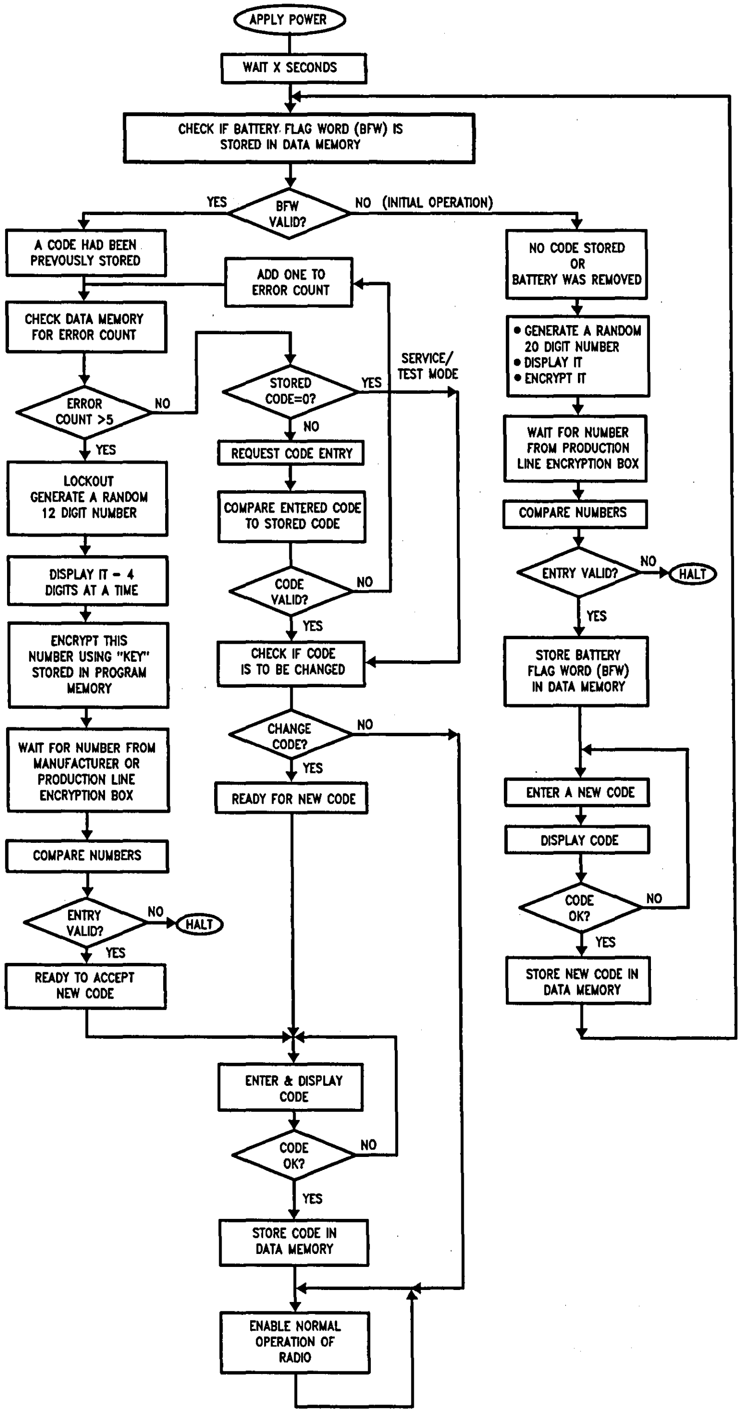
To enter this mode the user must first enter the correct PIN code. Then, when prompted for a new PIN code, the user enters a zero. The IFW is erased by the microcontroller at this point and the microcontroller will assume that this is an initial execution of the program with no PIN code stored. Each request for new PIN code entry on power-up can be rejected until the service or test procedure is complete. This will save time and remove the burden of keeping track of a PIN code during service and test procedures.


APPLY POWER
WAIT "X"SECONDS
CHECK IF INITIALIZATION FLAG WORK (IFW) IS STORED IN NON-VOLATILE DATA MEMORY
IFW VALID?
YES
NO (INITIAL OPERATION) OR (TEST)
A CODE HAD BEEN PREVIOUSLY STORED
ADD ONE TO ERROR COUNT
NO CODE STORED OR SERVICE/TEST MODE
CHECK NON-VOLATILE DATA MEMORY FOR ERROR COUNT
CHECK IF USER WANTS TO ENTER AND STORE A NEW CODE
ERROR COUNT >5
NO
YES
REQUEST CODE ENTRY
ENTER NEW CODE ?
NO
YES
LOCKOUT GENERATE A RANDOM 12 DIGIT NUMBER
COMPARE ENTERED CODE TO STORED CODE
STORE CODE IN NON-VIOLATILE DATA MEMORY
DISPLAY IT 4 DIGITS AT A TIME
CODE VALID?
NO
DISPLAY STORED CODE
YES
NO
CODE OK?
ENCRYPT THIS NUMBER USING "KEY" STORED IN PROGRAM MEMORY
CHECK IF CODE IS TO BE CHANGED
YES
CHANGE CODE?
NO
STORE INITIALIZATION FLAG WORD IN NON-VOLATILE DATA MEMORY
WAIT FOR NUMBER FROM MANUFACTURER
YES
READY FOR NEW CODE
COMPARE TO ENCRYPTED NUMBER
INITIATE (OPTIONAL) SELF-TEST
ENTRY VALID?
NO
HALT
YES
ENABLE NORMAL OPERATION OF RADIO
READY TO ACCEPT NEW CODE
ENTER & DISPLAY CODE
CODE OK?
NO
YES
STORE CODE AND INITIALIZATION FLAG WORLD IN NON-VOLATILE DATA MEMORY
ERASE INITIALIZATION FLAG WORLD FROM NON-VOLATILE DATA MEMORY
YES
CODE =0?
NO

TL/DD/10079-1

For microcontroller with external or internal non-volatile data memory (EEPROM)

FIGURE 1. Kaish Circuit Lockout System Software Flow Chart


APPLY POWER
WAIT X SECONDS
CHECK IF BATTERY FLAG WORD (BFW) IS STORED IN DATA MEMORY
YES
BFW VALID?
NO (INITIAL OPERATION)
A CODE HAD BEEN PREVOUSLY STORED
ADD ONE TO ERROR COUNT
NO CODE STORED OR BATTERY WAS REMOVED
CHECK DATA MEMORY FOR ERROR COUNT
• GENERATE A RANDOM 20 DIGIT NUMBER
• DISPLAY IT
• ENCRYPT IT
SERVICE/ TEST MODE
STORED CODE=0?
YES
ERROR COUNT >5
NO
NO
YES
REQUEST CODE ENTRY
WAIT FOR NUMBER FROM PRODUCTION LINE ENCRYPTION BOX
LOCKOUT GENERATE A RANDOM 12 DIGIT NUMBER
COMPARE ENTERED CODE TO STORED CODE
COMPARE NUMBERS
ENTRY VALID?
NO
HALT
DISPLAY IT - 4 DIGITS AT A TIME
CODE VALID?
NO
YES
YES
ENCRYPT THIS NUMBER USING "KEY" STORED IN PROGRAM MEMORY
CHECK IF CODE IS TO BE CHANGED
STORE BATTERY FLAG WORD (BFW) IN DATA MEMORY
CHANGE CODE?
NO
YES
WAIT FOR NUMBER FROM MANUFACTURER OR PRODUCTION LINE ENCRYPTION BOX
READY FOR NEW CODE
ENTER A NEW CODE
DISPLAY CODE
COMPARE NUMBERS
CODE OK?
NO
YES
ENTRY VALID?
NO
HALT
YES
READY TO ACCEPT NEW CODE
STORE NEW CODE IN DATA MEMORY
ENTER & DISPLAY CODE
CODE OK?
NO
YES
STORE CODE IN DATA MEMORY
ENABLE NORMAL OPERATION OF RADIO

TL/DD/10079-2

For microcontroller with internal battery backup RAM (separate V_{CC} supply for RAM and microcontroller)

FIGURE 2. Kaish Circuit Lockout System Software Flow Chart

B. Battery Backup System

Instead of using internal or external EEPROM, the CMOS microcontroller's RAM can be made non-volatile by using a long life or rechargeable battery. The COP800 family offers a minimum RAM retention voltage of 2.0V. *Figure 2* shows the flowchart for this implementation.

In this configuration there is no Initialization Flag Word. Instead a Battery Flag Word (BFW) is stored in RAM. The BFW is used to determine whether the battery has been removed for any period of time long enough to cause loss of data, or like the IFW, whether initial power-up has occurred.

If the BFW is not valid, it means that the PIN code is not valid. This occurs when the product is first shipped from the manufacturer and also when the battery backup has been removed. In either case the random number encryption algorithm must be used to re-activate the device.

The Service/Test mode is similar to that previously described in the section on EEPROM implementation except that there is no IFW. The user or service technician need only enter a zero when asked for a new PIN code. After each power-up upon checking that this code is zero, if no new code is desired then normal operation resumes without the need for a PIN code entry.

When using the product in the countdown timer lockout mode, a battery backup to the microcontroller is needed to maintain timekeeping functions.

III. HARDWARE

Figure 3 shows a block diagram of a typical car radio application using a National COP888C microcontroller, external EEPROM, display driver, and the Kaish circuit lockout system.

Reset

Power interrupt detect circuitry resets the microcontroller when power is disrupted. This occurs when the radio is taken out of the dash (but not when the radio is turned off). After this "theft event", it will be necessary for the user to input a proper PIN code before the radio will continue to function.

Tuner/Keyboard, Dual Function

Both the tuner keypad and display are used to input the security codes. The display will prompt the user with appropriate messages, "Enter PIN Code", "Incorrect, Try Again", and "New PIN Code Desired?." Also, the twelve digit encryption number that appears when in lockout will be displayed here. These dual functions are easily controlled in software.

Display Driver (COP472)

As shown in the system block diagram, the COP472 provides the segment drivers and control to drive the LCD display. The MICROWIRE/PLUS™ serial interface is used by the microcontroller to send the necessary information to the display driver. The D and I lines of the COP888C are used as keyboard strobes and inputs. The JID (jump indirect) and LAID (load accumulator indirect) instructions provide a fast and efficient means to decode any keyboard input.

External EEPROM (COP494/NMC9306)

The COP494 provides 256 bits of external EEPROM for use with microcontrollers that do not have internal non-volatile memory. G-lines are used as chip selects that allow the MICROWIRE/PLUS serial interface to be shared between the COP494 and the COP472.

Microcontrollers

National Semiconductor's COP888C microcontrollers, as shown in the block diagram of *Figure 3*, are well suited to this application. The COP888C family combines many advanced features onto a single chip. Features include low-power HALT and IDLE modes, MICROWIRE/PLUS serial communications, multiple multi-mode general purpose timers, multi-input wakeup/interrupt watchdog logic, clock monitor, and a full complement of maskable vectored interrupts.

The COP888CF contains an eight-channel, successive approximation A/D converter, while the COP888CG contains a full-duplex, double buffered UART and two differential comparators. Furthermore, an efficient instruction set using several addressing modes and many single-byte, single-cycle instructions provide minimal software overhead.

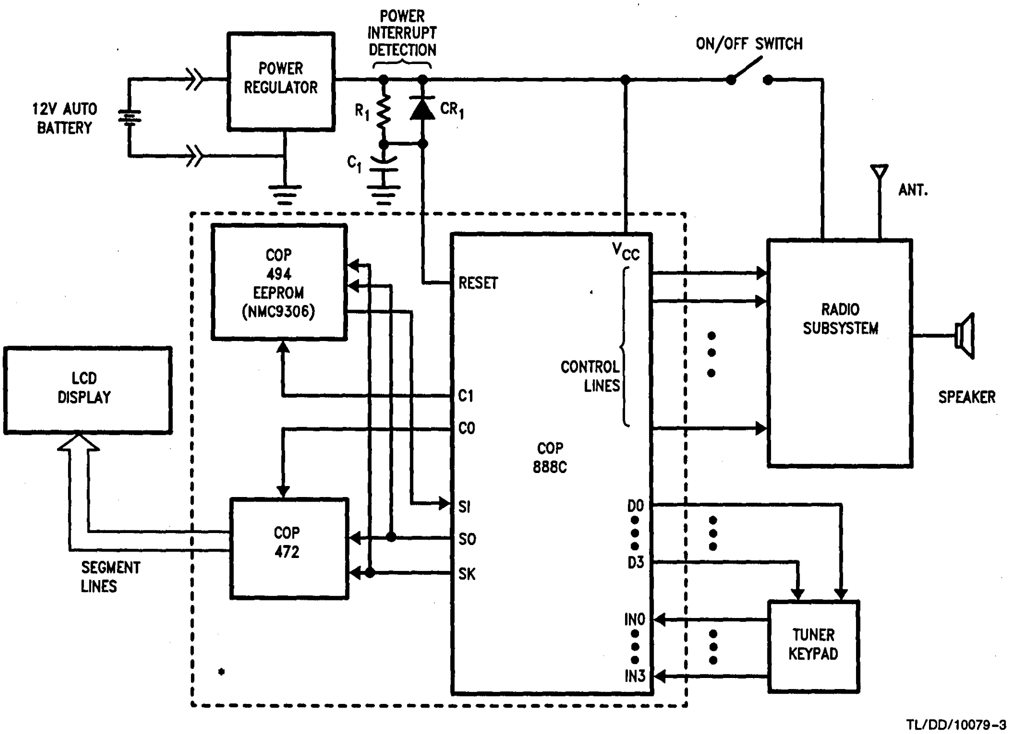
IV. SPECIAL CONSIDERATIONS

Due to the length of the encryption algorithm used in this security system, the size of the microcontroller program ROM must be greater than 1 kbyte. The encryption algorithm used in the prototype system made use of the National Bureau of Standards Data Encryption Standard. This algorithm required the use of approximately 1 kbyte of ROM. Shorter algorithms are available if needed. Nontheless, National Semiconductor's 2k and 4k devices (COP840C and COP888C) and 8k devices (HPC) provide the necessary ROM when using the battery backup design.

Memory Requirements

Category	Approx. Size (Bytes)	Notes
Security Logic	250	Required
Non-Volatile	16	Required
Keyboard Routine	100	Typical Overhead
Display Routine	275	Typical Overhead
DES Algorithm	500	Optional
Message Characters	50	Optional

Note: Specifications, drawings and operational prototypes of the KCL System were provided by International Electronic Technology Corp., 1931 Mott Avenue, Far Rockaway, NY 11691. Phone (718) 327-1119. Please contact IET for additional information on security algorithms, support services and licensing. U.S. Patent #4,494,114.



*Note: Components within the dotted line can be packaged as a Module, ASIC or Hybrid.

FIGURE 3. Typical Microcontroller-Tuned Radio System with Kaish Circuit Lockout Feature

COP800 MathPak

National Semiconductor
Application Note 596
Verne H. Wilson

OVERVIEW

This application note discusses the various arithmetic operations for National Semiconductor's COP800 family of 8-bit microcontrollers. These arithmetic operations include both binary and BCD (Binary Coded Decimal) operation. The four basic arithmetic operations (add, subtract, multiply, divide) are outlined in detail, with several examples shown for both binary and BCD addition and subtraction. Multiplication, division, and BCD conversion algorithms are also provided. Both BCD to binary and binary to BCD conversion subroutines are included, as well as the various multiplication and division subroutines.

Four sets of optimal subroutines are provided for

1. Multiplication
2. Division
3. Decimal (Packed BCD) to binary conversion
4. Binary to decimal (Packed BCD) conversion

One class of subroutines is optimized for minimal COP800 program code, while the second class is optimized for minimal execution time in order to optimize throughput time.

This application note is organized in four different sections. The first section outlines various addition and subtraction routines, including both binary and BCD (Binary Coded Decimal). The second section outlines the multiplication algorithm and provides several optimal multiply subroutines for 1, 2, 3, and 4 byte operation. The third section outlines the division algorithm and provides several optimal division subroutines for 1, 2, 3, and 4 byte operation. The fourth section outlines both the decimal (Packed BCD) to binary and binary to decimal (Packed BCD) conversion algorithms. This section provides several optimal subroutines for these BCD conversions.

The COP800 arithmetic instructions include the Add (ADD), Add with Carry (ADC), Subtract with Carry (SUBC), Increment (INCR), Decrement (DECR), Decimal Correct (DCOR), Clear Accumulator (ACC), Set Carry (SC), and Reset Carry (RC). The shift and rotate instructions, which include the Rotate Right through Carry (RRC) and the Swap Accumulator Nibbles (SWAP), may also be considered as arithmetic instruction variations. The RRC instruction is instrumental in writing a fast multiply routine.

1.0 BINARY AND BCD ADDITION AND SUBTRACTION

In subtraction, a borrow is represented by the absence of a carry and vice versa. Consequently, the carry flag needs to be set (no borrow) before a subtraction, just as the carry flag is reset before an addition. The ADD instruction does not use the carry flag as an input, nor does it change the carry flag. It should also be noted that both the carry and half carry flags (bits 6 and 7, respectively, of the PSW control register) are cleared with reset, and remain unchanged with the ADD, INC, DEC, DCOR, CLR and SWAP instructions. The DCOR instruction uses both the carry and half carry flags. The SC instruction sets both the carry and half carry flags, while the RC instruction resets both these flags.

The following program examples illustrate additions and subtractions of 4-byte data fields in both binary and BCD (Binary Coded Decimal). The four bytes from data memory locations 24 through 27 are added to or subtracted from the four bytes in data memory locations 16 through 19. The results replace the data in memory locations 24 through 27.

These operations are performed both in Binary and BCD. It should be noted that the BCD pre-conditioning of Adding (ADD) the hex 66 is only necessary with the BCD addition, not with the BCD subtraction. The (Binary Coded Decimal) DCOR (Decimal Correct) instruction uses both the carry and half carry flags as inputs, but does not change the carry and half carry flags. Also note that the #12 with the IFBNE instruction represents 28 − 16, since the IFBNE operand is modulo 16 (remainder when divided by 16).

BINARY ADDITION:

```
          LD        X,#16          ; NO LEADING ZERO
          LD        B,#24          ;    INDICATES DECIMAL
          RC                       ; RESET CARRY TO START
LOOP:     LD        A,[X+]         ; [X] TO ACC
          ADC       A,[B]          ; ADD [B] TO ACC
          X         A,[B+]         ; RESULT TO [B]
          IFBNE     #12            ; IF STILL IN DATA FIELD
          JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
          IFC                      ; IF TERMINAL CARRY,
          JP        OVFLOW         ;    JUMP TO OVERFLOW
```

BINARY SUBTRACTION:

```
          LD        X,#010         ; LEADING ZERO
          LD        B,#018         ;    INDICATES HEX
          SC                       ; RESET BORROW TO START
LOOP:     LD        A,[X+]         ; [X] TO ACC
          SUBC      A,[B]          ; SUBTRACT [B] FROM ACC
          X         A,[B+]         ; RESULT TO [B]
          IFBNE     #12            ; IF STILL IN DATA FIELD
          JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
          IFNC                     ; IF TERMINAL BORROW,
          JP        NEGRSLT        ;    JUMP TO NEGATIVE RESULT
```

BCD ADDITION:

```
          LD        X,#010         ; LEADING ZERO
          LD        B,#018         ;    INDICATES HEX
          RC                       ; RESET CARRY TO START
LOOP:     LD        A,[X+]         ; [X] TO ACC
          ADD       A,#066         ; ADD HEX 66 TO ACC
          ADC       A,[B]          ; ADD [B] TO ACC
          DCOR      A              ; DECIMAL CORRECT RESULT
          X         A,[B+]         ; RESULT TO [B]
          IFBNE     #12            ; IF STILL IN DATA FIELD
          JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
          IFC                      ; IF TERMINAL CARRY
          JP        OVFLOW         ;    JUMP TO OVERFLOW
```

BCD SUBTRACTION:

```
          LD        X,#16          ; NO LEADING ZERO
          LD        B,#24          ;    INDICATES DECIMAL
          C
LOOP:     LD        A,[X+]         ; [X] TO ACC
          SUBC      A,[B]          ; SUBTRACT [B] FROM ACC
          DCOR      A              ; DECIMAL CORRECT RESULT
          X         A,[B+]         ; RESULT TO [B]
          IFBNE     #12            ; IF STILL IN DATA FIELD
          JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
          IFNC                     ; IF TERMINAL BORROW
          JP        NEGRSLT        ;    JUMP TO NEGATIVE RESULT
```

The astute observer will notice that these previous additions and subtractions are not "adding machine" type arithmetic operations in that the result replaces the second operand rather than the first. The following program examples illustrate "adding machine" type operation where the result replaces the first operand. With subtraction, this entails the result replacing the minuend rather than the subtrahend. Note that the B and X pointers are now reversed.

BINARY ADDITION:

```
        LD        B,#16          ; B POINTER AT FIRST OPERAND
        LD        X,#24          ; X POINTER AT SECOND OPERAND
        RC                       ; RESET CARRY TO START
LOOP:   LD        A,[X+]         ; [X] TO ACC
        ADC       A,[B]          ; ADD [B] TO ACC
        X         A,[B+]         ; RESULT TO [B]
        IFBNE     #4             ; IF STILL IN DATA FIELD
        JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
        IFC                      ; IF TERMINAL CARRY
        JP        OVFLOW         ;    JUMP TO OVERFLOW
```

BINARY SUBTRACTION:

```
        LD        B,#010         ; B POINTER AT FIRST OPERAND
        LD        X,018          ; X POINTER AT SECOND OPERAND
        SC                       ; RESET BORROW TO START
LOOP:   LD        A,[X+]         ; [X] TO ACC
        X         A,[B]          ; EXCHANGE [B] AND ACC
        SUBC      A,[B]          ; SUBTRACT [B] FROM ACC
        X         A,[B+]         ; RESULT TO [B]
        IFBNE     #4             ; IF STILL IN DATA FIELD
        JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
        IFNC                     ; IF TERMINAL BORROW
        JP        NEGRSLT        ;    JUMP TO NEGATIVE RESULT
```

BCD ADDITION:

```
        LD        B,#010         ; B POINTER AT FIRST OPERAND
        LD        X,#018         ; X POINTER AT SECOND OPERAND
        RC                       ; RESET CARRY TO START
LOOP:   LD        A,[X+]         ; [X] TO ACC
        ADD       A,#066         ; ADD HEX66 TO ACC
        ADC       A,[B]          ; ADD [B] TO ACC
        DCOR      A              ; DECIMAL CORRECT RESULT
        X         A,[B+]         ; RESULT TO [B]
        IFBNE     #4             ; IF STILL IN DATA FIELD
        JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
        IFC       ;              ; IF TERMINAL CARRY
        JP        OVFLOW         ;    JUMP TO OVERFLOW
```

BCD SUBTRACTION:

```
        LD        B,#16          ; B POINTER AT FIRST OPERAND
        LD        X,#24          ; X POINTER AT SECOND OPERAND
        SC                       ; RESET BORROW TO START
LOOP:   LD        A,[X+]         ; [X] TO ACC
        X         A,[B]          ; EXCHANGE [B] AND ACC
        SUBC      A,[B]          ; SUBTRACT [B] FROM ACC
        DCOR      A              ; DECIMAL CORRECT RESULT
        X         A,[B+]         ; RESULT TO [B]
        IFBNE     #4             ; IF STILL IN DATA FIELD
        JP        LOOP           ;    JUMP BACK TO REPEAT LOOP
        IFNC                     ; IF TERMINAL BORROW
        JP        NEGRSLT        ;    JUMP TO NEGATIVE RESULT
```

Let us now consider a hybrid arithmetic example, where we wish to add five successive bytes of a data table in ROM program memory to a two byte sum, and then subtract the SUM result from a two byte total TOT. Let us further assume that the ROM table is located starting at program memory address 0401, while SUM and TOT are at RAM data memory locations [1, 0] and [3, 2] respectively, and that we wish to encode the program as a subroutine.

ROM Table:

```
. = 0401
. Byte 102
. Byte 41
. Byte 31
. Byte 26
. Byte 5
```

ROM Table Accessed Top Down

```
SUMLO = 0
SUMHI = 1
TOTLO = 2
TOTHI = 3
```

```
ARITH1: LD          X,#5        ; SET UP ROM TABLE POINTER
        LD          B,#0        ; SET UP SUM POINTER
LOOP:   RC                      ; RESET CARRY TO START ADDITION
        LD          A,X         ; ROM POINTER TO ACC
        LAID                    ; TABLE VALUE FROM ROM TO ACC
        ADC         A,[B]       ; ADD SUMLO TO ACC
        X           A,[B+]      ; RESULT TO SUMLO
        CLR         A           ; CLEAR ACC
        ADC         A,[B]       ; ADD SUMHI TO ACC
        X           A,[B-]      ; RESULT TO SUMHI
        DRSZ        X           ; DECR AND TEST ROM PTR FOR ZERO
        JP          LOOP        ; JUMP BACK TO REPEAT LOOP
                                ;    IF X PTR NOT ZERO
        SC                      ; RESET BORROW TO START SUBTRACTION
        LD          B,#2        ; SET UP TOT POINTER
LUP:    LD          A,[X+]      ; SUBTRAHEND (SUM) TO ACC
        X           A,[B]       ; REVERSE OPERANDS
        SUBC        A,[B]       ;    FOR SUBTRACTION
        X           A,[B+]      ; RESULT TO TOT
        IFBNE       #4          ; IF STILL IN TOT FIELD
        JP          LUP         ;    JUMP BACK TO REPEAT LUP
        RET                     ; RETURN FROM SUBROUTINE
```

2.0 MULTIPLICATION

The COP800 multiplications are all based on starting the multiplier in the low order end of the double length product space. The high end of the double length product space is initially cleared, and then the double length product is shifted right one bit. The bit shifted out from the low order end represents the low order bit of the multiplier. If this bit is a "1", the multiplicand is added to the high end of the double length product space. The entire shifting process and the conditional addition of the multiplicand to the upper end of the double length product is then repeated. The number of shift cycles is equal to the number of bit positions in the multiplier plus one extra shift cycle. This extra terminal shift cycle is necessary to correctly align the resultant product.

Note that an M byte multiplicand multiplied by an N byte multiplier will result in an M + N byte double length product. However, these multiplication subroutines will only use 2M + N + 1 bytes of RAM memory space, since the multiplier initially occupies the low order end of the double length product. The one extra byte is necessary for the shift counter CNTR.

The minimal code (28 byte) general multiplication subroutine is shown with two different examples, MY2448 and MY4824. Both examples multiply 24 bits by 48 bits. The MY2448 subroutine uses the 48-bit operand as the multiplier, and consequently uses minimal RAM as well as minimal program code. The MY4824 subroutine uses the 24-bit operand as the multiplier, and consequently executes considerably faster than the minimal RAM MY2448 subroutine.

MPY88 — 8 by 8 Multiplication Subroutine
— 19 Bytes
— 180 Instruction Cycles
— Minimum Code

MLT88 — Fast 8 by 8 Multiplication Subroutine
— 42 Bytes
— 145 Instruction Cycles

VFM88 — Very Fast 8 by 8 Multiply Subroutine
— 96 Bytes
— 116 Instruction Cycles

MPY168 — Fast 16 by 8 Multiplication Subroutine
— 36 Bytes
— 230 Instruction Cycles Average
— 254 Instruction Cycles Maximum

MPY816 (or MPY824, MPY832)
— 8 by 16 (or 24, 32) Multiply Subroutine
— 22 Bytes
— 589 (or 1065, 1669) Instruction Cycles Average
— 597 (or 1077, 1685) Instruction Cycles Maximum
— Minimum Code, Minimum RAM
— Extendable Routine for MPY8XX by Changing Parameters, with Number of Bytes (22) Remaining a Constant

MPY248 — Fast 24 by 8 Multiplication Subroutine
— 47 Bytes
— 289 Instruction Cycles Average
— 333 Instruction Cycles Maximum

MX1616 — Fast 16 by 16 Multiplication Subroutine
— 39 Bytes
— 498 Instruction Cycles Average
— 546 Instruction Cycles Maximum

MP1616 — 16 by 16 Multiplicand Subroutine
— 29 Bytes
— 759 Instruction Cycles Average
— 807 Instruction Cycles Maximum
— Almost Minimum Code

MY1616 (or MY1624, MY1632)
— 28 Bytes
— 16 by 16 (or 24, 32) Multiply Subroutine
— 861 (or 1473, 2213) Inst. Cycles Average
— 1029 (or 1725, 2549) Inst. Cycles Maximum
— Minimum Code, Minimum RAM
— Extendable Routne for MY16XX by Changing Parameters, with Number of Bytes (28) Remaining a Constant

Minimal general multiplication subroutine for any number of bytes in multiplicand and multiplier
— 28 Bytes
— Minimum Code
— MY2448 Used as First Example, with Minimum RAM and
4713 Instruction Cycles Average
5457 Instruction Cycles Maximum
— MY4824 Used as Second Example, with Non Minimal RAM and
2751 Instruction Cycles Average
3483 Instruction Cycles Maximum

MPY88—8 BY 8 MULTIPLICATION SUBROUTINE

```
          MINIMUM CODE
          19 BYTES
          180 INSTRUCTION CYCLES
          MULTIPLICAND IN [0]       (ICAND)
          MULTIPLIER IN [1]         (IER)
          PRODUCT IN [2,1]          (PROD)
MPY88:    LD           CNTR,#9      ;  LD CNTR WITH LENGTH OF
          RC                        ;     MULTIPLIER FIELD + 1
          LD           B,#2
          CLR          A            ;  CLEAR UPPER PRODUCT
M88LUP:   RRC          A            ;  RIGHT SHIFT
          X            A,[B-]       ;     UPPER PRODUCT
          LD           A,[B]
          RRC          A            ;  RIGHT SHIFT LOWER
          X            A,[B-]       ;     PRODUCT/MULTIPLIER
          CLR          A            ;  CLR ACC AND TEST LOW
          IFC                       ;     ORDER MULTIPLER BIT
          LD           A,[B]        ;  MULTIPLICAND TO ACC IF
          RC                        ;     LOW ORDER BIT = 1
          LD           B,#2         ;  ADD MULTIPLICAND TO
          ADC          A,[B]        ;     UPPER PRODUCT
          DRSZ         CNTR         ;  DECREMENT AND TEST
          JP           M88LUP       ;     CNTR FOR ZERO
          RET                       ;  RETURN FROM SUBROUTINE
```

MLT88—FAST 8 BY 8 MULTIPLICATION SUBROUTINE

```
          42 BYTES
          145 INSTRUCTION CYCLES
          MULTIPLICAND IN [0]          (ICAND)
          MULTIPLIER IN [1]            (IER)
          PRODUCT IN [2,1]             (PROD)
MLT88:    LD          CNTR,#3          ;  LOAD CNTR WITH
          RC                           ;   1/3 OF LENGTH OF
          LD          B,#2             ;   (MULTIPLIER FIELD + 1)
          CLR         A                ;  CLEAR UPPER PRODUCT
;
ML88LP:   RRC         A                ;  RIGHT SHIFT   ***
          X           A,[B-]           ;     UPPER PRODUCT
          LD          A,[B]
          RRC         A                ;  RIGHT SHIFT LOWER
          X           A,[B-]           ;     PRODUCT/MULTIPLIER
          CLR         A                ;  CLR ACC AND TEST LOW
          IFC                          ;     ORDER MULTIPLIER BIT
          LD          A,[B]            ;  MULTIPLICAND TO ACC IF
          RC                           ;     LOW ORDER BIT = 1
          LD          B,#2             ;  ADD MULTIPLICAND TO
          ADC         A,[B]            ;     UPPER PRODUCT ***
;
          RRC         A                ;  REPEAT THE ABOVE
          X           A,[B-]           ;     11 BYTE
          LD          A,[B]            ;     13 INSTRUCTION
          RRC         A                ;     CYCLE PROGRAM
          X           A,[B-]           ;     SECTION (WITH
          CLR         A                ;     THE *** DELIMITERS)
          IFC                          ;     TWICE MORE FOR A
          LD          A,[B]            ;     TOTAL OF THREE TIMES
          RC
          LD          B,#2
          ADC         A,[B]            ;  END OF SECOND REPEAT
;
          RRC         A                ;  START OF THIRD REPEAT
          X           A,[B-]
          LD          A,[B]
          RRC         A
          X           A,[B-]
          CLR         A
          IFC
          LD          A,[B]
          RC
          LD          B,#2
          ADC         A,[B]            ;  END OF THIRD REPEAT
;
          DRSZ        CNTR             ;  DECREMENT AND TEST
          JMP         ML88LP           ;   CNTR FOR ZERO
          RET                          ;  RETURN FROM SUBROUTINE
```

VFM88—VERY FAST 8 BY 8 MULTIPLY SUBROUTINE

```
          96 BYTES
          116 INSTRUCTION CYCLES

          MULTIPLICAND IN [0]        (ICAND)
          MULTIPLIER IN [1]          (IER)
          PRODUCT IN [2,1]           (PROD)
VFM88:    RC
          LD          B,#2
          LD          [B-],#0        ;  CLEAR UPPER PRODUCT
          LD          A,[B]
          RRC         A              ;  RIGHT SHIFT LOWER
          X           A,[B-]         ;   PRODUCT/MULTIPLIER
          CLR         A              ;  CLR ACC AND TEST LOW
          IFC                        ;     ORDER MULTIPLIER BIT
          LD          A,[B]          ;  MULTIPLICAND TO ACC IF
          RC                         ;     LOW ORDER BIT = 1
          LD          B,#2           ;  ADD MULTIPLICAND TO
          ADC         A,[B]          ;     UPPER PRODUCT
;
          RRC         A              ;  RIGHT SHIFT   ***
          X           A,[B-]         ;     UPPER PRODUCT
          LD          A,[B]
          RRC         A              ;  RIGHT SHIFT LOWER
          X           A,[B-]         ;     PRODUCT/MULTIPLIER
          CLR         A              ;  CLR ACC AND TEST LOW
          IFC                        ;     ORDER MULTIPLIER BIT
          LD          A,[B]          ;  MULTIPLICAND TO ACC IF
          RC                         ;     LOW ORDER BIT = 1
          LD          B,#2           ;  ADD MULTIPLICAND TO
          ADC         A,[B]          ;     UPPER PRODUCT ***
;
;    THE ABOVE 11 BYTE, 13 INSTRUCTION CYCLE SECTION WITH THE ***
;    DELIMITERS REPRESENTS THE PROCESSING FOR ONE MULTIPLIER BIT.
;
;
;         ---                        ;  REPEAT THE
;                                    ;  ABOVE SECTION
;         ---                        ;  SIX MORE TIMES,
;                                    ;  FOR A TOTAL
;         ---                        ;  OF SEVEN TIMES
;
          RRC         A              ;  RIGHT SHIFT
          X           A,[B-]         ;  UPPER PRODUCT
          LD          A,[B]
          RRC         A              ;  RIGHT SHIFT LOWER
          X           A,[B]          ;     PRODUCT/MULTIPLIER
          RET                        ;  RETURN FROM SUBROUTINE
;
;
;
```

MPY168—FAST 16 BY 8 MULTIPLICATION SUBROUTINE

```
            36 BYTES
            230 INSTRUCTION CYCLES AVERAGE
            254 INSTRUCTION CYCLES MAXIMUM

            MULTIPLICAND IN [1,0]          (ICAND)
            MULTIPLIER IN [2]              (IER)
            PRODUCT IN [4,3,2]             (PROD)
MPY168:     LD          CNTR,#9            ;  LD CNTR WITH LENGTH OF
            RC                             ;   MULTIPLIER FIELD + 1
            LD          B,#4
            LD          [B-],#0            ;  CLEAR
            LD          [B-],#0            ;     UPPER PRODUCT
            JP          MP168S
M168LP:     RRC         A                  ;  RIGHT SHIFT UPPER
            X           A,[B-]             ;   BYTE OF PRODUCT
            LD          A,[B]
            RRC         A                  ;  RIGHT SHIFT MIDDLE
            X           A,[B-]             ;   BYTE OF PRODUCT
MP168S:     LD          A,[B]
            RRC         A                  ;  RIGHT SHIFT LOWER
            X           A,[B]              ;     PRODUCT/MULTIPLIER
            IFNC                           ;  TEST LOWER BIT
            JP          MP168T             ;     OF MULTIPLIER
            RC                             ;  CLEAR CARRY
            LD          B,#0               ;  LOWER BYTE OF
            LD          A,[B]              ;     MULTIPLICAND TO ACC
            LD          B,#3               ;  ADD LOWER BYTE OF
            ADC         A,[B]              ;     MULTIPLICAND TO
            X           A,[B]              ;     MIDDLE BYTE OF PROD
            LD          B,#1               ;  UPPER BYTE OF
            LD          A,[B]              ;     MULTIPLICAND TO ACC
            LD          B,#4               ;  ADD UPPER BYTE OF ICAND
            ADC         A,[B]              ;     TO UPPER BYTE OF PROD
            DRSZ        CNTR               ;  DECREMENT CNTR AND JUMP
            JP          M168LP             ;     BACK TO LOOP; CNTR
                                           ;     CANNOT EQUAL ZERO
MP168T:     LD          B,#4               ;  HIGH ORDER PRODUCT
            LD          A,[B]              ;     BYTE TO ACC
            DRSZ        CNTR               ;  DECREMENT AND TEST IF
            JP          M168LP             ;     CNTR EQUAL TO ZERO
            RET                            ;  RETURN FROM SUBROUTINE
```

MPY816—(OR MPY824, MPY832) 8 BY 16 (OR 24, 32) MULTIPLY SUBROUTINE

```
          MINIMUM CODE, MINIMUM RAM
          22 BYTES
          589 (OR 1065, 1669) INSTR. CYCLES AVERAGE
          597 (OR 1077, 1685) INSTR. CYCLES MAXIMUM
          EXTENDABLE ROUTINE FOR MPY8XX BY CHANGING
           PARAMETERS, WITH NUMBER OF BYTES (22)
           REMAINING A CONSTANT.

          MULTIPLICAND IN [0]              (ICAND)
          MULTIPLIER IN [2,1] FOR 16 BIT (IER)
                       OR [3,2,1] for 24 BIT
                       OR [4,3,2,1] for 32 BIT
          PRODUCT IN [3,2,1] FOR 16 BIT    (PROD)
                       OR [4,3,2,1] FOR 24 BIT
                       OR [5,4,3,2,1] FOR 32 BIT

MPY816:   LD           CNTR,#17              ;  LD CNTR WITH LENGTH OF
                                             ;   MULTIPLIER FIELD + 1
                                             ;   #17 FOR MPY816 16 BIT
                                             ;   (#25 FOR MPY824 24 BIT)
                                             ;   (#33 FOR MPY832 32 BIT)
          RC
          LD           B,#3                  ;  #3 FOR MPY816
                                             ;  (#4 FOR MPY824)
                                             ;  (#5 FOR MPY832)
          LD           [B-],#0               ;  CLEAR UPPER PRODUCT
M8XXLP:   LD           A,[B]                 ;  FIVE INSTRUCTION
M8XXL:    RRC          A                     ;     PROGRAM LOOP TO
          X            A,[B-]                ;     RIGHT SHIFT
          IFBNE        #0                    ;     PRODUCT/MULTIPLIER
          JP           M8XXLP                ;  LOOP JUMP BACK
          CLR          A                     ;  CLR ACC AND TEST LOW
          IFNC                               ;     ORDER MULTIPLIER BIT
          JP           M8XXT                 ;  JP IF LOW ORDER BIT = 0
          RC
          LD           B,#0
          LD           A,[B]                 ;  MULTIPLICAND TO ACC
M8XXT:    LD           B,#3                  ;  #3 FOR MPY816
                                             ;  (#4 FOR MPY824)
                                             ;  (#5 FOR MPY832)
          ADC          A,[B]                 ;  ADD MULTIPLICAND TO
                                             ;     UPPER BYTE OF PRODUCT
          DRSZ         CNTR                  ;  DECREMENT AND TEST
          JP           M8XXL                 ;     CNTR FOR ZERO
          RET                                ;  RETURN FROM SUBROUTINE
```

MPY248—FAST 24 BY 8 MULTIPLICATION SUBROUTINE

```
          47 BYTES
          289 INSTRUCTION CYCLES AVERAGE
          333 INSTRUCTION CYCLES MAXIMUM
          MULTIPLICAND IN [2,1,0]      (ICAND)
          MULTIPLIER IN [3]            (IER)
          PRODUCT IN [6,5,4,3]         (PROD)

MPY248:   LD        CNTR,#9          ; LD CNTR WITH LENGTH OF
          RC                         ;  MULTIPLIER FIELD + 1
          LD        B,#6
          LD        [B-],#0          ; CLEAR THREE
          LD        [B-],#0          ;    UPPER BYTES
          LD        [B-],#0          ;    OF PRODUCT
          JP        MP248S           ; JUMP TO START
M248LP:   RRC       A                ; RIGHT SHIFT HIGH
          X         A,[B-]           ;    ORDER PRODUCT BYTE
          LD        A,[B]
          RRC       A                ; RIGHT SHIFT NEXT LOWER
          X         A,[B-]           ;    ORDER PRODUCT BYTE
          LD        A,[B]
          RRC       A                ; RIGHT SHIFT NEXT LOWER
          X         A,[B-]           ;    ORDER PRODUCT BYTE
MP248S:   LD        A,[B]
          RRC       A                ; RIGHT SHIFT LOW ORDER
          X         A,[B]            ;    PRODUCT/MULTIPLIER
          IFNC                       ; TEST LOW ORDER
          JP        MP248T           ;    MULTIPLIER BIT
          RC
          LD        B,#0             ; LOAD ACC WITH LOW ORDER
          LD        A,[B]            ;    MULTIPLICAND BYTE
          LD        B,#4             ; ADD LOW ORDER ICAND
          ADC       A,[B]            ;    BYTE TO NEXT TO LOW
          X         A,[B]            ;    ORDER PRODUCT BYTE
          LD        B,#1             ; LOAD ACC WTIH MIDDLE
          LD        A,[B]            ;    MULTIPLICAND BYTE
          LD        B,#5             ; ADD MIDDLE ICAND BYTE
          ADC       A,[B]            ;    TO NEXT TO HIGH ORDER
          X         A,[B]            ;    MULTIPLICAND BYTE
          LD        B,#2             ; LOAD ACC WITH HIGH ORDER
          LD        A,[B]            ;    MULTIPLICAND BYTE
          LD        B,#6             ; ADD HIGH ORDER ICAND BYTE
          ADC       A,[B]            ;    TO HIGH ORDER PROD BYTE
          DRSZ      CNTR             ; DECREMENT CNTR AND JUMP
          JP        M248LP           ;    BACK TO LOOP; CNTR
                                     ;    CANNOT EQUAL ZERO
MP248T:   LD        B,#6             ; HIGH ORDER PRODUCT
          LD        A,[B]            ;    BYTE TO ACC
          DRSZ      CNTR             ; DECREMENT AND TEST
          JMP       M248LP           ;    CNTR FOR ZERO
          RET                        ; RETURN FROM SUBROUTINE
```

MX1616—FAST 16 BY 16 MULTIPLICATION SUBROUTINE

```
          39 BYTES
          498 INSTRUCTION CYCLES AVERAGE
          546 INSTRUCTION CYCLES AVERAGE

          MULTIPLICAND IN [1,0]      (ICAND)
          MULTIPLIER IN [3,2]        (IER)
          PRODUCT IN [5,4,3,2]       (PROD)

MX1616:   LD           CNTR,#17      ;  LD CNTR WITH LENGTH OF
          RC                         ;     MULTIPLIER FIELD + 1
          LD           B,#5
          LD           [B-],#0       ;  CLEAR UPPER TWO
          LD           [B-],#0       ;     PRODUCT BYTES
          JP           MXSTRT        ;  JUMP TO START
MX1616L:  RRC          A             ;  RIGHT SHIFT
          X            A,[B-]        ;   UPPER PRODUCT BYTE
          LD           A,[B]
          RRC          A             ;  RIGHT SHIFT NEXT LOWER
          X            A,[B-]        ;     PRODUCT BYTE
MXSTRT:   LD           A,[B]
          RRC          A             ;  RIGHT SHIFT PRODUCT
          X            A,[B-]        ;     UPPER MULTIPLIER BYTE
          LD           A,[B]
          RRC          A             ;  RIGHT SHIFT PRODUCT
          X            A,[B]         ;     LOWER MULTIPLIER BYTE
          IFNC                       ;  TEST LOW ORDER
          JP           MX1616T       ;     MULTIPLIER BIT
          RC
          LD           B,#0          ;  LOAD ACC WITH LOWER
          LD           A,[B]         ;     MULTIPLICAND BYTE
          LD           B,#4          ;  ADD LOWER ICAND BYTE
          ADC          A,[B]         ;     TO NEXT TO HIGH
          X            A,[B]         ;     ORDER PRODUCT BYTE
          LD           B,#1          ;  LOAD ACC WITH UPPER
          LD           A,[B]         ;     MULTIPLICAND BYTE
          LD           B,#5          ;  ADD UPPER ICAND BYTE TO
          ADC          A,[B]         ;     HIGH ORDER PRODUCT
          DRSZ         CNTR          ;  DECREMENT CNTR AND JUMP
          JP           MX1616L       ;     BACK TO LOOP; CNTR
                                     ;     CANNOT EQUAL ZERO
MX1616T:  LD           B,#5          ;  HIGH ORDER PRODUCT
          LD           A,[B]         ;     BYTE TO ACC
          DRSZ         CNTR          ;  DECREMENT AND TEST
          JP           MX1616L       ;   CNTR FOR ZERO
          RET                        ;  RETURN FROM SUBROUTINE
```

MP1616—16 BY 16 MULTIPLICATION SUBROUTINE

```
          MINIMUM CODE
          29 BYTES
          759 INSTRUCTION CYCLES AVERAGE
          807 INSTRUCTION CYCLES MAXIMUM]
          MULTIPLICAND IN [1,0]     (ICAND)
          MULTIPLIER IN [3,2]       (IER)
          PRODUCT IN [5,4,3,2]      (PROD)


MP1616:   LD        CNTR,#17     ;  LD CNTR WITH LENGTH OF
          RC                     ;     MULTIPLIER FIELD + 1
          LD        B,#5
          LD        [B-],#0      ;  CLEAR UPPER TWO
          LD        [B-],#0      ;     PRODUCT BYTES
M1616X:   LD        A,[B]        ;  FIVE INSTRUCTION
M1616L:   RRC       A            ;     PROGRAM LOOP TO
          X         A,[B-]       ;     RIGHT SHIFT
          IFBNE     #1           ;     PRODUCT/MULTIPLIER.
          JP        M1616X       ;     LOOP JUMP BACK
          CLR       A            ;  CLEAR ACC
          IFNC                   ;  TEST LOW ORDER
          JP        M1616T       ;     MULTIPLIER BIT
          RC
          LD        B,#0         ;  LOAD ACC WITH LOWER
          LD        A,[B]        ;     MULTIPLICAND BYTE
          LD        B,#4         ;  ADD LOWER ICAND BYTE
          ADC       A,[B]        ;     TO NEXT TO LOW
          X         A,[B]        ;     ORDER PRODUCT BYTE
          LD        B,#1         ;  LOAD ACC WITH UPPER
          LD        A,[B]        ;     MULTIPLICAND BYTE
M1616T:   LD        B,#5         ;  ADD UPPER ICAND BYTE TO
          ADC       A,[B]        ;     HIGH ORDER PRODUCT
          DRSZ      CNTR         ;  DECREMENT AND TEST
          JP        M1616L       ;     CNTR EQUAL TO ZERO
          RET                    ;  RETURN FROM SUBROUTINE
```

MY1616 (OR MY1624, MY1632)—16 BY 16 (OR 24, 32) MULTIPLY SUBROUTINE

```
          MINIMUM CODE, MINIMUM RAM
          28 BYTES
          861 (OR 1473, 2213) INST. CYCLES AVERAGE
          1029 (OR 1725,1473) INST. CYCLES MAXIMUM
          EXTENDABLE ROUTINE FOR MY16XX BY CHANGING
            PARAMETERS, WITH NUMBER OF BYTES (28)
            REMAINING A CONSTANT
          MULTIPLICAND IN [1,0]                 (ICAND)
          MULTIPLIER IN [3,2] FOR 16 BIT        (IER)
                     OR [4,3,2] FOR 24 BIT
                     OR [5,4,3,2] FOR 32 BIT
          PRODUCT IN [5,4,3,2] FOR 16 BIT       (PROD)
                  OR [6,5,4,3,2] FOR 24 BIT
                  OR [7,6,5,4,3,2] FOR 32 BIT


MY1616:   LD           CNTR,#17      ; LD CNTR WITH LENGTH OF
                                     ;    MULTIPLIER FIELD + 1
                                     ; #17 FOR MY1616
                                     ; (#25 FOR MY1624)
                                     ; (#33 FOR MY1632)
          LD           B,#5          ; #5 FOR MY1616
                                     ; (#6 FOR MY1624)
                                     ; (#7 FOR MY1632)
          LD           [B-],#0       ; CLEAR UPPER TWO
          LD           [B-],#0       ;    PRODUCT BYTES
          RC
MY16XS:   LD           A,[B]         ; FIVE INSTRUCTION
          RRC          A             ;    PROGRAM LOOP TO
          X            A,[B-]        ;    RIGHT SHIFT
          IFBNE        #1            ;    PRODUCT/MULTIPLIER
          JP           M16XS         ;    LOOP JUMP BACK
          IFNC                       ; TEST LOW ORDER
          JP           MY16XT        ;    MULTIPLIER BIT
          RC
          LD           B,#4          ; #4 FOR MY1616
                                     ; (#5 FOR MY1624)
                                     ; (#6 FOR MY1632)
          LD           X,#0          ; LOAD ACC WITH
MY16XL:   LD           A,[X+]        ;    MULTIPLICAND BYTES
          ADC          A,[B]         ; ADD MULTIPLICAND TO
          X            A,[B+]        ;    HI TWO PROD. BYTES
          IFBNE        #2            ; LOOP BACK FOR SECOND
          JP           MY16XL        ;    MULTIPLICAND BYTE
MY16XT:   LD           B,#5          ; #5 FOR MY1616
                                     ; (#6 FOR MY1624)
                                     ; (#7 FOR MY1632)
          DRSZ         CNTR          ; DECREMENT AND TEST
          JP           MY16XS        ;    CNTR EQUAL TO ZERO
          RET                        ; RETURN FROM INTERRUPT
;
```

MY2448—MINIMAL GENERAL MULTIPLICATION SUBROUTINE (28 BYTES)

```
    ANY NUMBER OF BYTES IN MULTIPLICAND
    AND MULTIPLIER
 FIRST EXAMPLE:     (MY2448)
    24 BY 48 MULTIPLICATION SUBROUTINE
          --28 BYTES
          --MINIMAL CODE, MINIMAL RAM
          --4713 INSTRUCTION CYCLES AVERAGE
          --5457 INSTRUCTION CYCLES MAXIMUM
    MULTIPLICAND IN [2,1,0]                    (ICAND)
    MULTIPLIER IN [8,7,6,5,4,3]                (IER)
    PRODUCT IN [11,10,9,8,7,6,5,4,3]           (PROD)

 SECOND EXAMPLE: (MY4824)
    48 BY 24 MULTIPLICATION SUBROUTINE
          --28 BYTES
          --MINIMAL CODE, NON MINIMAL RAM
          --2751 INSTRUCTION CYCLES AVERAGE
          --3483 INSTRUCTION CYCLES MAXIMUM
    MULTIPLICAND IN [5,4,3,2,1,0]              (ICAND)
    MULTIPLIER IN [8,7,6]                      (IER)
    PRODUCT IN [14,13,12,11,10,9,8,7,6]        (PROD)
```

```
MY2448:    ; (OR MY4824)
          LD           CNTR, #49  ; LD CNTR WITH LENGTH OF
                                  ;    MULTIPLIER FIELD + 1
                                  ; #49 FOR MY2448
                                  ; (#25 FOR MY4824)
          LD           B,#11      ; TOP OF PROD TO B PTR
                                  ; #11 FOR MY2448
                                  ; (#14 FOR MY4824)
CLRLUP:   LD           [B-],#0    ; CLR UNTIL TOP OF IER
          IFBNE        #8         ; #8 FOR BOTH MY2448
          JP           CLRLUP     ;    AND MY4824
          RC                      ; INITIALIZE CARRY
SHFTLP:   LD           A,[B]      ; RIGHT SHIFT PRODUCT
          ADC          A,[B]      ;    AND MULTIPLIER
          X            A,[B-]     ;    UNTIL TOP OF ICAND
          IFBNE        #2         ; #2 FOR MY2448
          JP           SHFTLP     ; (#5 FOR MY4824)
          IFNC                    ; TEST LOW ORDER
          JP           MYTEST     ;    MULTIPLIER BIT
          LD           B,#9       ; TOP OF IER + 1 TO B PTR
          LD           X,#0       ; START OF ICAND TO X PTR
          RC
ADDLUP:   LD           A,[X+]     ; ADD MULTIPLICAND TO TOP
          ADC          A,[B]      ;    OF PRODUCT ABOVE
          X            A,[B+]     ;    MULTIPLIER UNTIL TOP
          IFBNE        #12        ;    OF PRODUCT + 1
          JP           ADDLUP     ; #12 FOR MY2448
                                  ; (#15 FOR MY4824)
MYTEST:   LD           B,#11      ; TOP OF PROD TO B PTR
                                  ; #11 FOR MY2448
                                  ; (#14 FOR MY4824)
          DRSZ         CNTR       ; DECREMENT AND TEST
          JP           SHFTLP     ;    CNTR FOR ZERO
          RET                     ; RETURN FROM SUBROUTINE
```

3.0 DIVISION

The COP 800 divisions are all based on shifting the dividend left up into a test field equal in length to the number of bytes in the divisor. The divisor is resident immediately above this test field. After each shift cycle of the dividend into the test field, a trial subtraction is made of the test field minus the divisor. If the divisor is found equal to or less than the contents of the test field, then the divisor is subtracted from the test field and a 1's quotient digit is recorded by setting the low order bit of the dividend field. The dividend and test field left shift cycle is then repeated. The number of left shift cycles is equal to the number of bit positions in the dividend. The quotient from the division is formed in the dividend field, while the remainder from the division is resident in the test field.

Note that an M byte dividend divided by an N byte divisor will result in an M byte quotient and an N byte remainder.

These division algorithms will use M + 2N + 1 bytes of RAM memory space, since the test field is equal to the length of the divisor. The one extra byte is necessary for the shift counter CNTR.

In special cases where the dividend has an upper bound and the divisor has a lower bound, the upper bytes of the dividend may be used as the test field. One example is shown (DV2815), where a 28 bit dividend is divided by a 15-bit divisor. The dividend is less than 2**28 (upper nibble of high order byte is zero), while the divisor is greater than 2**12 (4096) and less than 2**15 (32768). In this case, the upper limit for the quotient is 2**28/2**12, which indicates a 16-bit quotient (2**16) and a 15-bit remainder. Consequently, the upper two bytes of the dividend may be used as the test field for the remainder, since the divisor is greater than the test field (upper two bytes of the 28-bit dividend) initially.

The minimal code (40 byte) general division subroutine is shown with the example DV3224, which divides a 32 bit dividend by a 24 bit divisor.

DIV88 — 8 by 8 Division Subroutine
— 24 Bytes
— 201 Instruction Cycles Average
— 209 Instruction Cycles Maximum
Minimum code

DV88 — Fast 8 by 8 Division Subroutine
— 28 Bytes
— 194 Instruction Cycles Average
— 202 Instruction Cycles Maximum

FDV88 — Very Fast 8 by 8 Division Subroutine
— 131 Bytes
— 146 Instruction Cycles Average
— 159 Instruction Cycles Maximum

DIV168 (or DIV248, DIV328)
— 16 (or 24, 32) by 8 Division Subroutine
— 26 Bytes
— 649 (or 1161, 1801) Instruction Cycles Average
— 681 (or 1209,1865) Instruction Cycles Maximum
— Minimum Code
— Extendable Routine for DIVXX8 by Changing Parameters, with Number of Bytes (26) Remaining a Constant

FDV168 — Fast 16 by 8 Division Subroutine
— 35 Bytes
— 481 Instruction Cycles Average
— 490 Instruction Cycles Maximum

FDV248 — Fast 24 by 8 Division Subroutine
— 38 Bytes
— 813 Instruction Cycles Average
— 826 Instruction Cycles Maximum

FDV328 — Fast 32 by 8 Division Subroutine
— 42 Bytes
— 1209 Instruction Cycles Average
— 1226 Instructions Maximum

Divide by 16 Subroutines:

DV1616 — 16 by 16 Division Subroutine
— 34 Bytes
— 979 Instruction Cycles Average
— 1067 Instruction Cycles Maximum
— Minimum Code

DV2416 (or DV3216)
— 24 (or 32) by 16 Division Subroutine
— 39 Bytes
— 1694 (or 2410) Inst. Cycles Average
— 1886 (or 2766) Inst. Cycles Maximum
— Minimum code
— Extendable Routine for DVXX16 by Changing Parameters, with Number of Bytes (39) Remaining a Constant

DX1616 — Fast 16 by 16 Division Subroutine
— 53 Bytes
— 638 Instruction Cycles Average
— 678 Instruction Cycles Maximum

DV2815 — Fast 28 by 15 Division Subroutine, Where the Dividend is Less Than 2**28 and the Divisor
is Greater than 2**12 (4096)
and Less than 2**15 (32768)
— 43 Bytes
— 640 Instruction Cycles Average
— 696 Instruction Cycles Maximum

DX3216 — Fast 32 by 16 Division Subroutine
— 70 Bytes
— 1511 Instruction Cycles Average
— 1591 Instruction Cycles Maximum

Minimal General Division Subroutine for any Number of Bytes in Dividend and Divisor
— 40 Bytes
— Minimal Code
— DV3224 Used as Example, with
3879 Instruction Cycles Average
4535 Instruction Cycles Maximum

DIV88—8 BY 8 DIVISION SUBROUTINE

```
          MINIMUM CODE
          24 BYTES
          201 INSTRUCTION CYCLES AVERAGE
          209 INSTRUCTION CYCLES MAXIMUM
          DIVIDEND     IN [0]      (DD)
          DIVISOR IN [2]           (DR)
          QUOTIENT IN [0]          (QUOT)
          REMAINDER IN [1]         (TEST FIELD)

DIV88:    LD           CNTR,#8     ;  LOAD CNTR WITH LENGTH
          LD           B,#1        ;     OF DIVIDEND FIELD
          LD           [B],#0      ;  CLEAR TEST FIELD
DIV88S    RC
          LD           B,#0
          LD           A,[B]
          ADC          A,[B]       ;  LEFT SHIFT DIVIDEND
          X            A,[B+]
          LD           A,[B]
          ADC          A,[B]       ;  LEFT SHIFT TEST FIELD
          X            A,[B]
          LD           A,[B+]      ;  TEST FIELD TO ACC
          SC                       ;  TEST SUBTRACT DIVISOR
          SUBC         A,[B]       ;     FROM TEST FIELD
          IFNC                     ;  TEST IF BORROW
          JP           DIV88B      ;     FROM SUBTRACTION
          LD           B,#1        ;  SUBTRACTION RESULT
          X            A,[B-]      ;     TO TEST FIELD
          SBIT         0,[B]       ;  SET QUOTIENT BIT
DIV88B:   DRSZ         CNTR        ;  DECREMENT AND TEST
          JP           DIV88S      ;     CNTR FOR ZERO
          RET                      ;  RETURN FROM SUBROUTINE
```

DV88—FAST 8 BY 8 DIVISION SUBROUTINE

```
          28 BYTES
          194 INSTRUCTION CYCLES AVERAGE
          202 INSTRUCTION CYCLES MAXIMUM
          DIVIDEND     IN [0]      (DD)
          DIVISOR IN [2]           (DR)
          QUOTIENT IN [0]          (QUOT)
          REMAINDER IN [1]         (TEST FIELD)
DV88:     LD           CNTR,#8     ;  LOAD CNTR WITH LENGTH
          LD           B,#1        ;   OF DIVIDEND FIELD
          LD           [B-],#0     ;  CLEAR TEST FIELD
          RC
DV88S:    LD           A,[B]
          ADC          A,[B]       ;  LEFT SHIFT DIVIDEND
          X            A,[B+]
          LD           A,[B]
          ADC          A,[B]       ;  LEFT SHIFT TEST FIELD
          X            A,[B]
          LD           A,[B+]      ;  TEST FIELD TO ACC
          SC                       ;  TEST SUBTRACT DIVISOR
          SUBC         A,[B]       ;     FROM TEST FIELD
          IFNC                     ;  TEST IF BORROW
          JP           DV88B       ;     FROM SUBTRACTION
          LD           B,#1        ;  SUBTRACTION RESULT
          X            A,[B-]      ;  TO TEST FIELD
          SBIT         0,[B]       ;  SET QUOTIENT BIT
          RC
          DRSZ         CNTR        ;  DECREMENT AND TEST
          JP           DV88S       ;     CNTR FOR ZERO
          RET                      ;  RETURN FROM SUBROUTINE
DV88B:    LD           B,#0
          DRSZ         CNTR        ;  DECREMENT AND TEST
          JP           DV88S       ;     CNTR FOR ZERO
          RET                      ;  RETURN FROM SUBROUTINE
```

FDV88—VERY FAST 8 BY 8 DIVISION SUBROUTINE

```
          131 BYTES
          146 INSTRUCTION CYCLES AVERAGE
          159 INSTRUCTION CYCLES MAXIMUM
          DIVIDEND IN [0]          (DD)
          DIVISOR IN [2]           (DR)
          QUOTIENT IN [0]          (QUOT)
          REMAINDER IN [1]         (TEST FIELD)
FDV88:    LD          B,#1
          LD          [B-],#0     ;  CLEAR TEST FIELD
          RC
          LD          A,[B]
          ADC         A,[B]       ;  LEFT SHIFT DIVIDEND
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]       ;  LEFT SHIFT TEST FIELD
          X           A,[B]
          LD          A,[B+]      ;  TEST FIELD TO ACC
          SC                      ;  TEST SUBTRACT DIVISOR
          SUBC        A,[B]       ;     FROM TEST FIELD
          IFNC                    ;  TEST IF BORROW
          JP          DVBP1       ;     FROM SUBTRACTION
          LD          B,#1        ;  SUBTRACTION RESULT
          X           A,[B-]      ;     TO TEST FIELD
          SBIT        0,[B]       ;  SET QUOTIENT BIT
          RC
DVBP1:    LD          B,#0        ;  THIS 16 BYTE SECTION
          LD          A,[B]       ;  OF PROGRAM CODE
          ADC         A,[B]       ;  CONTAINS
          X           A,[B+]      ;  16 INSTRUCTIONS,
          LD          A,[B]       ;  AND REPRESENTS THE
          ADC         A,[B]       ;  PROCESSING FOR THE
          X           A,[B]       ;  GENERATION OF
          LD          A,[B+]      ;  1 QUOTIENT BIT.
          SC                      ;
          SUBC        A,[B]       ;  THE PROGRAM CODE
          IFNC                    ;  EXECUTION TIMES IS 16
          JP          DVBP2       ;  INSTRUCTION CYCLES
          LD          B,#1        ;  FOR A 0'S QUOTIENT BIT
          X           A,[B-]      ;  AND 19 INSTRUCTION
          SBIT        0,[B]       ;  CYCLES FOR A 1'S
          RC                      ;  QUOTIENT BIT.
;         ---                     ;
DVBP2:    LD          B,#0        ;  REPEAT THE ABOVE
;         ---                     ;
;DVBP3:
;         ---                     ;SECTION OF CODE FIVE
;DVBP4:
;         ---                     ;MORE TIMES FOR A
;DVBP5:
;         ---                     ;TOTAL OF SIX TIMES
;DVBP6:
;         ---                     ;
;
DVBP7:    LD          B,#0
          LD          A,[B]
          ADC         A,[B]       ;  LEFT SHIFT DIVIDEND
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]       ;  LEFT SHIFT TEST FIELD
          X           A,[B]
          LD          A,[B+]      ;  TEST FIELD TO ACC
          SC                      ;  TEST SUBTRACT DIVISOR
          SUBC        A,[B]       ;     FROM TEST FIELD
          IFNC                    ;  TEST BORROW FROM SUBC
          RET                     ;  RETURN FROM SUBROUTINE
          LD          B,#1        ;  SUBTRACTION RESULT
          X           A,[B-]      ;     TO TEST FIELD
          SBIT        0,[B]       ;  SET QUOTIENT BIT
          RET                     ;  RETURN FROM SUBROUTINE
```

DIV168—16 (OR 24, 32) BY 8 DIVISION SUBROUTINE

```
          MINIMUM CODE
          26 BYTES
          649 (or 1161,1801) INST. CYCLES AVERAGE
          681 (or 1209,1865) INST. CYCLES MAXIMUM
          EXTENDABLE ROUTINE FOR DIVXX8 BY CHANGING
          PARAMETERS, WITH NUMBER OF BYTES (26)
          REMAINING A CONSTANT
          DIVIDEND IN [1,0] FOR 16 BIT              (DD)
                   OR [2,1,0] FOR 24 BIT
                   OR [3,2,1,0] FOR 32 BIT
          DIVISOR IN [3] FOR 16 BIT                 (DR)
                   OR [4] FOR 24 BIT
                   OR [5] FOR 32 BIT
          QUOTIENT IN [1,0] FOR 16 BIT              (QUOT)
                   OR [2,1,0] FOR 24 BIT
                   OR [3,2,1,0] FOR 32 BIT
          REMAINDER IN [2] FOR 16 BIT               (TEST FIELD)
                   OR [3] FOR 24 BIT
                   OR [4] FOR 32 BIT

DIV168:   LD       CNTR,#16     ; LOAD CNTR WITH LENGTH
                                ;    OF DIVIDEND FIELD
                                ; #16 FOR DIV168
                                ; (#24 FOR DIV248)
                                ; (#32 FOR DIV328)
          LD       B,#2         ; (#3 FOR DIV168)
                                ; (#3 FOR DIV248)
                                ; (#4 FOR DIV328)
          LD       [B],#0       ; CLEAR TEST FIELD
DVXX8L:   RC
          LD       B,#0
DXX8LP:   LD       A,[B]        ; LEFT SHIFT DIVIDEND
          ADC      A,[B]        ;    AND TEST FIELD
          X        A,[B+]
          IFBNE    #3           ; #3 FOR DIV168
          JP       DXX8LP       ; (#4 FOR DIV248)
                                ; (#5 FOR DIV328)
          LD       A,[B-]       ; DIVISOR TO ACCUMULATOR
          IFC                   ; TEST IF BIT SHIFTED OUT
          JP       DVXX8S       ;    OF TEST FIELD***
          IFGT     A,[B]        ; TEST DIVISOR GREATER
          JP       DVXX8T       ;    THAN REMAINDER
          SC                    ;
DVXX8S:   X        A,[B]        ; REMAINDER TO ACC
          SUBC     A,[B]        ; SUBTRACT DIVISOR
          X        A,[B]        ;    FROM REMAINDER
          LD       B,#0
          SBIT     0,[B]        ; SET QUOTIENT BIT
DVXX8T:   DRSZ     CNTR         ; DECREMENT AND TEST
          JP       DVXX8L       ;    CNTR FOR ZERO
          RET                   ; RETURN FROM SUBROUTINE


;
;
;   ***   SPECIAL CASE FOR DIVISION WHERE NUMBER OF BYTES
;         IN DIVIDEND IS GREATER THAN NUMBER OF BYTES IN DIVISOR, AND
;         DIVISOR CONTAINS A HIGH ORDER 1'S BIT. THE SHIFTED DIVIDEND
;         MAY CONTAIN A HIGH ORDER 1'S BIT IN THE TEST FIELD AND
;         YET BE SMALLER THAN THE DIVISOR SO THAT NO SUBTRACTION
;         OCCURS. IN THIS CASE A 1'S BIT WILL BE SHIFTED OUT OF
;         THE TEST FIELD AND AN OVERRIDE SUBTRACTION MUST BE PERFORMED
```

FDV168—FAST 16 BY 8 DIVISION SUBROUTINE

```
           35 BYTES
           481 INSTRUCTION CYCLES AVERAGE
           490 INSTRUCTION CYCLES MAXIMUM
           DIVIDEND IN [1,0]           (DD)
           DIVISOR IN [3]              (DR)
           QUOTIENT IN [1,0]           (QUOT)
           REMAINDER IN [2]            (TEST FIELD)

FDV168:    LD          CNTR,#16        ;  LOAD CNTR WITH LENGTH
           LD          B,#3            ;     OF DIVIDEND FIELD
           LD          [B],#0          ;  CLEAR TEST FIELD
FD168S:    LD          B,#0
FD168L:    RC
           LD          A,[B]
           ADC         A,[B]           ;  LEFT SHIFT DIVIDEND LO
           X           A,[B+]
           LD          A,[B]
           ADC         A,[B]           ;  LEFT SHIFT DIVIDEND HI
           X           A,[B+]
           LD          A,[B]
           ADC         A,[B]           ;  LEFT SHIFT TEST FIELD
           X           A,[B]
           LD          A,[B+]          ;  TEST FIELD TO ACC
           IFC                         ;  TEST IF BIT SHIFTED OUT
           JP          FD168B          ;     OF TEST FIELD***
           SC                          ;  TEST SUBTRACT DIVISOR
           SUBC        A,[B]           ;     FROM TEST FIELD
           IFNC                        ;  TEST IF BORROW
           JP          FD168T          ;     FROM SUBTRACTION
FD168R:    LD          B,#2            ;  SUBTRACTION RESULT
           X           A,[B]           ;     TO TEST FIELD
           LD          B,#0
           SBIT        0,[B]           ;  SET QUOTIENT BIT
           DRSZ        CNTR            ;  DECREMENT AND TEST
           JP          FD168L          ;     CNTR FOR ZERO
           RET                         ;  RETURN FROM SUBROUTINE
FD168T:    DRSZ        CNTR            ;  DECREMENT AND TEST
           JP          FD168S          ;     CNTR FOR ZERO
           RET                         ;  RETURN FROM SUBROUTINE
FD168B:    SUBC        A,[B]           ;  SUBTRACT DIVISOR FROM
           JP          FD168R          ;     TEST FIELD***
```

FDV248—FAST 24 BY 8 DIVISION SUBROUTINE

```
          38 BYTES
          813 INSTRUCTION CYCLES AVERAGE
          826 INSTRUCTION CYCLES MAXIMUM
          DIVIDEND IN [2,1,0]       (DD)
          DIVISOR IN [4]            (DR)
          QUOTIENT IN [2,1,0]       (QUOT)
          REMAINDER IN [3]          (TEST FIELD)

FDV248:   LD          CNTR,#24      ;  LOAD CNTR WITH LENGTH
          LD          B,#4          ;     OF DIVIDEND FIELD
          LD          [B],#0        ;  CLEAR TEST FIELD
FD248S:   LD          B,#0
FD248L:   RC
          LD          A,[B]
          ADC         A,[B]         ;  LEFT SHIFT DIVIDEND LO
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]         ;  LEFT SHIFT DIVIDEND MID
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]         ;  LEFT SHIFT DIVIDEND HI
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]         ;  LEFT SHIFT TEST FIELD
          X           A,[B]
          LD          A,[B+]
          IFC                       ;  TEST IF BIT SHIFTED OUT
          JP          FD248B        ;     OF TEST FIELD ***
          SC                        ;  TEST SUBTRACT DIVISOR
          SUBC        A,[B]         ;     FROM TEST FIELD
          IFNC                      ;  TEST IF BORROW
          JP          FD248T        ;     FROM SUBTRACTION
FD248R:   LD          B,#3          ;  SUBTRACTION RESULT
          X           A,[B]         ;     TO TEST FIELD
          LD          B,#0
          SBIT        0,[B]         ;  SET QUOTIENT BIT
          DRSZ        CNTR          ;  DECREMENT AND TEST
          JP          FD248L        ;     CNTR FOR ZERO
          RET                       ;  RETURN FROM SUBROUTINE
FD248T:   DRSZ        CNTR          ;  DECREMENT AND TEST
          JP          FD248S        ;     CNTR FOR ZERO
          RET                       ;  RETURN FROM SUBROUTINE
FD248B:   SUBC        A,[B]         ;  SUBTRACT DIVISOR FROM
          JP          FD248R        ;     TEST FIELD ***
```

DV1616—16 (OR 24, 32) BY 16 DIVISION SUBROUTINE

```
          MINIMUM CODE
          34 BYTES
          979 (OR 1655,2459) INSTRUCTION CYCLES AVERAGE
          1067 (OR 1787,2635) INSTRUCTION CYCLES MAXIMUM
          DIVIDEND IN [1,0]          (DD)
          DIVISOR IN [5,4]           (DR)
          QUOTIENT IN [1,0]          (QUOT)
          REMAINDER IN [3,2]         (TEST FIELD)

DV1616:   LD           CNTR,#16      ;  LOAD CNTR WITH LENGTH
                                     ;     OF DIVIDEND FIELD
          LD           B,#3
          LD           [B-],#0       ;  CLEAR
          LD           [B],#0        ;     TEST FIELD
DV616S:   RC
          LD           X,#2          ;  INITIALIZE X POINTER
          LD           B,#0          ;  INITIALIZE B POINTER
DV616L:   LD           A,[B]         ;  LEFT SHIFT DIVIDEND
          ADC          A,[B]         ;     AND TEST FIELD
          X            A,[B+]
          IFBNE        #4
          JP           DV616L
          SC                         ;  RESET BORROW
          LD           A,[X+]        ;  TEST FIELD LO TO ACC
          SUBC         A,[B]         ;  SUBT DR LO FROM REM LO
          LD           A,[X]         ;  TEST FIELD HI TO ACC
          LD           B,#5
          SUBC         A,[B]         ;  SUBT DR HI FROM REM HI
          IFNC                       ;  TEST IF BORROW
          JP           DV616T        ;     FROM SUBTRACTION
          X            A,[X-]        ;  SUBT RESULT HI TO REM HI
          LD           A,[X]         ;  TEST FIELD LO TO ACC
          LD           B,#4
          SUBC         A,[B]         ;  SUBT DR LO FROM REM LO
          X            A,[X]         ;  RESULT LO TO REM LO
          LD           B,#0
          SBIT         0,[B]         ;  SET QUOTIENT BIT
DV616T:   DRSZ         CNTR          ;  DECREMENT AND TEST
          JP           DV616S        ;     CNTR FOR ZERO
          RET                        ;  RETURN FROM SUBROUTINE
```

DX1616—FAST 16 BY 16 DIVISION SUBROUTINE

```
          53 BYTES
          638 INSTRUCTION CYCLES AVERAGE
          678 INSTRUCTION CYCLES MAXIMUM
          DIVIDEND IN [1,0]         (DD)
          DIVISOR IN [5,4]          (DR)
          QUOTIENT IN [1,0]         (QUOT)
          REMAINDER IN [3,2]        (TEST FIELD)

DX1616:   LD          CNTR,#16     ;  LOAD CNTR WITH LENGTH
          LD          B,#5         ;     OF DIVIDEND FIELD
          LD          A,[B]        ;  REPLACE DIVISOR WITH
          XOR         A,#0FF       ;     1'S COMPLEMENT OF
          X           A,[B-]       ;     DIVISOR TO ALLOW
          LD          A,[B]        ;     OPTIONAL ADDITION OF
          XOR         A,#0FF       ;     DIVISOR'S COMPLEMENT
          X           A,[B-]       ;     IN MAIN PROG. LOOP
          LD          [B-],#0      ;  CLEAR
          LD          [B],#0       ;     TEST FIELD
DX616S:   LD          B,#0
DX616L:   RC
          LD          A,[B]
          ADC         A,[B]        ;  LEFT SHIFT DIVIDEND LO
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]        ;  LEFT SHIFT DIVIDEND HI
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]        ;  LEFT SHIFT TEST FIELD LO
          X           A,[B+]
          LD          A,[B]
          ADC         A,[B]        ;  LEFT SHIFT TEST FIELD HI
          X           A,[B+]
          SC
          LD          A,[B]        ;  DIVISORX (DRX) LO TO ACC
          LD          B,#2         ;     (1'S COMPLEMENT)
          ADC         A,[B]        ;  ADD REM LO TO DRX LO
          LD          B,#5
          LD          A,[B]        ;  DIVISORX (DRX) HI TO ACC
          LD          B,#3         ;     (1'S COMPLEMENT)
          ADC         A,[B]        ;  ADD REM HI TO DRX HI
          IFNC                     ;  TEST IF NO CARRY FROM
          JP          DX616T       ;     1'S COMPL.ADDITION
          X           A,[B+]       ;  RESULT TO REM HI
          LD          A,[B]        ;  DRX LO TO ACCUMULATOR
          LD          B,#2
          ADC         A,[B]        ;  ADD REM LO TO DRX LO
          X           A,[B]        ;  RESULT TO REM LO
          LD          B,#0
          SBIT        0,[B]        ;  SET QUOTIENT BIT
          DRSZ        CNTR         ;  DECREMENT AND TEST
          JP          DX616L       ;     CNTR FOR ZERO
          RET                      ;  RETURN FROM SUBROUTINE
DX616T:   DRSZ        CNTR         ;  DECREMENT AND TEST
          JMP         DX616S       ;     CNTR FOR ZERO
          RET                      ;  RETURN FROM SUBROUTINE
```

DV2815—FAST 28 BY 15 DIVISION SUBROUTINE

```
          WHERE THE DIVIDEND IS LESS THAN 2**28
          AND THE DIVISOR IS GREATER THAN 2**12 (4096) AND LESS THAN 2**15 (32768)
          43 BYTES
          640 INSTRUCTION CYCLES AVERAGE
          696 INSTRUCTION CYCLES MAXIMUM
          DIVIDEND IN [3,2,1,0]       (DD)
          DIVISOR IN [5,4]            (DR)
          QUOTIENT IN [1,0]           (QUOT)
          REMAINDER IN [3,2]          (TEST FIELD)

DV2815:   LD          CNTR,#16        ;  LOAD CNTR WITH LENGTH OF QUOTIENT FIELD
D2815S:   LD          B,#0
D2815L:   RC
          LD          A,[B]
          ADC         A,[B]           ;  LEFT SHIFT LOWER
          X           A,[B+]          ;     BYTE OF DIVIDEND
          LD          A,[B]
          ADC         A,[B]           ;  LEFT SHIFT NEXT HIGHER
          X           A,[B+]          ;     BYTE OF DIVIDEND
          LD          A,[B]
          ADC         A,[B]           ;  LEFT SHIFT NEXT HIGHER
          X           A,[B+]          ;     BYTE OF DIVIDEND
          LD          A,[B]
          ADC         A,[B]           ;  LEFT SHIFT UPPER
          X           A,[B-]          ;     BYTE OF DIVIDEND
          ***
          NOTE THAT WITH A 16 BIT DIVISOR (DIV 2816) SUBROUTINE, A TEST FOR A HIGH
          ORDER BIT SHIFTED OUT OF THE TEST FIELD WOULD BE NECESSARY AT THIS POINT.
          IFC
          JP          SUBTRMD         ;  SUBTRACT REM MINUS DR
          THE PRESENCE OF THIS CARRY WOULD REQUIRE THAT THE DIVISOR BE SUBTRACTED
          FROM THE REMAINDER AS SHOWN WITH THE DIV168*** SUBROUTINE.
          LD          A,[B]           ;  REM LOWER BYTE TO ACC
          SC                          ;  TEST SUBTRACT LOWER
          LD          B,#4            ;     BYTE OF DR FROM
          SUBC        A,[B]           ;     LOWER BYTE OF REM
          LD          B,#3            ;  TEST SUBTRACT UPPER
          LD          A,[B]           ;     BYTE OF DIVISOR
          LD          B,#5            ;     FROM UPPER BYTE
          SUBC        A,[B]           ;     OF REMAINDER
          IFNC                        ;  TEST IF BORROW
          JP          D2815T          ;     FROM SUBTRACTION
          LD          B,#3            ;  UPPER BYTE OF RESULT
          X           A,[B+]          ;     TO UPPER BYTE OF REM
          LD          A,[B]           ;  DR LOWER BYTE TO ACC
          LD          B,#2            ;  SUBTRACT LOWER BYTE
          X           A,[B]           ;     OF DIVISOR FROM
          SUBC        A,[B]           ;     LOWER BYTE OF
          X           A,[B]           ;     REMAINDER
          LD          B,#0
          SBIT        0,[B]           ;  SET QUOTIENT BIT
          DRSZ        CNTR            ;  DECREMENT AND TEST
          JMP         D2815L          ;     CNTR FOR ZERO
          RET                         ;  RETURN FROM SUBROUTINE
D2815T:   DRSZ        CNTR            ;  DECREMENT AND TEST
          JMP         D2815S          ;     CNTR FOR ZERO
          RET                         ;  RETURN FROM SUBROUTINE
```

DX3216—FAST 32 BY 16 DIVISION SUBROUTINE

```
            70 BYTES
            1510 INSTRUCTION CYCLES AVERAGE
            1590 INSTRUCTION CYCLES MAXIMUM
            DIVIDEND IN [3,2,1,0]       (DD)
            DIVISOR IN [7,6]            (DR)
            QUOTIENT IN [3,2,1,0]       (QUOT)
            REMAINDER IN [5,4]          (TEST FIELD)

DX3216:     LD          CNTR,#32        ; LOAD CNTR WITH LENGTH
            LD          B,#7            ;    OF DIVIDEND FIELD
            LD          A,[B]           ; REPLACE DIVISOR WITH
            XOR         A,#0FF          ;    1'S COMPLEMENT OF
            X           A,[B-]          ;    DIVISOR TO ALLOW
            LD          A,[B]           ;    OPTIONAL ADDITION OF
            XOR         A,#0FF          ;    DIVISOR'S COMPLEMENT
            X           A,[B-]          ; IN MAIN PROG. LOOP
            LD          [B-],#0         ; CLEAR
            LD          [B],#0          ;   TEST FIELD
DX326S:     LD          B,#0
DX326L:     RC
            LD          A,[B]
            ADC         A,[B]           ; LEFT SHIFT DIVIDEND LO
            X           A,[B+]
            LD          A,[B]
            ADC         A,[B]           ; LEFT SHIFT NEXT HIGHER
            X           A,[B+]          ;    DIVIDEND BYTE
            LD          A,[B]
            ADC         A,[B+]          ; LEFT SHIFT NEXT HIGHER
            X           A,[B+]          ;    DIVIDEND BYTE
            LD          A,[B]
            ADC         A,[B]           ; LEFT SHIFT DIVIDEND HI
            X           A,[B+]
            LD          A,[B]
            ADC         A,[B]           ; LEFT SHIFT TST FIELD LO
            X           A,[B+]
            LD          A,[B]
            ADC         A,[B]           ; LEFT SHIFT TST FIELD HI
            X           A,[B+]
            IFC                         ; **TEST IF BIT SHIFTED
            JP          DX326B          ; ** OUT OF TEST FIELD
            SC
            LD          A,[B]           ; DVSORX (DRX) LO TO ACC
            LD          B,#4            ;    (1'S COMPLEMENT)
            ADC         A,[B]           ; ADD REM LO TO DRX LO
            LD          B,#7
            LD          A,[B]           ; DVSORX (DRX) HI TO ACC
            LD          B,#5            ;    (1'S COMPLEMENT)
            ADC         A,[B]           ; ADD REM HI TO DRX HI
            IFNC                        ; TEST IF NO CARRY FROM
            JP          DX326T          ;    1'S COMPL. ADDITION
            X           A,[B+]          ; RESULT TO REM NI
            LD          A,[B]           ; DRX LO TO ACCUMULATOR
            LD          B,#4
DX326R:     ADC         A,[B]           ; ADD REM LO TO DRX LO
                                        ; ** ADD REM HI TO DRX HI
            X           A,[B]           ; RESULT TO REM LO
                                        ; ** RESULT TO REM HI
LD                      B,#0
            SBIT        0,[B]           ; SET QUOTIENT BIT
            DRSZ        CNTR            ; DECREMENT AND TEST
            JMP         DX326L          ;    CNTR FOR ZERO
            RET                         ; RETURN FROM SUBROUTINE
DX326T:     DRSZ        CNTR            ; DECREMENT AND TEST
            JMP         DX326S          ;    CNTR FOR ZERO
            RET                         ; RETURN FROM SUBROUTINE
DX326B:     LD          A,[B]           ; ** REM LO TO ACC
            LD          B,#6            ; ** B PTR TO DRX LO
            ADC         A,[B]           ; ** ADD DRX LO TO REM LO
            X           A,[B]           ; ** RESULT TO REM LO
            LD          B,#7            ; **
            LD          A,[B]           ; ** DRX HI TO ACC
            LD          B,#5            ; ** B PTR TO REM HI
            JP          DX36R           ; **
   **       THESE INSTRUCTIONS UNNECESSARY IF DIVISOR
            LESS THAN 2**15 (DX3215 SUBROUTINE)
```

MINIMAL GENERAL DIVISION SUBROUTINE (40 BYTES)

```
          ANY NUMBER OF BYTES IN DIVIDEND AND DIVISOR
          DV3224 SERVES AS EXAMPLE
          32 BY 24 DIVISION SUBROUTINE
                       --40 BYTES
                       --MINIMAL CODE
                       --3879 INSTRUCTION CYCLES AVERAGE
                       --4535 INSTRUCTION CYCLES MAXIMUM

          DIVIDEND IN [3,2,1,0]        (DD)
          DIVISOR IN [9,8,7]           (DR)
          QUOTIENT IN [3,2,1,0]        (QUOT)
          REMAINDER IN [6,5,4]         (TEST FIELD)

DV3224:   LD           CNTR,#32        ; LOAD CNTR WITH LENGTH
          LD           B,#6            ;    OF DIVIDEND FIELD
CLRLUP:   LD           [B-],#0         ; CLEAR TEST FIELD
          IFBNE        #3              ; TOP OF DIVIDEND FIELD
          JP           CLRLUP
DVSHFT:   RC
          LD           B,#0
SHFTLP:   LD           A,[B]
          ADC          A,[B]           ; LEFT SHIFT DIVIDEND
          X            A,[B+]          ;    AND TEST FIELD
          IFBNE        #7              ; BOTTOM OF DR FIELD
          JP           SHFTLP
          IFC                          ; TEST IF BIT SHIFTED
          JP           DVSUBT          ; *** OUT OF TEST FIELD
          SC                           ; RESET BORROW
          LD           X,#4
TSTLUP:   LD           A,[X+]          ; TEST SUBTRACT DIVISOR
          SUBC         A,[B]           ;    FROM TEST FIELD
          LD           A,[B+]          ; INCREMENT B POINTER
          IFBNE        #10             ; TOP OF DIVISOR + 1
          JP           TSTLUP
          IFNC                         ; TEST IF BORROW
          JP           DVTEST          ;    FROM SUBTRACTION
          LD           B,#7
DVSUBT:   LD           X,#4
SUBTLP:   LD           A,[X]           ; SUBTRACT DIVISOR
          SUBC         A,[B]           ;    FROM REMAINDER
          X            A,[X+]          ;    IN TEST FIELD
          LD           A,[B+]          ; INCREMENT B POINTER
          IFBNE        #10             ; TOP OF DIVISOR + 1
          JP           SUBTLP
          LD           B,#0
          SBIT         0,[B]           ; SET QUOTIENT BIT
DVTEST:   DRSZ         CNTR            ; DECREMENT AND TEST
          JP           DVSHFT          ;    CNTR FOR ZERO
          RET                          ; RETURN FROM SUBROUTINE
```

4.0 DECIMAL (PACKED BCD)/BINARY CONVERSION

Subroutines For Two Byte Conversion:

DECBIN — Decimal (Packed BCD) to Binary
— 24 Bytes ***
— 1030 Instruction Cycles

FDTOB — Fast Decimal (Packaged BCD) to Binary
— 76 Bytes
— 92 Instruction Cycles

BINDEC — Binary to Decimal (Packed BCD)
— 25 Bytes ***
— 856 Instruction Cycles

FBTOD — Fast Binary to Decimal (Packed BCD)
— 59 Bytes
— 334 Instruction Cycles

VFBTOD — Very Fast Binary to Decimal (Packed BCD)
— 189 Bytes
— 144 Instruction Cycles Average
— 208 Instruction Cycles Maximum

***These subroutines extendable to multiple byte conversion by simply changing parameters within subroutine as shown, with number of bytes in subroutine remaining constant.

DECBIN—Decimal (Packed BCD) to Binary

This 24 byte subroutine represents very minimal code for translating a packed BCD decimal number of any length to binary.

ALGORITHM:

The binary result is resident just below the packed BCD decimal number. During each cycle of the algorithm, the decimal operand and the binary result are shifted right one bit position, with the low order bit of the decimal operand shifting down into the high order bit position of the binary field. The residual decimal operand is then tested for a high order bit in each of its nibbles. A three is subtracted from each nibble in the BCD operand space that is found to contain a high order bit equal to one. (This process effectively right shifts the BCD operand one bit position, and then corrects the result to BCD format.) The entire cycle is then repeated, with the total number of cycles being equal to the number of bit positions in the decimal field.

16 Bit: Binary IN [1,0]
Packed BCD in [3, 2]

24 Bit: Binary in [2, 1, 0]
Packed BCD in [5, 4, 3]

32 Bit: Binary in [3, 2, 1, 0]
Packed BCD in [7, 6, 5, 4]

24 Bytes
1030 Instruction Cycles (16 Bit)

```
DECBIN:    LD       CNTR,#16       ; LOAD CNTR WITH NUMBER
                                   ;    OF BIT POSITIONS
                                   ;    IN BCD FIELD
                                   ; #16 FOR 16 BIT (2 BYTE)
                                   ; #'S 24/32 FOR 24/32 BIT
DB1:       LD       B,#3           ; #'S 5/7 FOR 24/32 BIT
           RC
DB2:       LD       A,[B]          ; PROGRAM LOOP TO
           RRC      A              ;    RIGHT SHIFT
           X        A,[B-]         ;    DECIMAL (BCD) AND
           IFBNE    #0F            ;    BINARY FIELDS.
           JP       DB2            ;    LOOP JUMP BACK
           LD       B,#3           ; #'S 5/7 FOR 24/32 BIT
           SC                      ; SET CARRY FOR SUBTRACT
DB3:       LD       A,[B]          ; TEST HIGH ORDER BITS
           IFBIT    7,[B]          ;    OF BCD NIBBLES, AND
           SUBC     A,#030         ;    SUBTRACT A THREE
           IFBIT    3,[B]          ;    FROM EACH NIBBLE IF
           SUBC     A,#3           ;    HIGH ORDER BIT OF
           X        A,[B-]         ;    NIBBLE IS A ONE
           IFBNE    #1             ; #'S 2/3 FOR 24/32 BIT
           JP       DB3            ; LOOP BACK FOR MORE BCD BYTES
           DRSZ     CNTR           ; DECREMENT AND TEST IF
           JP       DB1            ; CNTR EQUAL TO ZERO
           RET                     ; RETURN FROM SUBROUTINE
```

FDTOB—FAST DECIMAL (PACKED BCD) TO BINARY

BCD Format: Four Nibbles – W, X, Y, Z, with W = Hi Order Nibble
*** [1] = 16W + X
*** [0] = 16Y + Z

Algorithm: Binary Result is equal to 100(10W + X) + (10Y + Z)
BCD IN [1, 0]***
Temp in [2]
Binary in [4, 3]
76 Bytes
92 Instruction Cycles

```
FDTOB:  RC
        LD      B,#1
        LD      A,[B+]          ;  16W + X
        AND     A,#0F0          ;  EXTRACT 16W
        RRC     A               ;  8W
        X       A,[B]           ;  8W TO TEMP
        RRC     A               ;  4W
        RRC     A               ;  2W
        ADD     A,[B]           ;  2W + 8W = 10W
        X       A,[B-]          ;  10W TO TEMP
        LD      A,[B+]          ;  16W + X
        AND     A,#0F           ;  EXTRACT X
        ADC     A,[B]           ;  10W + X
        X       A,[B]           ;  10W + X TO TEMP
        LD      A,[B]
        ADC     A,[B]           ;  2.(10W + X)
        X       A,[B]           ;  2.(10W + X) TO TEMP
        ADC     A,[B]           ;  3.(10W + X)
        LD      B,#3            ;     = 16P + Q
        X       A,[B+]          ;  16P + Q TO [3]
        CLR     A
        IFC
        LD      A,#010          ;  16C TO A (C = CARRY)
        X       A,[B-]          ;  16C TO [4]
        LD      A,[B]           ;  16P + Q
        SWAP    A               ;  16Q + P
        X       A,[B]           ;  16Q + P TO [3]
        LD      A,[B+]          ;  16Q + P
        AND     A,#0F           ;  EXTRACT P
        ADD     A,[B]           ;  16C + P
        X       A,[B-]          ;  16C + P TO [4]**
        LD      A,[B]           ;  16Q + P
        AND     A,#0F0          ;  EXTRACT 16Q
        X       A,[B-]          ;  16Q TO [3]**
        LD      A,[B+]          ;  2.(10W + X)
        ADC     A,[B]           ;  2.(10W + X) + 16Q
```

```
X           A,[B+]          ;  2 BYTE 2.(10W + X)
CLR         A,[B-]          ;     ADD: + 48.**(10W + X)
ADC         A,[B]           ;  16C + P + NU C
X           A,[B-]          ;  50.(10W + X)
LD          A,[B]
ADC         A,[B]           ;  DOUBLE
X           A,[B+]          ;     50.(10W + X)
LD          A,[B]           ;     TO FORM
ADC         A,[B]           ;     100.(10W + X)
X           A,[B]           ;     IN [3,4]
LD          B,#0
LD          A,[B]           ;  16Y + Z
AND         A,#0F0          ;  EXTRACT 16Y
LD          B,#2
RRC         A               ;  8Y
X           A,[B]           ;  8Y TO TEMP
LD          A,[B]
RRC         A               ;  4Y
RRC         A               ;  2Y
ADC         A,[B]           ;  2Y + 8Y = 10Y
X           A,[B]           ;  10Y TO TEMP
LD          B,#0
LD          A,[B]           ;  16Y + Z
AND         A,#0F           ;  EXTRACT Z
LD          B,#2
ADD         A,[B]           ;  10Y + Z
LD          B, #3
ADC         A,[B]           ;  TWO BYTE ADD
X           A,[B+]          ;     100.(10W + X)
CLR         A               ;     + (10Y + Z)
ADC         A,[B]           ;     WITH BINARY
X           A,[B]           ;     RESULT TO [3,4]
RET
```

BINDEC—Binary to Decimal (Packed BCD)

This 25 byte subroutine represents very minimal code for translating a binary number of any length to packed BCD decimal.

ALGORITHM:

The packed BCD decimal result is resident just above the binary number. A sufficient number of bytes must be allowed for the BCD result. During each cycle of the algorithm the binary number is shifted left one bit position. The packed BCD decimal result is also shifted left one bit position, with the high order bit of the binary field being shifted up into the low order bit position of the BCD field. The shifted result in the BCD field is decimal corrected by using the DCOR instruction. Note that for addition an "ADD A, #066" instruction must be used in conjunction with the DCOR (Decimal Correct) instruction. The entire cycle is then repeated, with the total number of cycles being equal to the number of bit positions in the binary field.

16 Bit:	Binary in [1, 0] Packed BCD in [4, 3, 2]
24 Bit:	Binary in [2, 1, 0] Packed BCD in [6, 5, 4, 3]
32 Bit:	Binary in [3, 2, 1, 0] Packed BCD in [8, 7, 6, 5, 4]

25 Bytes
856 Instructions Cycles (16 Bit)

```
BINDEC:   LD        CNTR,#16        ;  LOAD CNTR WITH NUMBER OF BIT POSITIONS
                                    ;    IN BINARY FIELD
                                    ; #16 FOR 16 BIT (2 BYTE)
                                    ; #'S 24/32 FOR 24/32 BIT
          RC
          LD        B,#2            ; #'S 3/4 FOR 24/32 BIT
BD1:      LD        [B+],#0         ; CLEAR BCD FIELD
          IFBNE     #5              ; #'S 7/9 FOR 24/32 BIT
          JP        BD1             ; JUMP BACK FOR CLR LOOP
BD2:      LD        B,#0
BD3:      LD        A,[B]           ; PROGRAM LOOP TO
          ADC       A,[B]           ;    LEFT SHIFT
          X         A,[B+]          ;    BINARY FIELD
          IFBNE     #2              ; #'S 3/4 FOR 24/32 BIT
          JP        BD3             ; JUMP BACK FOR SHIFT LOOP1
BD4:      LD        A,[B]           ; PROGRAM LOOP TO
          ADD       A,#066          ;    LEFT SHIFT AND
          ADC       A,[B]           ;    DECIMAL CORRECT
          DCOR      A               ;    RESULT OF SHIFT
          X         A,[B+]          ;    IN BCD FIELD
          IFBNE     #5              ; #'S 7/9 FOR 24/32 BIT
          JP        BD4             ; JUMP BACK FOR SHIFT LOOP2
          DRSZ      CNTR            ; DECREMENT AND TEST IF
          JP        BD2             ;  CNTR EQUAL TO ZERO
          RET                       ; RETURN FROM SUBROUTINE
```

FBTOD—FAST BINARY TO DECIMAL (PACKED BCD)

Algorithm: This algorithm is based on the BINDEC algorithm, except that it is optimized for speed of execution.

Binary in [1, 0]
Packed BCD in [4, 3, 2]

59 Bytes
334 Instruction Cycles

```
FBTOD:      RC
            LD          B,#1
            LD          A,[B]
            SWAP        A               ;  REVERSE NIBBLES IN
            X           A,[B]           ;     UPPER BINARY BYTE
            LD          A,[B+]          ;  EXTRACT ORIGINAL UPPER
            AND         A,#0F           ;     NIBBLE OF HI BYTE
            IFGT        A,#9            ;  IF NIBBLE GREATER THAN
            ADD         A,#06           ;     NINE, THEN ADD SIX TO CORRECT BCD NIBBLE
            X           A,[B+]          ;  NIBBLE TO LOWER BCD BYTE
            LD          [B+],#0         ;  CLEAR UPPER BCD BYTES
            LD          [B],#0          ;  INITIALIZE CNTR TO COVER
            LD          CNTR,#4         ;     REMAINING HI NIBBLE (ORIGINALLY LO NIBBLE)
                                        ;  IN UPPER BINARY BYTE
FBD1:       LD          B,#1            ;  PROGRAM LOOP TO
            LD          A,[B]           ;     LEFT SHIFT A BIT
            ADC         A,[B]           ;     OUT OF UPPER BINARY
            X           A,[B+]          ;     BYTE INTO LOW ORDER
            LD          A,[B]           ;     BIT POSITION OF BCD
            ADD         A,#066          ;     FIELD, AS LOWER TWO
            ADC         A,[B]           ;     BYTES OF BCD FIELD
            DCOR        A               ;     ARE LEFT SHIFTED WITH
            X           A,[B+]          ;     THE LOWER BYTE BEING
            LD          A,[B]           ;     DECIMAL CORRECTED
            ADC         A,[B]           ;  MIDDLE BYTE OF BCD FIELD
            X           A,[B]           ;     NEED NOT BE DECIMAL CORRECTED, SINCE
                                        ;     MAX VALUE IS 2 (256)
            DRSZ        CNTR            ;  DECREMENT AND TEST IF
            JP          FBD1            ;     CNTR EQUAL TO ZERO
            LD          CNTR,#8         ;  INITIALIZE CNTR TO COVER
FBD2:       LD          B,#0            ;     LOWER BINARY BYTE
            LD          A,[B]           ;  PROGRAM LOOP TO
            ADC         A,[B]           ;     LEFT SHIFT A BIT
            X           A,[B]           ;     OUT OF LOWER BINARY
            LD          B,#2            ;     BYTE INTO LOW ORDER
            LD          A,[B]           ;     BIT POSITION OF BCD
            ADD         A,#066          ;     FIELD, AS BCD FIELD
            ADC         A,[B]           ;     IS LEFT SHIFTED WITH
            DCOR        A               ;     THE LOWER TWO BYTES
            X           A,[B+]          ;     OF THE FIELD BEING
            LD          A,[B]           ;     DECIMAL CORRECTED
            ADD         A,#066          ;  ADD (NOT ADC) HEX 66
            ADC         A,[B]           ;     TO SET UP "ADD" DCOR
            DCOR        A               ;  DECIMAL CORRECT MIDDLE
            X           A,[B+]          ;     BYTE OF BCD FIELD
            LD          A,[B]           ;  UPPER BYTE OF BCD FIELD
            ADC         A,[B]           ;     NEED NOT BE DECIMAL
            X           A,[B]           ;     CORRECTED, SINCE MAX
                                        ;     VALUE IS 6 (65535)
            DRSZ        CNTR            ;  DECREMENT AND TEST IF
            JP          FBD2            ;     CNTR EQUAL TO ZERO
            RET                         ;  RETURN FROM SUBROUTINE
```

VFBTOD—VERY FAST BINARY TO DECIMAL (PACKED BCD)

Algorithm: Decimal (Packed BCD) result is equal to summation in BCD of powers of two corresponding to 1's bits present in binary number.

Note that binary field (2 bytes) is initially one's complemented by program, in order to facilitate bypass branching when a tested bit in the binary field is found equal to zero.

Binary in [1, 0]
BCD in [4, 3, 2]

189 Bytes
144 Instruction Cycles Average
208 Instruction Cycles Maximum

```
VFBTOD:     RC
            LD          B,#0
            LD          A,[B]
            AND         A,#0F           ;  EXTRACT LO NIBBLE
            IFGT        A,#9            ;  TEST NIBBLE 9
            ADD         A,#6            ;  ADD 6 FOR CORRECTION
            LD          B,#2
            X           A,[B+]          ;  STORE IN LO BCD NIBBLE
            LD          [B+],#0         ;  CLEAR UPPER
            LD          [B],#0          ;     BCD NIBBLES
            LD          B,#1
            LD          A,[B]
            XOR         A,#0FF          ;  COMPLEMENT HI BYTE
            X           A,[B-]          ;     FOR REVERSE TESTING
            LD          A,[B]           ;     OF BINARY NUMBER
            XOR         A,#0FF          ;  COMPLEMENT LO BYTE
            X           A,[B]           ;     FOR REVERSE TESTING
            IFBIT       4,[B]           ;  TEST BINARY BIT 4
            JP          VFB1            ;     TO CONDITIONALLY
            LD          B,#2            ;     ADD BCD 16
            LD          A,#07C          ;  16 + 66
            ADC         A,[B]           ;  ADD BCD 16
            DCOR        A
            X           A,[B]
            LD          B,#0
VFB1:       IFBIT       5,[B]           ;  TEST BINARY BIT 5
            JP          VFB2            ;     TO CONDITIONALLY
            LD          B,#2            ;     ADD BCD 32
            LD          A,#098          ;  32 + 66
            ADC         A,[B]           ;  ADD BCD 32
            DCOR        A
            X           A,[B]
            LD          B,#0
VFB2:       IFBIT       6,[B]           ;  TEST BINARY BIT 6
            JP          VFB3            ;     TO CONDITIONALLY
            LD          B,#2            ;     ADD BCD 64
            LD          A,#0CA          ;  64 + 66
            ADC         A,[B]           ;  ADD BCD 64
            DCOR        A
            X           A,[B+]
            CLR         A
            ADC         A,[B]           ;  ADD CARRY
            X           A,[B]
            LD          B,#0
```

```
VFB3:   IFBIT   7,[B]           ; TEST BINARY BIT 7
        JP      VFB4            ;    TO CONDITIONALLY
        LD      B,#2            ;    ADD BCD 128
        LD      A,#08E          ; 28 + 66
        ADC     A,[B]           ; ADD BCD 28
        DCOR    A
        X       A,[B+]
        LD      A,#1
        ADC     A,[B]           ; ADD BCD 1
        X       A,[B]
VFB4:   LD      B,#1            ; HI BINARY BYTE
        IFBIT   0,[B]           ; TEST BINARY BIT 8
        JP      VFB5            ;    TO CONDITIONALLY
        LD      B,#2            ;    ADD BCD 256
        LD      A,#0BC          ; 56 + 66
        ADC     A,[B]           ; ADD BCD 56
        DCOR    A
        X       A,[B+]
        LD      A,#2
        ADC     A,[B]           ; ADD BCD 2
        X       A,[B]
        LD      B,#1
VFB5:   IFBIT   1,[B]           ; TEST BINARY BIT 9
        JP      VFB6            ;    TO CONDITIONALLY
        LD      B,#2            ;    ADD BCD 512
        LD      A,#078          ; 12 + 66
        ADC     A,[B]           ; ADD BCD 12
        DCOR    A
        X       A,[B+]
        LD      A,#06B          ; 5 + 66
        ADC     A,[B]           ; ADD BCD 5
        DCOR    A
        X       A,[B]
        LD      B,#1
VFB6:   IFBIT   2,[B]           ; TEST BINARY BIT 10
        JP      VFB7            ;    TO CONDITIONALLY
        LD      B,#2            ;    ADD BCD 1024
        LD      A,#08A          ; 24 + 66
        ADC     A,[B]           ; ADD BCD 24
        DCOR    A
        X       A,[B+]
        LD      A,#076          ; 10 + 66
        ADC     A,[B]           ; ADD BCD 10
        DCOR    A
        X       A,[B]
        LD      B,#1
VFB7:   IFBIT   3,[B]           ; TEST BINARY BIT 11
        JP      VFB8            ;    TO CONDITIONALLY
        LD      B,#2            ;    ADD BCD 2048
        LD      A,#0AE          ; 48 + 66
        ADC     A,[B]           ; ADD BCD 48
        DCOR    A
        X       A,[B+]
        LD      A,#086          ; 20 + 66
        ADC     A,[B]           ; ADD BCD 20
        DCOR    A
        X       A,[B]
        LD      B,#1
```

```
VFB8:     IFBIT      4,[B]          ; TEST BINARY BIT 12
          JP         VFB9           ;    TO CONDITIONALLY
          LD         B,#2           ;    ADD BCD 4096
          LD         A,#0FC         ; 96 + 66
          ADC        A,[B]          ; ADD BCD 96
          DCOR       A
          X          A,[B+]
          LD         A,#0A6         ; 40 + 66
          ADC        A,[B]          ; ADD BCD 40
          DCOR       A
          X          A,[B]
          LD         B,#1
VFB9:     IFBIT      5,[B]          ; TEST BINARY BIT 13
          JP         VFB10          ;    TO CONDITIONALLY
          LD         B,#2           ;    ADD BCD 8192
          LD         A,#0F8         ; 92 + 66
          ADC        A,[B]          ; ADD BCD 92
          DCOR       A
          X          A,[B+]
          LD         A,#0E7         ; 81 + 66
          ADC        A,[B]          ; ADD BCD 81
          DCOR       A
          X          A,[B]
          CLR        A
          ADC        A,[B]          ; ADD CARRY
          X          A,[B]
          LD         B,#1
VFB10:    IFBIT      6,[B]          ; TEST BINARY BIT 14
          JP         VFB11          ;    TO CONDITIONALLY
          LD         B,#2           ;    ADD BCD 16384
          LD         A,#0EA         ; 84 + 66
          ADC        A,[B]          ; ADD BCD 84
          DCOR       A
          X          A,[B+]
          LD         A,#0C9         ; 63 + 66
          ADC        A,[B]          ; ADD BCD 63
          DCOR       A
          X          A,[B+]
          LD         A,#1
          ADC        A,[B]          ; ADD BCD 1
          X          A,[B]
          LD         B,#1
VFB11:    IFBIT      7,[B]          ; TEST BINARY BIT 15
          RET                       ;    TO CONDITIONALLY
          LD         B,#2           ;    ADD BCD 32768
          LD         A,#0CE         ; 68 + 66
          ADC        A,[B]          ; ADD BCD 68
          DCOR       A
          X          A,[B+]
          LD         A,#08D         ; 27 + 66
          ADC        A,[B]          ; ADD BCD 27
          DCOR       A
          X          A,[B+]
          LD         A,#3
          ADC        A,[B]          ; ADD BCD 3
          X          A,[B]
          RET
```

Pulse Width Modulation A/D Conversion Techniques with COP800 Family Microcontrollers

National Semiconductor
Application Note 607
Kevin Daugherty

1.0 BASIC TECHNIQUE

This application note describes a technique for creating an analog to digital converter using a microcontroller with other low cost components. Many applications do not require the speed associated with a dedicated hardware A/D converter and it is worth evaluating a more cost effective approach.

With a high speed CMOS microcontroller an eight bit A/D can be implemented that converts in approximately 10 ms. This method is based on the fact that if a repetitive waveform is applied to an RC network, the capacitor will charge to the average voltage, provided that the RC time constant is much larger than the pulse widths. The basic equation for computing the analog to digital result is:

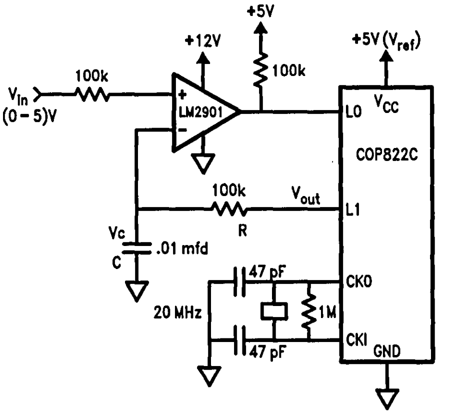
$$V_{in} = V_{ref}[T_{on}/(T_{on} + T_{off})] \qquad (1)$$

With this equation it is necessary to precisely measure several time periods within both the T_{on} and T_{off} in order to achieve the desired resolution. Additionally, the waveform would have to be gradually adjusted to allow for the large RC time constant to settle out. This results in a relatively long conversion cycle. Modifying the equation and technique slightly, significantly speeds up the process. This technique works by averaging several pulses over a fixed period of time and is based on the following equation:

$$V_{in} = V_{ref}[\text{Sum of } T_{on}/(\text{Sum of } (T_{on} + T_{off}))] \qquad (2)$$

2.0 IMPLEMENTATION

Figure 1 describes the basic circuit schematic that uses a National Semiconductor COP822C microcontroller, a low cost LM2901 comparator, three 100k resistors, and a 0.01 mfd film capacitor. The CMOS COP822C microcontroller provides a squarewave signal with logic levels very close to GND and V_{CC}. This generates a small ramp voltage on the capacitor for the LM2901 quad comparator input.



TL/DD/10407-1

FIGURE 1. Basic Circuit

To minimize error, a tradeoff must be made when selecting the resistor. The microcontroller output (L1) should have a large resistor to minimize the output switching offset (V_{OS}), and the comparator should have a small resistor due to error caused by I_{bos} (input bias offset current).

Once the resistor is determined, the capacitor should be chosen so that the RC time constant is large enough to provide a small incremental voltage ramp. This design has a sample time of 20 μs and has a 1 ms time constant with a 0.01 mfd film type capacitor which has low leakage current to prevent errors. Since a 100k resistor is used in the RC network for one comparator input, another 100k resistor is required for the V_{in} input to balance the offset voltage caused by the comparator I_b (input bias current).

Figure 2 illustrates the relationship between the microcontroller squarewave output and the capacitor charge and discharge. Every 20 μs the comparator is sampled. If the capacitor voltage (V_c) is below V_{in} the RC network will receive a positive pulse. The inverse is true if V_c is above V_{in} at sample time. Note that with this approach, the PWM waveform is broken up into several small pulses over a fixed period instead of having a single pulse represent the duty cycle; thus a relatively small RC time constant can be used.

Mathematical Analysis:

let n = total number of T_{on} pulses and

m = total number of T_{off} pulses

then $V_c(t) = V_c + n[\,(V_{out} - V_c)\,(1 - e - t/RC)] - m[\,(V_c - V_o)\,(1 - e - t/RC)]$

let $V_c = V_{in}$ at start of conversion and

$K = (1 - e - t/RC)$

then $V_{in} = V_{in} + K_nV_{out} - K_nV_{in} - K_mV_{in} + K_mV_O$

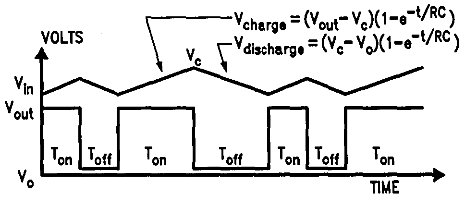
$0 = K_nV_{out} + K_mV_o - KV_{in}\,(n + m)$

let $V_{out} = V_{ref} - V_{os}$

solving for V_{in}:

$$V_{in} = nV_{ref}/(n + m) - (nV_{os} - mV_o)\,(1/(n + m) \qquad (3)$$

Note that the RC value drops out of the equation and therefore is not an error factor.



TL/DD/10407-2

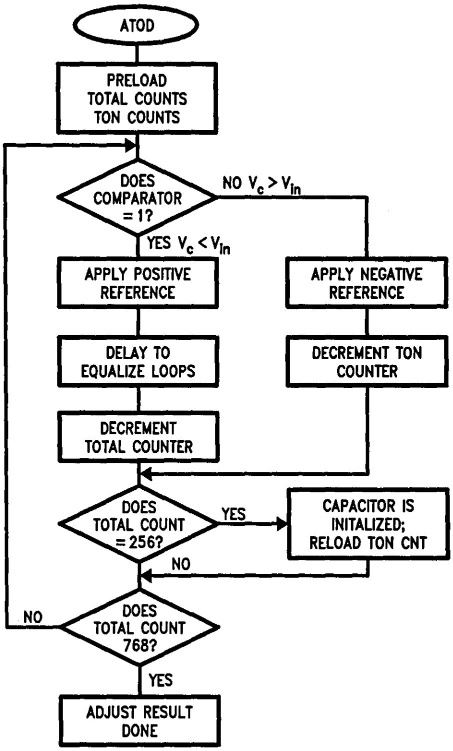
FIGURE 2. PWM Signal

3.0 SOFTWARE DESCRIPTION

Referring to the PWM flow chart in *Figure 3*, the software counters (Total and T_{on}) are initially loaded with the maximum value. Then the microcontroller samples the comparator output and determines whether to drive the RC with a "1" or "0" pulse. Each time the RC receives a "0" pulse, the T_{on} counter is decremented, and each time a loop is completed the Total counter is decremented. For this technique to work accurately, it is important that the high and low loops be exactly the same. This is necessary because the pulses are time weighted and averaged over a period of 512 samples.

Before a conversion is started, it is critical for the capacitor to be initialized close to V_{in} otherwise an error will result that is equal to the number of high or low pulses required to restore the capacitor back to equal V_{in}. The program achieves this by doing a short conversion of 256 samples that provides enough time to fully charge the capacitor close to V_{in}. Then the T_{on} software counter is reloaded and the actual conversion cycle is started. When the Total counter has been decremented from the maximum count to zero, the conversion is done. The value left in the T_{on} counter represents the result, and after it is loaded in the accumulator it is right shifted with the carry bit to adjust for 8-bit accuracy.

The L output port can TRI-STATE® to provide a shorter capacitor initialization time and when interrupting the A/D program. At the start of the interrupt routine the output state can be saved then restored back to the previous state when the interrupt has been serviced. This way the capacitor will not decay significantly and cause error when the A/D program resumes.



TL/DD/10407–3

FIGURE 3. PWM A/D Flow Chart

The program listed below in *Figure 4* will work on any COP800 microcontroller (i.e. COP820, COP840, COP888).

```
       LD A,#03      ;LOAD A FOR INITIALIZATION
       LD 0F0,#0FF   ;PRELOAD TOTAL COUNTS
       LD 0F1,#3     ;MULTIPLIER FOR 2X256=512 COUNTS PLUS 256 FOR INIT.
       LD 0F2,#0FF   ;PRELOAD Ton
       LD 0F3,#2     ;MULTIPLIER FOR 2X256=512 POSSIBLE Ton COUNTS
       LD 0FE,#0D0   ;LOAD B FOR L data REG
       RC            ;CLEAR CARRY FOR RESULT ADJUST
       LD 0D0,#01    ;L PORT DATA REG, L0=WEAK PULL UP, L1=HIGH
       LD 0D1,#02    ;L PORT CONFIG REG, L0=INPUT, L1=OUTPUT
LOOP:  IFBIT 0,0D2   ;TEST COMPARATOR OUTPUT
       JMP HIGH      ;JUMP IF L0=1
       NOP           ;EQUALIZE TIME FOR SETTING AND RESETTING
       RBIT 1,[B]    ;DRIVE L1 LOW
       DRSZ 0F2      ;DECREMENT Ton WHEN DRIVING LOW
       JMP COUNT
       DRSZ 0F3      ;DECREMENT Ton MULTIPLIER IF BEYOND 256 COUNTS
       JMP COUNT
HIGH:  SBIT 1,[B]    ;DRIVE L1 HIGH
       NOP
       NOP
       NOP
       NOP
       NOP
       NOP           ;EQUALIZE HIGH AND LOW LOOPS
COUNT: DRSZ 0F0      ;DECREMENT TOTAL COUNTS
       JMP LOOP
       IFEQ A,0F1    ;INITIALIZE Ton FOR FIRST TIME THRU LOOP
       LD 0F2,#0FF   ;BY RELOADING Ton WITH FF
       DRSZ 0F1      ;DECREMENT MULTIPLIER
       JMP LOOP
       LD A,0F2      ;LOAD A WITH Ton
       IFBIT 1,0F3   ;CHECK Ton FOR > 256 COUNTS
       SC            ;SET CARRY IF >256
       RRC A         ;ADJUST RESULT FOR A 8 BIT ACCURACY
       X, A,00       ;STORE RESULT IN RAM LOCATION 00
       LD 0D0,#00    ;TRISTATE L PORT TO PREVENT CAP FROM DECAYING
       LD 0D1,#00
.END
```

FIGURE 4. COP800 PWM A/D Program Listing

4.0 ACCURACY AND CIRCUIT CONSIDERATIONS

The basic circuit will provide 8 bits ±1 LSB accuracy depending on the choice of comparator, and passive components. With this type of design several tradeoffs and error sources should be considered. First of all, conversion equation 2 assumes that the microcontroller output switches exactly to GND and V_{CC} (or V_{ref}). The COP822C will typically switch between 10 mV and 20 mV from GND and V_{CC} with a light load. This will cause an error equal to the offset voltage times the duty cycle (equ. 3). Fortunately, the offsets tend to cancel each other at mid range voltages. At near GND and V_{CC} input voltages the offsets are minimal due to the very small voltage drop across the resistor. If the error is undesirable, the offset voltage can be reduced by paralleling outputs with the same levels together, or by using a CMOS buffer such as a 74HC04 to drive the RC network (see *Figure 5* for suggested circuits).

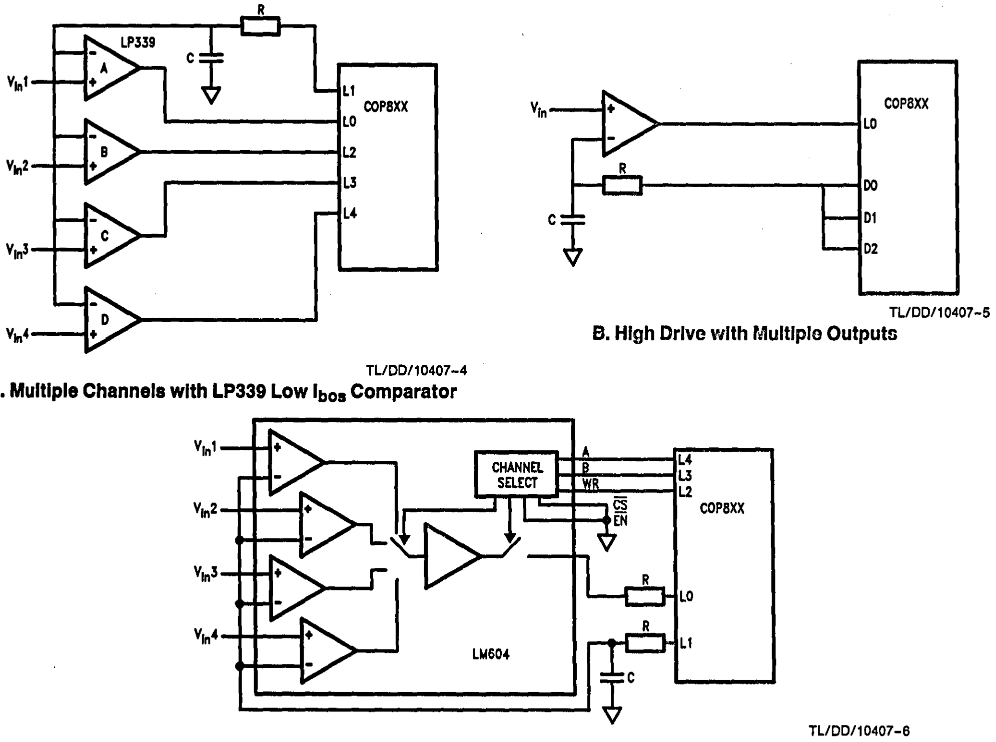
Another possible source of error is with the LM2901 worst case input bias offset current of 200 nA over temperature. This will cause an error equal to R_{in} X I_{bos}, which equals 20 mV with a 100k resistor. Either the resistor or the I_{bos} can be reduced to improve the error. If the resistor is reduced then the L port offset voltages will increase so the preferred approach is to select a comparator with lower I_{bos} such as the LP339 which has an I_{bos} of only ±15 nA. The comparator V_{os} may also introduce error. The LM2901 V_{os} is ±9 mV, the LP339 V_{os} is only ±5 mV. An added benefit of using the LP339 is that since the I_{bos} is so small, the resistor for the RC network can be larger. In addition, one RC network could be used for several comparator input channels (refer to *Figure 5a*).

By using the LM604 *(Figure 5c)* the basic software can be easily extended for converting several channels. This will only require a control line to be selected before a conversion is started. Since the LM604 needs to be powered from a higher voltage than the input voltage range, the output voltage will also be higher than the microcontroller supply. This requires a current limiting resistor to be used in series between the LM604 output and the COP8XX. Note that two or more LM604's can be paralleled for providing several more A/D channels by utilizing the EN control input that can TRI-STATE the LM604 output when high.

Depending on the speed and accuracy requirements, the total number of counts used in the conversion can be changed. Increasing the counts will give more accuracy with the practical limit of about 9–10 bits. With increased resolution, the capacitor ramp voltage per sample time should be decreased so that the capacitor can be initialized to within 1 LSB prior to conversion. This can be done by either increasing the RC time constant, or by using an initialization routine with a shorter sample time. The conversion time will depend on the total counts and the microcontroller oscillator frequency as described below:

$$T_{con} = \text{Total counts} \times (20\ \text{cycles}) \times (\text{instruction cycle time})$$

Another factor to consider is when a non-ratiometric conversion is required, the reference voltage must have the tolerance to match the desired accuracy.



TL/DD/10407-4

A. Multiple Channels with LP339 Low I_{bos} Comparator

TL/DD/10407-5

B. High Drive with Multiple Outputs

TL/DD/10407-6

C. Four Channel A/D with LM604 MUX-Amplifier

FIGURE 5. Suggested Circuits

5.0 CONCLUSION

The PWM A/D technique described in this application note provides a relatively fast discrete implementation with substantial cost savings compared to a dedicated hardware A/D. Minimal microcontroller I/O and software is required to interface with a comparator and RC network. Depending on the application requirements, the designer can tailor the basic 8-bit A/D a number of ways. By varying the total software counts, the desired speed and resolution can be adjusted. The number of A/D channels will determine the number of comparators used. In chosing the comparator, it is recommended that the designer refer to the data sheets and match the I_{bos} and V_{os} to the desired accuracy.

When other than a 1 μs instruction cycle is used, the RC value should be scaled to provide a peak-peak ramp voltage on the capacitor of < 1 LSB of the desired resolution. Any type of microcontroller oscillator can be used (i.e., RC, ceramic resonator, appreciably within one conversion cycle.

Section 4
HPC Family

Section 4 Contents

National Semiconductor

The 16-Bit HPC™ Family: Optimized for Performance

Key Features

- World's first 16-bit CMOS microcontroller
- World's fastest CMOS microcontroller
- 67 ns instruction-cycle time at 30 MHz
- Full 16-bit architecture and implementation
- 64 kbyte address space
- High code efficiency with single-byte, multiple-function instructions
- 16 x 16-bit multiply, 32 x 16-bit divide
- Eight vectored interrupt sources
- Watchdog logic monitors
- 16-bit timer/counters
- Up to 52 general-purpose high-speed I/O lines
- On-chip ROM to 16 kbytes
- On-chip RAM to 512 bytes
- On-chip peripherals
  - — DMA
  - — HDLC
  - — Timers
  - — Input-capture registers
  - — A/D converter
  - — UART
  - — User-programmable memory
  - — High speed SRAM
- M²CMOS fabrication
- MICROWIRE/PLUS™ serial interface
- ROMless versions available
- Wide operating voltage range: +4.5V to +5.5V
- Military temp range available (−55°C to +125°C)
- MIL-STD-883C versions available
- 68-pin PGA, PLCC, LDCC packages and 84-pin Tape-Pak®

National's High Performance Controller (HPC) family is not only the world's first 16-bit CMOS microcontroller family, but also the world's fastest.

Currently operating at a clock rate of 30 MHz, the HPC fabricated in scalable M²CMOS™, allowing die-shrinks ultimately, to submicron levels. Meaning the HPC will be operating at much higher frequencies in the future.

The HPC is designed for high-performance applications. With its 67 ns instruction cycle and its 16 x 16-bit multiply and 32 x 16-bit divide, the HPC is appropriate for compute-intensive environments that used to be the sole domain of the microprocessor.

The HPC is ideal, for example, for signal conditioning applications. The HPC's high throughput helps eliminate external components from typical signal processing/control circuits, and allows key parts of the application to be implemented in software rather than hardware.

This not only reduces system cost and development time, but also increases the flexibility and market life of the product.

At the same time, because the HPC has a control-oriented architecture, important functions are still implemented in hardware, providing critical performance advantages unavailable in a pure-software solution, such as a general microprocessor-based design.

It is this powerful performance capability that, when combined with the wide range of peripheral functions that are available (such as UARTs, A/D converters, and HDLC), make the HPC a true systems solution on a chip.

The Powerful HPC Core

The HPC is an "application-specific" microcontroller.

Based on a common, high-performance CPU "core", each HPC family member can be "customized" to meet the exact needs of a particular application.

The core, based on a microprocessor-like von Neumann architecture, contains seven key functional elements:

1. Arithmetic Logic Unit (ALU)
2. 6 working registers
3. 8 interrupts
4. 3 timers
5. Control logic
6. Watchdog circuitry
7. MICROWIRE/PLUS interface

The internal data paths, registers, timers, and ALU are all 16 bits wide.

So the HPC can directly address up to 64 kbytes of "external" memory.

The external data bus, however, is dynamically configurable as 8 or 16 bits, allowing it to efficiently interface with a variety of peripheral devices.

Flexible Peripheral Support

The HPC core can support a full range of peripheral functions:

- High-level Data Link Control (HDLC) for ISO-standard data communications

Flexible Peripheral Support (Continued)

- Universal Asynchronous Receiver/Transmitters (UARTs) for full-duplex, 300/1200/2400/9600-baud serial communications
- High-Speed Outputs and Pulse-Width Modulated (PWM) timers for efficient external interfaces
- User-programmable memory
- Analog-to-Digital (A/D) converters for interfacing "real-world" inputs

Plus:

- Up to 64 kbytes of direct-addressable memory
- Up to 52 I/O ports on a 68-pin package

Efficient Instruction Set

The HPC family achieves much of its performance through its unique, highly optimized instruction set. Unlike the instruction set of a typical microprocessor, the HPC instruction set is designed for maximum code efficiency. Because ROM-space is necessarily limited on a single-chip solution, programs must be compact and economical.

The HPC instruction set supports nine addressing modes, like a high-performance 16-bit microprocessor. And each instruction in the set is designed to execute a number of individual functions, so the same operations can be executed with tighter code.

As a result, the typical HPC instruction cycle is only 67 ns at 30 MHz. And the typical HPC 16-bit multiply or divide takes less than 4 μs.

To achieve the same level of performance in other 16-bit and high-end 8-bit microcontrollers, as indicated by recent benchmark studies, would require up to *two times the memory space* as the HPC.

Low Power Operation

The HPC uses power as efficiently as it uses memory space.

The HPC draws only 47 mA of current at 20 MHz. And its even less at lower clock rates.

In addition, the HPC has two software-selectable power-down modes:

1. IDLE, which stops all operations except for the oscillator and one timer, thereby maintaining all RAM, registers, and I/O in a static state.

2. HALT, which stops all operations including the oscillator and timers, but holds RAM, registers, and I/O stable.

Key Applications

- Signal conditioning/processing/control
- Automotive systems
- Data processing
- Telecommunications
- Military
- Embedded controllers
- Medical
- Factory automation
- Industrial control
- Compute-intensive environments
- High-end control
- Tape and disk drives
- Security systems
- Laser printers
- SCSI control

High Level Language Support

A C compiler is already available for software development on standard platforms: the IBM PC running DOS or UNIX® or the DEC™ VAX™ running VMS™ or UNIX.

With powerful tools such as these, the HPC can be quickly and efficiently programmed for any high-performance application.

HPC Family of Microcontrollers

			Memory		Features						
Commercial Temp Version 0°C to +70°C	Industrial Temp Version −40°C to +85°C	Military Temp Version −55°C to +125°C	ROM (Bytes)	RAM (Bytes)	I/O		Interrupt	Stack	Timer Base Counters	Size (Pins)	Other*
					I/O Pins	Serial I/O					
HPC46003	HPC36003	HPC16003	ROMless	256	52	YES	8 Sources	In RAM	8	68	4 ICR's
HPC46004	HPC36004	HPC16004	ROMless	512	52	YES	8 Sources	In RAM	8	68	4 ICR's
HPC46064	HPC36064	HPC16064	16.0k	512	52	YES	8 Sources	In RAM	8		4 ICR's
HPC46083	HPC36083	HPC16083	8.0k	256	52	YES	8 Sources	In RAM	8	68	4 ICR's
HPC46104	HPC36104	HPC16104	ROMless	512	52	YES	8 Sources	In RAM	8		4 ICR's & 8 CH A/D
HPC46164	HPC36164	HPC16164	16.0k	512	52	YES	8 Sources	In RAM	8	68	4 ICR's & 8 CH A/D
HPC46400	HPC36400	HPC16400	N/A	256	56	YES	8 Sources	In RAM	4	68	HDLC & DMA

*ICR = Input Capture Registers

HDLC = High-Level Data Link Control

PEARL = Port Expanded and Recreation Logic

National Semiconductor

PRELIMINARY

HPC16083/HPC26083/HPC36083/HPC46083/ HPC16003/HPC26003/HPC36003/HPC46003 High-Performance microControllers

General Description

The HPC16083 and HPC16003 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC16083 has 8k bytes of on-chip ROM. The HPC16003 has no on-chip ROM and is intended for use with external direct memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

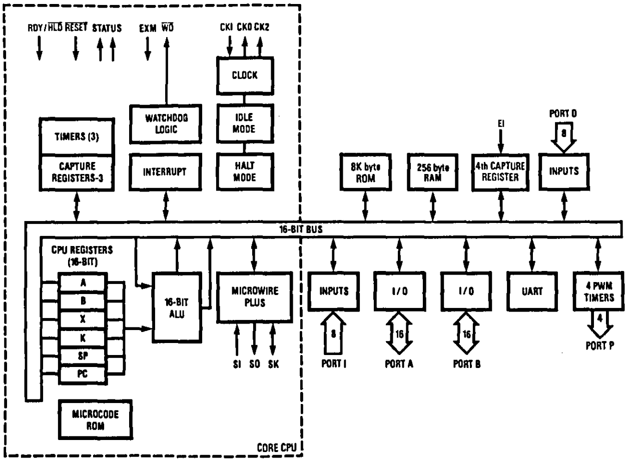
The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC16083" is used throughout this datasheet to refer to the HPC16083 and HPC16003 devices unless otherwise specified.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LCC, LDCC, PGA and 84-Pin TapePak® packages.

Features

- HPC family—core features:
  - 16-bit architecture, both byte and word
  - 16-bit data bus, ALU, and registers
  - 64k bytes of external direct memory addressing
  - FAST—200 ns for fastest instruction when using 20.0 MHz clock, 134 ns at 30 MHz
  - High code efficiency—most instructions are single byte
  - 16 x 16 multiply and 32 x 16 divide
  - Eight vectored interrupt sources
  - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
  - MICROWIRE/PLUS serial I/O interface
  - CMOS—very low power with two power save modes: IDLE and HALT
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 8k bytes of ROM, 256 bytes of RAM on chip
- ROMless version available (HPC16003)
- Commercial (0°C to +70°C), industrial (−40°C to +85°C), automotive (−40°C to +105°C) and military (−55°C to +125°C) temperature ranges

Block Diagram (HPC16083 with 8k ROM shown)



TL/DD/8801-1

20 MHz Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 20 MHz (Note 1)		47	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		10	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 20 MHz, (Note 1)		3.0	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		1	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		200	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		50	μA
INPUT VOLTAGE LEVELS $\overline{\text{RESET}}$, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI1}	Input Leakage Current			±1	μA
I_{LI2}	Input Leakage Current RDY/HLD, EXUI		−3	−50	μA
I_{LI3}	Input Leakage Current B12		0.5	7	mA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA (Note 2)	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA (Note 2)		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	I_{OH} = −6 mA	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{\text{RESET}}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}, CKI driven to V_{IH1} and V_{IL1}, with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

20 MHz

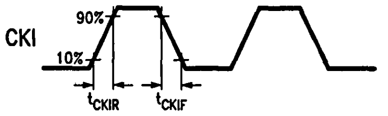
AC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	20	MHz
$t_{C1} = 1/f_C$	Clock Period	50	500	ns
t_{CKIR} (Note 3)	CKI Rise Time		7	ns
t_{CKIF} (Note 3)	CKI Fall Tiime		7	ns
$[t_{CKIH}/(t_{CKIH} + t_{CKIL})]100$	Duty Cycle	45	55	%
$t_C = 2/f_C$	Timing Cycle	100		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns
t_{DC1C2R} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		45	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		45	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	18		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	20		ns
$t_{WAIT} = t_C$	Wait State Period	100		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.052	MHz
$t_{XIN} = t_c$	Pulse Width for Timer Inputs	100		ns
$f_{XOUT} = f_C/16$	Timer Output Frequency		1.25	MHz
$f_{MW} = f_C/19$	External MICROWIRE/PLUS Clock Input Frequency		1.052	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz

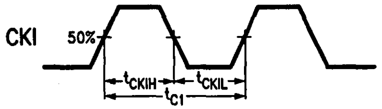
CKI Input Signal Characteristics

Rise/Fall Time



TL/DD/8801-35

Duty Cycle



TL/DD/8801-36

20 MHz
Read Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{\text{RD}}$ Falling Edge	20		ns
$t_{RW} = \frac{1}{2} t_C + WS - 10$	$\overline{\text{RD}}$ Pulse Width	140		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{\text{RD}}$	0	60	ns
$t_{ACC} = t_C + WS - 55$ (Note 2)	Address Valid to Input Data Valid		145	ns
$t_{RD} = \frac{1}{2} t_C + WS - 65$	$\overline{\text{RD}}$ Falling Edge to Input Data Valid		85	ns
$t_{RDA} = t_C - 5$	$\overline{\text{RD}}$ Rising Edge to Address Valid	95		ns

Write Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{\text{WR}}$ Falling Edge	45		ns
$t_{WW} = \frac{3}{4} t_c + WS - 15$	$\overline{\text{WR}}$ Pulse Width	160		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{\text{WR}}$	20		ns
$t_V = \frac{1}{2} t_C + WS - 5$	Data Valid before Rising Edge of $\overline{\text{WR}}$	145		ns

Note: Bus Output (Port A) $C_L = 100$ pF, CK2 Output $C_L = 50$ pF, other Outputs $C_L = 80$ pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurements will be skewed.

Note: WS = t_{WAIT} * number of pre-programmed wait states. Minimum and maximum values are calculated from maximum operating frequency with one (1) wait state pre-programmed.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not directly tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a high confidence level.

Note 3: This is guaranteed by design and not tested.

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns
$t_{RWP} = t_C$	RDY Pulse Width	100		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{\text{HLD}}$ to Rising Edge of ALE	115		ns
$t_{HWP} = t_C + 10$	$\overline{\text{HLD}}$ Pulse Width	110		ns
$t_{HAD} = \frac{3}{4} t_C + 85$	Rising Edge on $\overline{\text{HLD}}$ to Rising Edge on $\overline{\text{HLDA}}$		160	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{\text{HLD}}$ to Falling Edge on $\overline{\text{HLDA}}$		200*	ns
$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after Falling Edge on $\overline{\text{HLDA}}$		116†	ns
$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable before Rising Edge of $\overline{\text{HLDA}}$	116†		ns

***Note:** t_{HAE} may be as long as ($3t_C + 4ws + 72t_C + 90$) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value tested is for the optimal case.

†**Note:** Due to emulation restrictions—actual limits will be better.

MICROWIRE/PLUS Timing

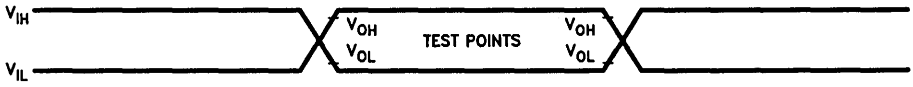
Symbol	Parameter	Min	Max	Units
t_{UWS} Master Slave	MICROWIRE Setup Time	 100 20		ns
t_{UWH} Master Slave	MICROWIRE Hold Time	 20 50		ns
t_{UWV} Master Slave	MICROWIRE Output Valid Time		 50 150	ns

UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{URD}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{URD}$	10		ns
t_{RPW}	$\overline{URD}$ Pulse Width	100		ns
t_{OE}	$\overline{URD}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{URD}$ to Output Data Invalid (Note 4)	5	35	ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Rising Edge of $\overline{URD}$		70	ns
t_{WDW}	$\overline{UWR}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{UWR}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{UWR}$	15		ns
t_A	$\overline{WRRDY}$ Delay from Rising Edge of $\overline{UWR}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

Note 4: Guaranteed by design.



TL/DD/8801-38

Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

Input and Output for AC Tests

30 MHZ

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 30.0 MHz (Note 1)		65	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		10	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 30.0 MHz, (Note 1)		5	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		1	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		200	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		50	μA
INPUT VOLTAGE LEVELS $\overline{\text{RESET}}$, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI1}	Input Leakage Current			±1	μA
I_{LI2}	Input Leakage Current RDY/HLD, EXUI		−3	−50	μA
I_{LI3}	Input Leakage Current B12		0.5	7	mA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA (Note 2)	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA (Note 2)		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	I_{OH} = −6 mA	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{\text{RESET}}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}, CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

30 MHZ

AC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46083/HPC46003, −40°C to +85°C for HPC36083/HPC36003, −40°C to +105°C for HPC26083/HPC26003, −55°C to +125°C for HPC16083/HPC16003

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	30	MHz
$t_{C1} = 1/f_C$	Clock Period	33	500	ns
t_{CKIR} (Note 3)	CKI Rise Time		7	ns
t_{CKIF} (Note 3)	CKI Fall Tiime		7	ns
$[t_{CKIH}/(t_{CKIH} + t_{CKIL})]100$	Duty Cycle	45	55	%
$t_C = 2/f_C$	Timing Cycle	66		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	24		ns
t_{DC1C2R}	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F}	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		37	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		37	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	9		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	11		ns
$t_{WAIT} = t_C = WS$	Wait State Period	66		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.579	MHz
$t_{XIN} = t_C$	Pulse Width for Timer Inputs	66		ns
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.875	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		3.75	MHz

Read Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{RD}$ Falling Edge	12		ns
$t_{RW} = \frac{1}{2} t_C + WS - 14$	$\overline{RD}$ Pulse Width	85		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{RD}$	0	35	ns
$t_{ACC} = t_C + WS - 32$ (Note 2)	Address Valid to Input Data Valid		100	ns
$t_{RD} = \frac{1}{2} t_C + WS - 39$	$\overline{RD}$ Falling Edge to Input Data Valid		60	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	61		ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 pF, other Outputs C_L = 80 pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurements will be skewed.

Note: WS = t_{WAIT} * number of pre-programmed wait states. Minimum and maximum values are calculated from maximum operating frequency with one (1) wait state pre-programmed.

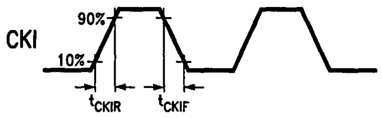
Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not directly tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a high confidence level.

Note 3: This is guaranteed by design and not tested.

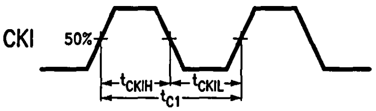
CKI Input Signal Characteristics

Rise/Fall Time



TL/DD/8801-35

Duty Cycle



TL/DD/8801-36

30 MHz

Write Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{WR}$ Falling Edge	28		ns
$t_{WW} = \frac{3}{4} t_c + WS - 15$	$\overline{WR}$ Pulse Width	101		ns
$t_{HW} = \frac{1}{4} t_C - 10$	Data Hold after Rising Edge of $\overline{WR}$	7		ns
$t_V = \frac{1}{2} t_C + WS - 5$	Data Valid before Rising Edge of $\overline{WR}$	94		ns

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		33	ns
$t_{RWP} = t_C$	RDY Pulse Width	66		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	90		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	76		ns
$t_{HAD} = \frac{3}{4} t_C + 85$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		135	ns
$t_{HAE} = t_C + 85$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		151*	ns
$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after Falling Edge on $\overline{HLDA}$		99†	ns
$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable before Rising Edge of $\overline{HLDA}$	99†		ns

***Note:** t_{HAE} may be as long as ($3t_C + 4ws + 72t_C + 90$) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value is for the optimal case.

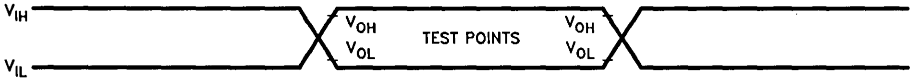
†**Note:** Due to emulation restrictions—actual limits will be better.

UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{\text{URD}}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{\text{URD}}$	10		ns
t_{RPW}	$\overline{\text{URD}}$ Pulse Width	100		ns
t_{OE}	$\overline{\text{URD}}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{\text{URD}}$ to Output Data Invalid (Note 4)	5	35	ns
t_{DRDY}	$\overline{\text{RDRDY}}$ Delay from Rising Edge of $\overline{\text{URD}}$		70	ns
t_{WDW}	$\overline{\text{UWR}}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{\text{UWR}}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{\text{UWR}}$	15		ns
t_A	$\overline{\text{WRRDY}}$ Delay from Rising Edge of $\overline{\text{UWR}}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

Note 4: Guaranteed by design.

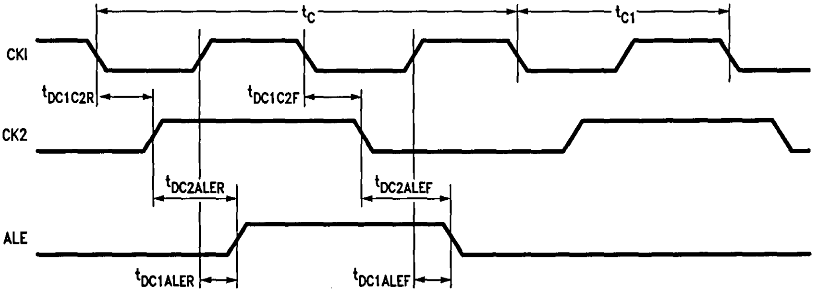


TL/DD/8801-39

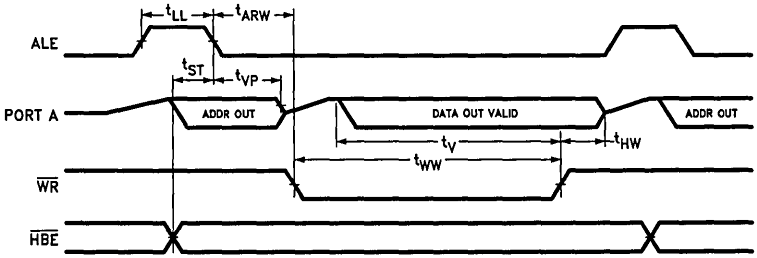
Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{OL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

Timing Waveforms

CKI, CK2, ALE Timing Diagram



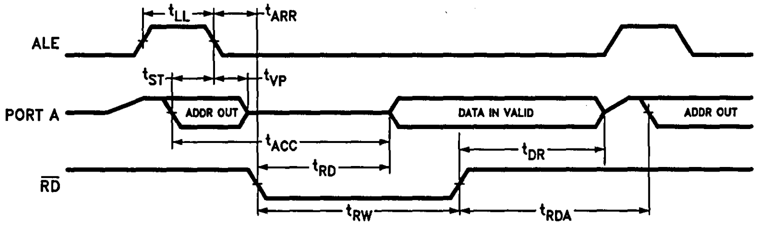
TL/DD/8801-33



TL/DD/8801-3

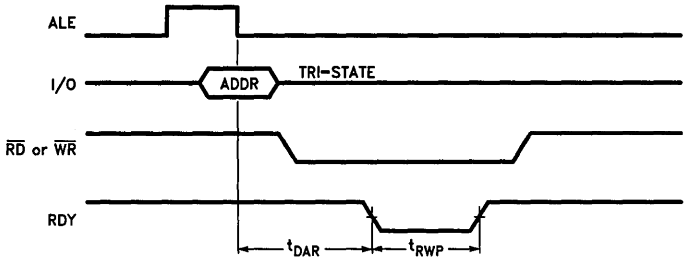
FIGURE 1. Write Cycle

Timing Waveforms (Continued)



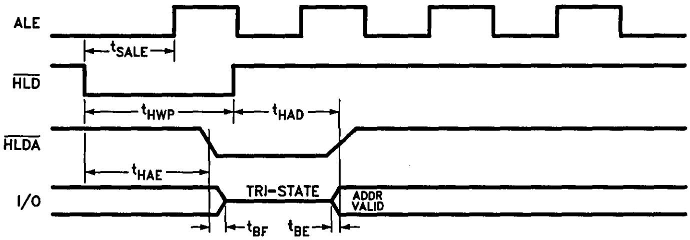
TL/DD/8801-4

FIGURE 2. Read Cycle



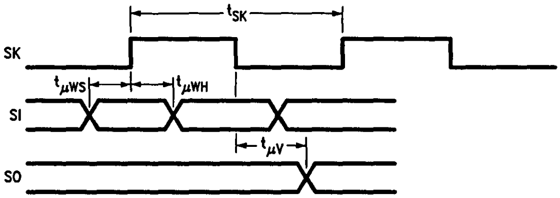
TL/DD/8801-5

FIGURE 3. Ready Mode Timing



TL/DD/8801-6

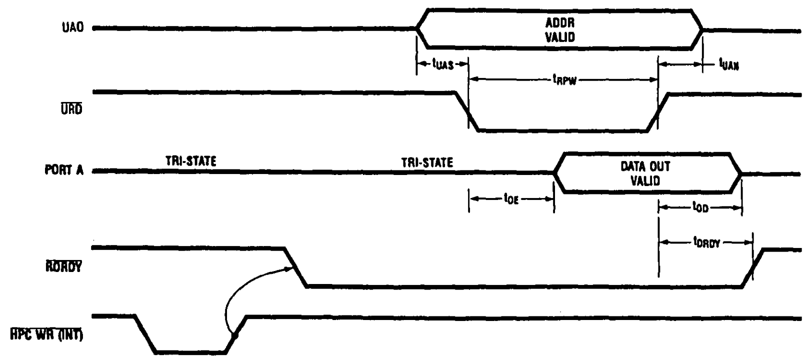
FIGURE 4. Hold Mode Timing



TL/DD/8801-37

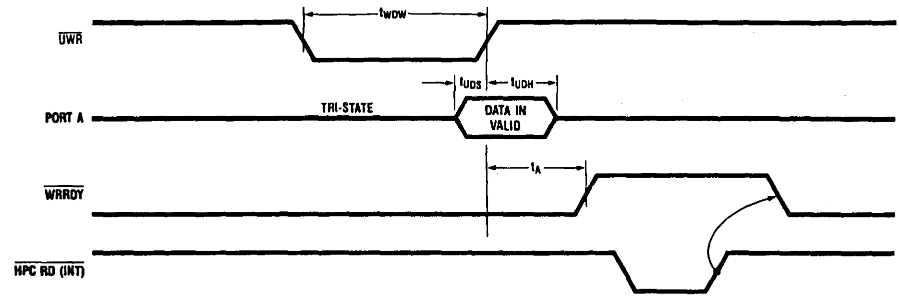
FIGURE 5. MICROWIRE Setup/Hold Timing

Timing Waveforms (Continued)



TL/DD/8801-9

FIGURE 5. UPI Read Timing



TL/DD/8801-10

FIGURE 6. UPI Write Timing

Pin Descriptions

The HPC16083 is available in 68-pin PLCC, LCC, LDCC, PGA and TapePak packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T310	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{HLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	$\overline{\text{WRRDY}}$	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RDRDY}}$	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	$\overline{\text{WR}}$	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output/Input (sampled at reset)
B15:	$\overline{\text{RD}}$	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/$\overline{\text{URD}}$
I3:	INT3	Maskable Interrupt/Input Capture/$\overline{\text{UWR}}$
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4

Pin Descriptions (Continued)

through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V_{CC1} and V_{CC2}	Positive Power Supply
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI	The Chip System Clock Input
CKO	The Chip System Clock Output (inversion of CKI)

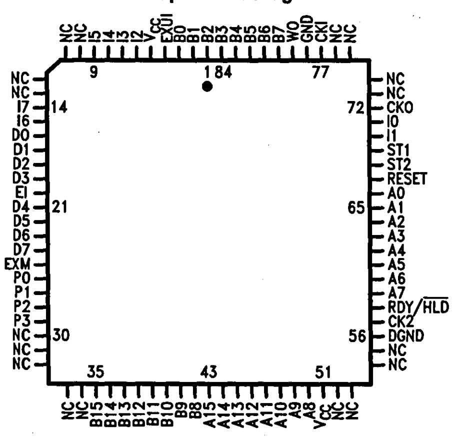
Pins CKI and CKO are usually connected across an external crystal.

CK2	Clock Output (CKI divided by 2)

OTHER PINS

$\overline{\text{WO}}$	This is an active low open drain output that signals an illegal situation has been detected by the Watch Dog logic.
ST1	Bus Cycle Status Output: indicates first opcode fetch.
ST2	Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).
$\overline{\text{RESET}}$	is an active low input that forces the chip to restart and sets the ports in a TRI-STATE® mode.
RDY/$\overline{\text{HLD}}$	has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.
NC	(no connection) do not connect anything to this pin.
EXM	External memory enable (active high) disables internal ROM and maps it to external memory.
EI	External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.
$\overline{\text{EXUI}}$	External interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 (Active Low).

TapePak Package


NC NC I5 I4 I3 I2 VCC2 EXUI B0 B1 B2 B3 B4 B5 B6 B7 WO GND CKI NC NC
9
1 84
77
NC NC I7 I6 D0 D1 D2 D3 EI D4 D5 D6 D7 EXM P0 P1 P2 P3 NC NC NC
14
21
30
72
65
56
NC NC CKO I0 I1 ST1 ST2 RESET A0 A1 A2 A3 A4 A5 A6 A7 RDY/HLD CK2 DGND NC NC
35
43
51
NC B15 B14 B13 B12 B11 B10 B9 B8 A15 A14 A13 A12 A11 A10 A9 A8 VCC1 NC NC NC

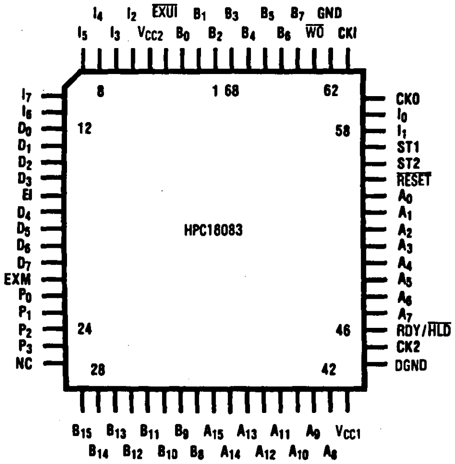
TL/DD/8801-34

Top View

Order Number HPC16083T
Available in TapePak

Connection Diagrams

Plastic, Leadless and Leaded Chip Carriers

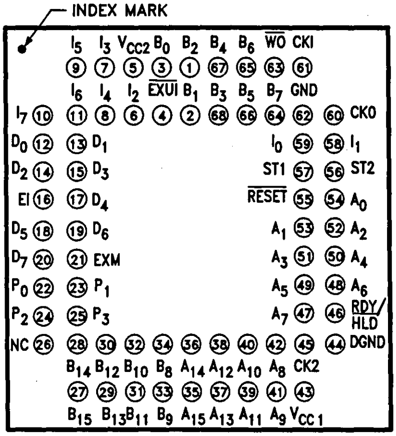

I4 I2 EXUI B1 B3 B5 B7 GND
I5 I3 VCC2 B0 B2 B4 B6 WO CKI
8
1 68
62
12
58
24
46
28
42
I7 I6 D0 D1 D2 D3 EI D4 D5 D6 D7 EXM P0 P1 P2 P3 NC
CKO I0 I1 ST1 ST2 RESET A0 A1 A2 A3 A4 A5 A6 A7 RDY/HLD CK2 DGND
HPC16083
B15 B13 B11 B9 A15 A13 A11 A9 VCC1
B14 B12 B10 B8 A14 A12 A10 A8

TL/DD/8801-11

Top View

Order Number HPC16083E, EL or V
See NS Package Number E68B, EL68A or V68A

Pin Grid Array Pinout


INDEX MARK
I5 I3 VCC2 B0 B2 B4 B6 WO CKI
9 7 5 3 1 67 65 63 61
I6 I4 I2 EXUI B1 B3 B5 B7 GND
I7 10 11 8 6 4 2 68 66 64 62 60 CKO
D0 12 13 D1 I0 59 58 I1
D2 14 15 D3 ST1 57 56 ST2
EI 16 17 D4 RESET 55 54 A0
D5 18 19 D6 A1 53 52 A2
D7 20 21 EXM A3 51 50 A4
P0 22 23 P1 A5 49 48 A6
P2 24 25 P3 A7 47 46 RDY/HLD
NC 26 28 30 32 34 36 38 40 42 45 44 DGND
B14 B12 B10 B8 A14 A12 A10 A8 CK2
27 29 31 33 35 37 39 41 43
B15 B13 B11 B9 A15 A13 A11 A9 VCC1

TL/DD/8801-12

Top View
(looking down on component side of PC Board)

Order Number HPC16083U
See NS Package Number U68A

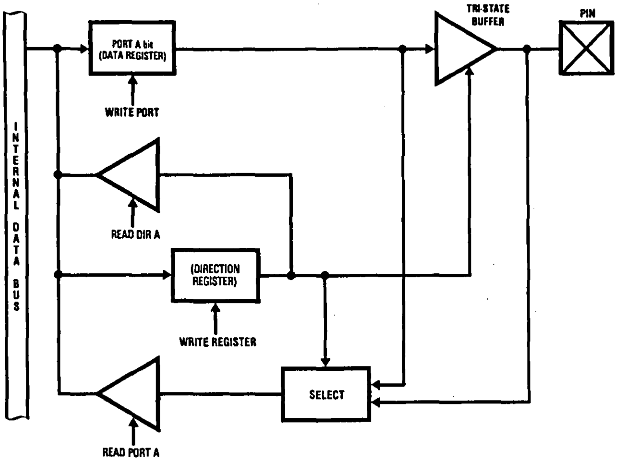
Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 7*), consists of a data register and a direction register. Port B (see *Figures 8, 9, 10*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

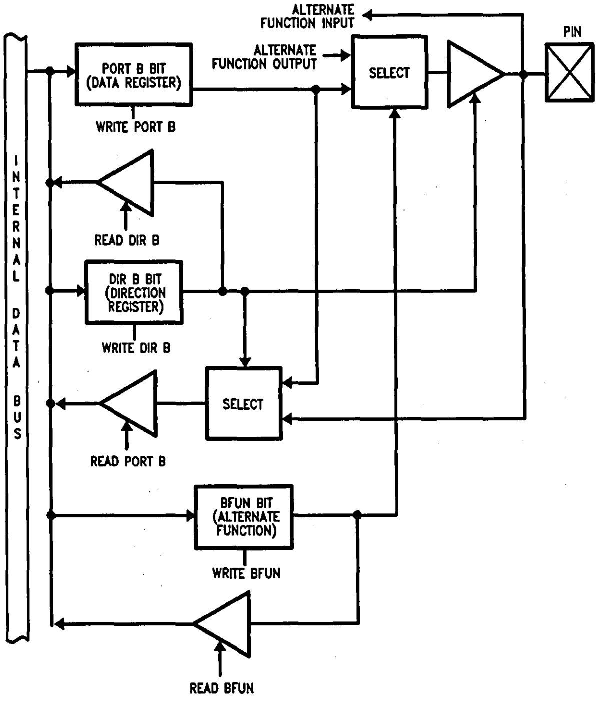
A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.



TL/DD/8801-13

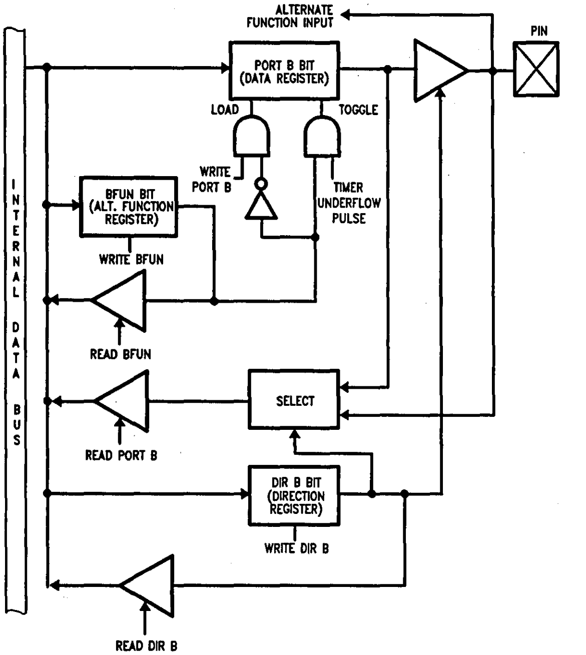
FIGURE 7. Port A: I/O Structure



TL/DD/8801-14

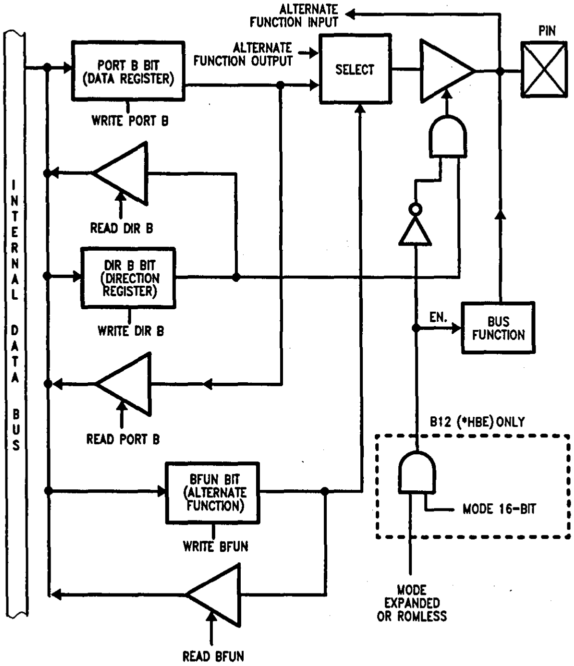
FIGURE 8. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

Ports A & B (Continued)



TL/DD/8801-15

FIGURE 9. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)



TL/DD/8801-16

FIGURE 10. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC16083 has four operating modes. The ROMless HPC16003 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC16083 is E000 to FFFF (8k bytes). The HPC16003 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC16003 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range and on-chip RAM and Register range, and the "illegal address detection" feature of the Watchdog logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the Watchdog logic is disabled. The EA bit should be set to "1" by software when using the HPC16003 to disable the "illegal address detection" feature of Watchdog.

All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, $\overline{\text{RD}}$, $\overline{\text{WR}}$ and $\overline{\text{HBE}}$. The High Byte Enable pin ($\overline{\text{HBE}}$) is used in 16-bit mode to select high order memory bytes. The $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling $\overline{\text{HBE}}$ high at reset. If $\overline{\text{HBE}}$ is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC16083 and HPC16003.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC16083 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC16083 functions as a self-contained microcomputer (see *Figure 11*) with all memory (RAM and ROM) on-chip. It can address internal memory only, consisting of 8k bytes of ROM (E000 to FFFF) and 256 bytes of on-chip RAM and registers (0000 to 01FF). The "illegal address detection" feature of the Watchdog is enabled in the Single-Chip Normal mode and a Watchdog Output ($\overline{\text{WO}}$) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC16083 to address external memory in addition to the on-chip ROM and RAM (see Table II). Watchdog illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC16083 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 8k bytes of external memory may be used with the HPC16083 (see Table II). The Watchdog circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

EXPANDED ROMLESS MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of Watchdog is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

TABLE II. HPC16083 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	E000:FFFF on-chip
Expanded Normal	0	1	E000:FFFF on-chip 0200:DFFF off-chip
Single-Chip ROMless	1	0	E000:FFFF off-chip
Expanded ROMless	1	1	0200:FFFF off-chip

Note: In all operating modes, the on-chip RAM and Registers (0000:01FF) may be accessed.

HPC16003 Operating Modes

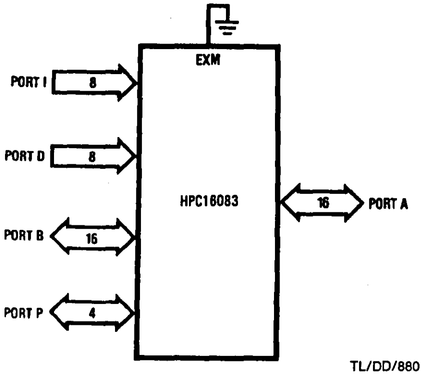
EXPANDED ROMLESS MODE (HPC16003)

Because the HPC16003 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC16003 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between F000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the Watchdog logic.

TABLE III. HPC16003 Operating Modes

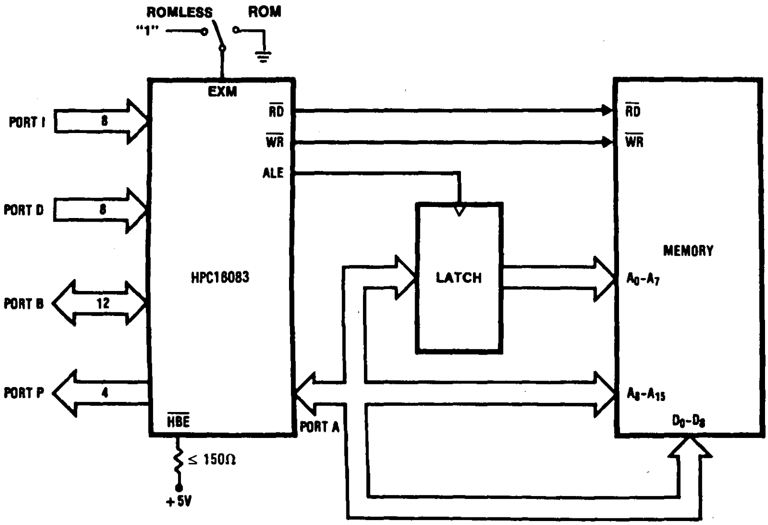
Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0200:FFFF off-chip

Note: The on-chip RAM and Registers (0000:01FF) of the HPC16003 may be accessed at all times.



TL/DD/8801-17

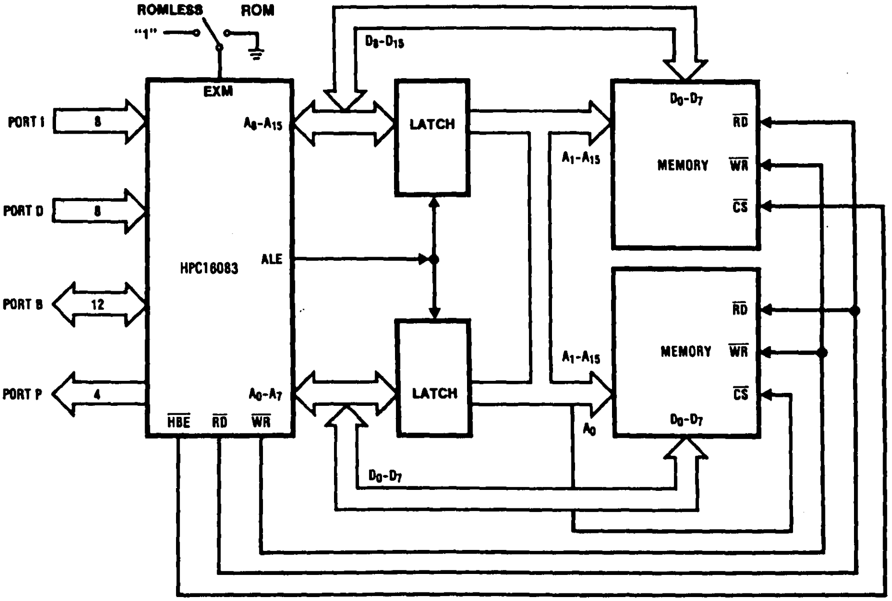
FIGURE 11. Single-Chip Mode



TL/DD/8801-18

FIGURE 12. 8-Bit External Memory

HPC16003 Operating Modes (Continued)



TL/DD/8801-19

FIGURE 13. 16-Bit External Memory

Wait States

The internal ROM can be accessed at the maximum operating frequency with one wait state. With 0 wait states, internal ROM accesses are limited to $\frac{2}{3}$ f_C max.

The HPC16083 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC16083: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16083 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16083 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the $\overline{\text{RESET}}$ or the NMI. The $\overline{\text{RESET}}$ input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16083 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT mode, the processor is returned to full operation by the $\overline{\text{RESET}}$ or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC16083 to resume normal operation.

HPC16083 Interrupts

Complex interrupt handling is easily accomplished by the HPC16083's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table IV.

TABLE IV. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
$FFFF:FFFE	$\overline{\text{RESET}}$	0
$FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
$FFFB:FFFA	External interrupt on I2 pin	2
$FFF9:FFF8	External interrupt on I3 pin	3
$FFF7:FFF6	External interrupt on I4 pin	4
$FFF5:FFF4	Overflow on internal timers	5
$FFF3:FFF2	Internal on the UART transmit/receive complete or external on EXUI	6
$FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC16083 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table IV. The interrupt on $\overline{\text{RESET}}$ has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the $\overline{\text{RESET}}$, which is serviced immediately.

$\overline{\text{RESET}}$ and $\overline{\text{EXUI}}$ are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt ($\overline{\text{EXUI}}$) is shared with the UART interrupt. This interrupt is level-low sensitive. To select this interrupt disable the ERI and ETI UART interrupt bits in the ENUI register. To select the UART interrupt leave this pin floating or tie it high.

Interrupt Control Registers

The HPC16083 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

$\overline{\text{RESET}}$ and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16083 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

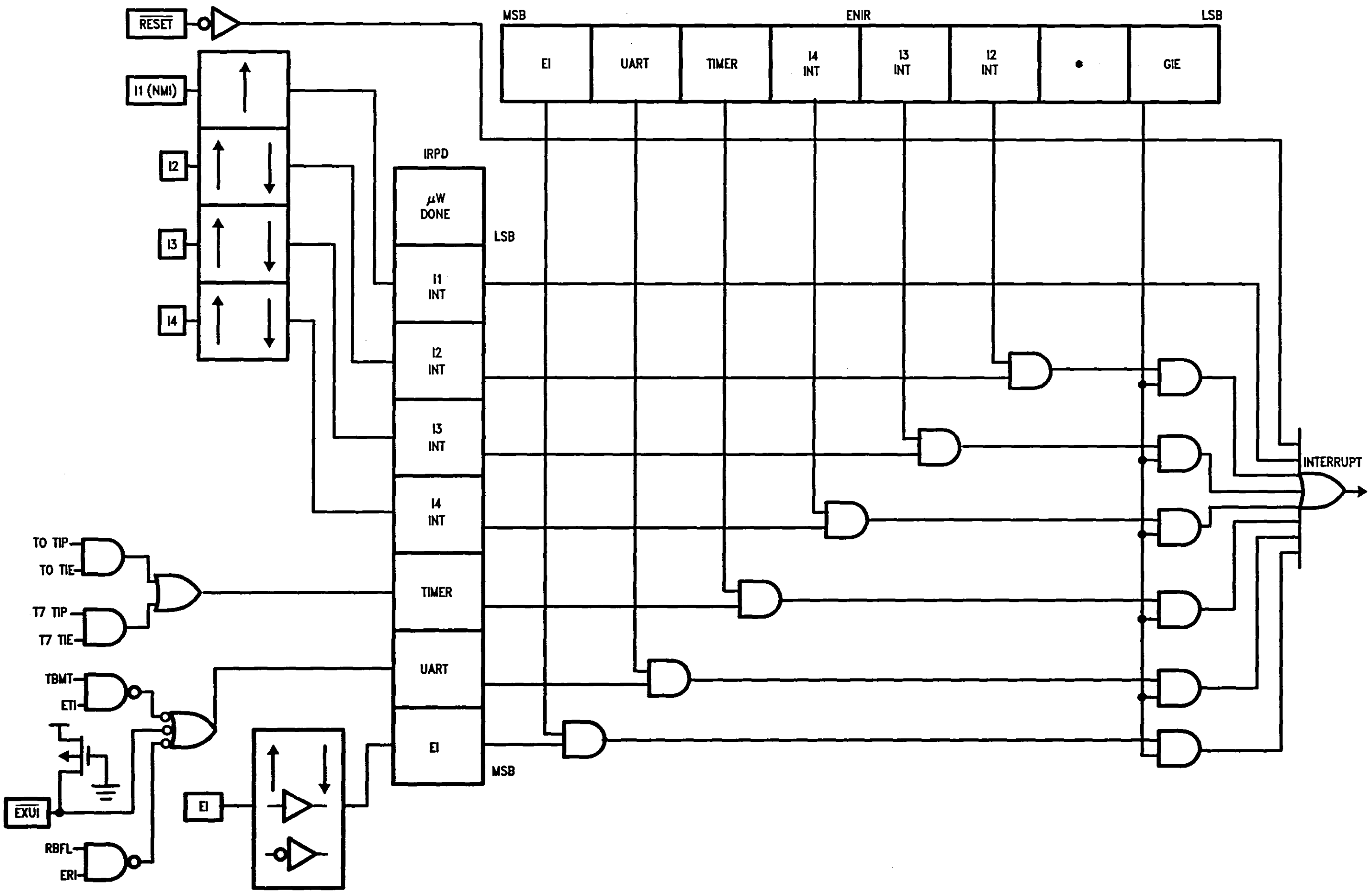
Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 14* shows the Interrupt Enable Logic.

$\overline{\text{RESET}}$

The $\overline{\text{RESET}}$ input initializes the processor and sets ports A and B in the TRI-STATE condition and port P in the LOW state. $\overline{\text{RESET}}$ is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between F000 and FFFF when using the HPC16003.



TL/DD/8801-20

FIGURE 14. Block Diagram of Interrupt Logic

Timer Overview

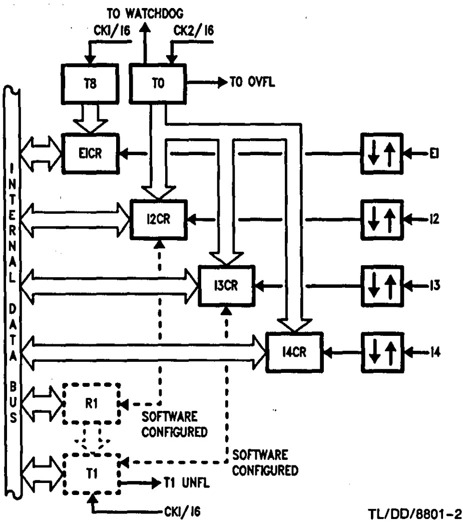
The HPC16083 contains a powerful set of flexible timers enabling the HPC16083 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16083 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watchdog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 15*).

The HPC16083 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 16*).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.



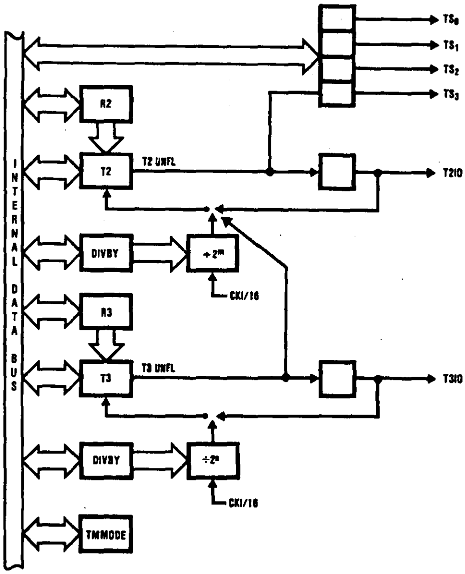
TL/DD/8801-21

FIGURE 15. Timers T0, T1 and T8 with Four Input Capture Registers

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16083 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 16*).

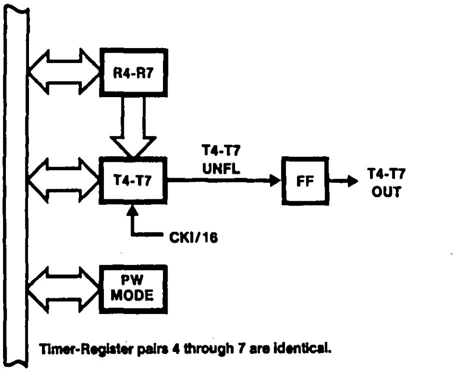
Timer/register pairs 4–7 form four identical units which can generate synchronous outputs on port P (see *Figure 17*).



TL/DD/8801-22

FIGURE 16. Timers T2–T3 Block

Timer Overview (Continued)



Timer-Register pairs 4 through 7 are identical.

TL/DD/8801-23

FIGURE 17. Timers T4-T7 Block

Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to ½ the frequency of the source used for clocking the timer.

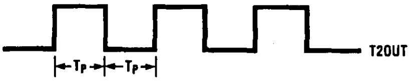
Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16083.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.



TL/DD/8801-24

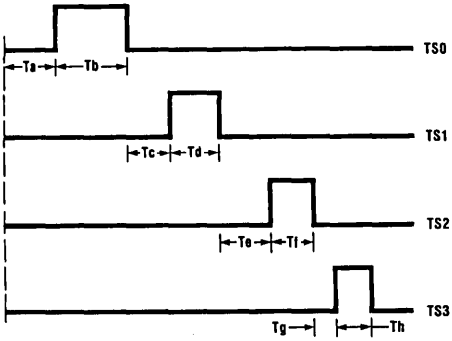
FIGURE 18. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0-TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 19* is an example of synchronous pulse train generation.

Watchdog Logic

The Watchdog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watchdog logic are potentially infinite loops and illegal addresses. Should the Watchdog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the Watchdog Output ($\overline{\text{WO}}$) pin low. The $\overline{\text{WO}}$ pin is an open drain output and can be connected to the $\overline{\text{RESET}}$ or NMI inputs or to the users external logic.

*Note: See Operating Modes for details.

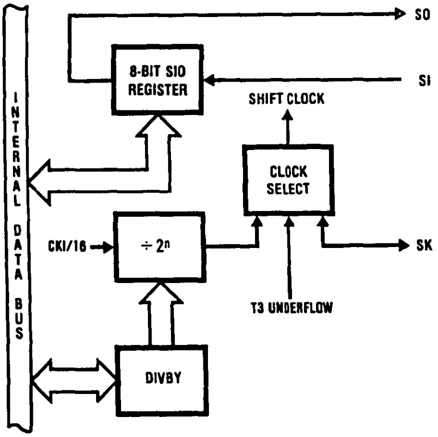


TL/DD/8801-25

FIGURE 19. Synchronous Pulse Generation

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 20*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.



TL/DD/8801-26

FIGURE 20. MICROWIRE/PLUS

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS Operation

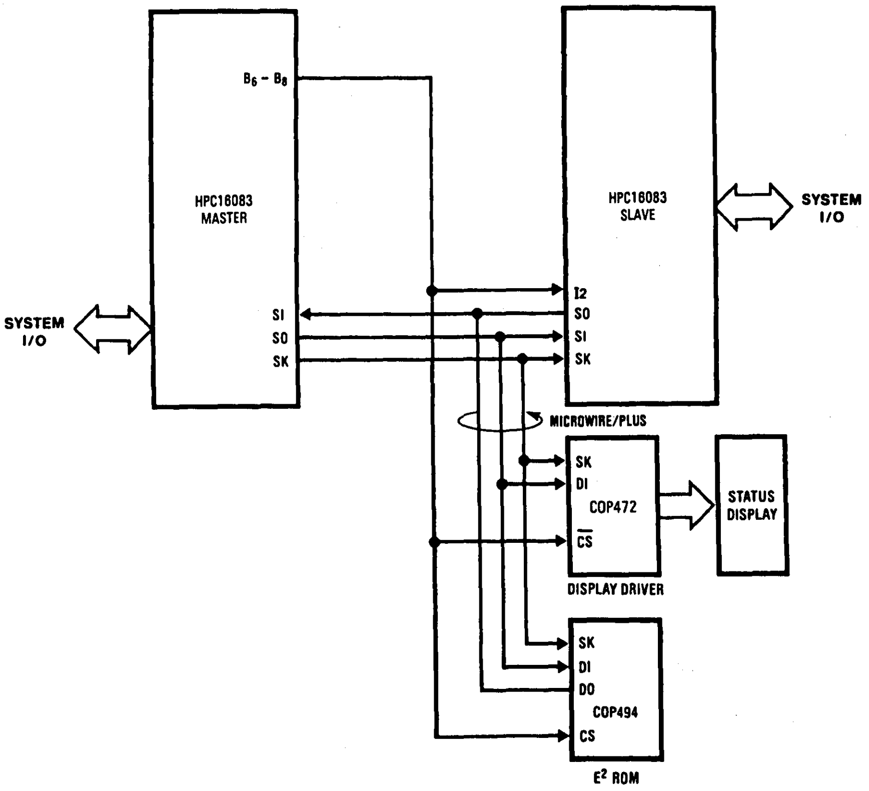
The HPC16083 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16083 is the master or slave. The shift clock is generated when the HPC16083 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16083 is configured as a slave. When the HPC16083 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 21 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC16083 microcontrollers interconnected to other MICROWIRE peripherals. HPC16083 #1 is set up as the master and initiates all data transfers. HPC16083 #2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of a LCD display controlled by the COP472 display driver. The data to be displayed is sent serially to the COP472 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC16083 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.



TL/DD/8801-27

FIGURE 21. MICROWIRE/PLUS Application

HPC16083 UART

The HPC16083 contains a software programmable UART. The UART (see *Figure 22*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

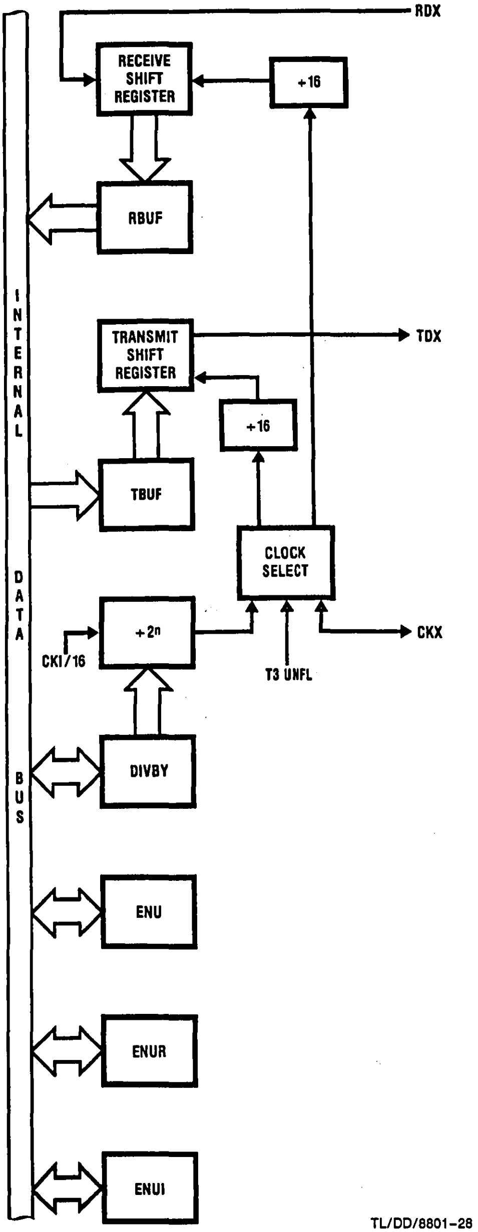
The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC16083 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16083 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16083 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16083 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.



TL/DD/8801-28

FIGURE 22. UART Block Diagram

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC16083 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC16083's and set up systems with very high data exchange rates. Another area of application could be where a HPC16083 is programmed as an intelligent peripheral to a host system such as the Series 32000® microprocessor. *FIGURE 23* illustrates how a HPC16083 could be used an an intelligent peripherial for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe ($\overline{\text{URD}}$), a Write Strobe ($\overline{\text{UWR}}$), a Read Ready Line ($\overline{\text{RDRDY}}$), a Write Ready Line ($\overline{\text{WRRDY}}$) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The $\overline{\text{URD}}$ and $\overline{\text{UWR}}$ inputs may be used to interrupt the HPC16083. The $\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$ outputs may be used to interrupt the host processor.

The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC16083 is the data bus. UPI can only be used if the HPC16083 is in the Single-Chip mode.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16083 supports shared memory access with two pins. The pins are the RDY/$\overline{\text{HLD}}$ input pin and the $\overline{\text{HLDA}}$ output pin. The user can software select either the Hold or Ready function by the state of a control bit. The $\overline{\text{HLDA}}$ output is multiplexed onto port B.

The host uses DMA to interface with the HPC16083. The host initiates a data transfer by activating the $\overline{\text{HLD}}$ input of the HPC16083. In response, the HPC16083 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{\text{HLDA}}$) from the HPC16083 indicating that the sytem bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16083 resumes normal operations.

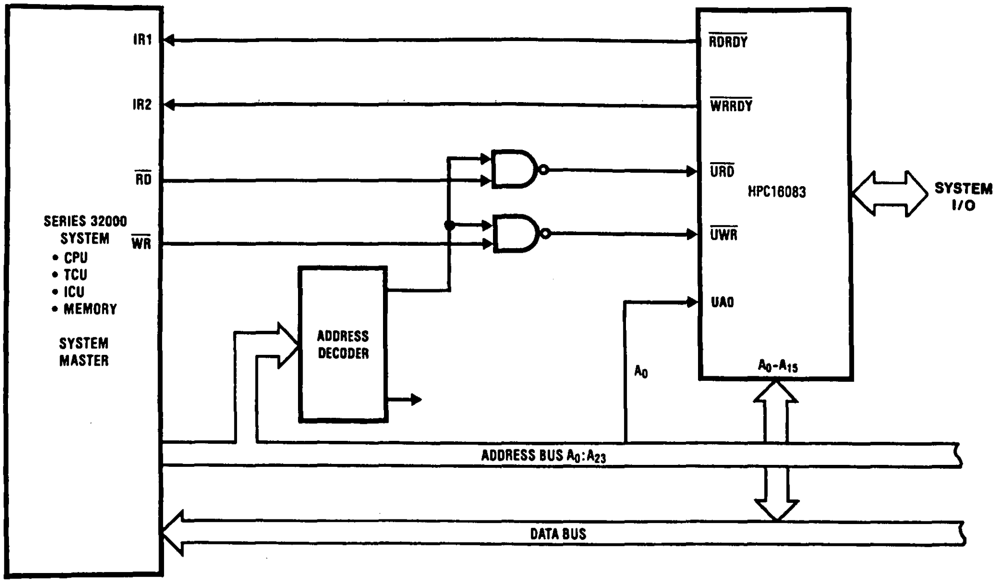
FIGURE 24 illustrates an application of the shared memory interface between the HPC16083 and a Series 32000 system. To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus.

Memory

The HPC16083 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 8 kbytes of ROM and 256 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16083 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

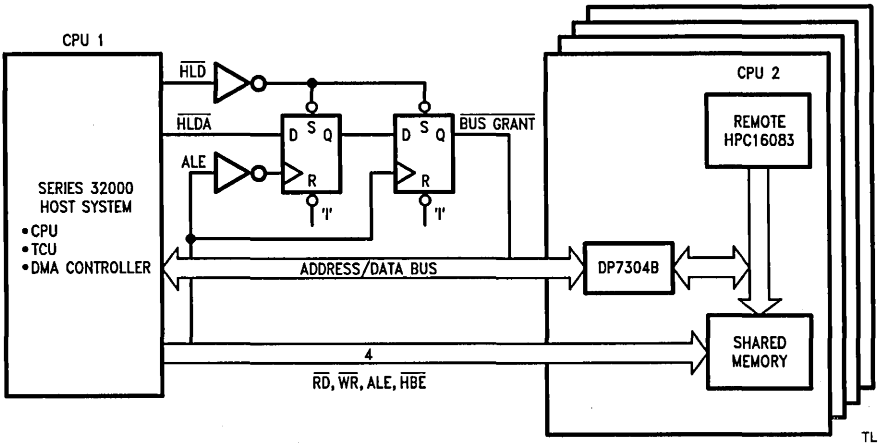
The HPC16083 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table V.



TL/DD/8801-29

FIGURE 23. HPC16083 as a Peripheral: (UPI Interface to Series 32000 Application)

Shared Memory Support (Continued)



TL/DD/8801-30

FIGURE 24. Shared Memory Application: HPC16083 Interface to Series 32000 System

TABLE V. HPC16083 Memory Map

FFFF:FFF0	Interrupt Vectors	USER MEMORY
FFEF:FFD0	JSRP Vectors	
FFCF:FFCE : : E001:E000	On-Chip ROM	
DFFF:DFFE : : 0201:0200	External Expansion Memory	
01FF:01FE : : 01C1:01C0	On-Chip RAM	USER RAM
0195:0194	Watchdog Address	Watchdog Logic
0192	T0CON Register	Timer Block T0:T3
0191:0190	TMMODE Register	
018F:018E	DIVBY Register	
018D:018C	T3 Timer	
018B:018A	R3 Register	
0189:0188	T2 Timer	
0187:0186	R2 Register	
0185:0184	I2CR Register/ R1	
0183:0182	I3CR Register/ T1	
0181:0180	I4CR Register	
015E:015F	EICR	Timer Block T4:T7
015C	EICON	
0153:0152	Port P Register	
0151:0150	PWMODE Register	
014F:014E	R7 Register	
014D:014C	T7 Timer	
014B:014A	R6 Register	
0149:0148	T6 Timer	
0147:0146	R5 Register	
0145:0144	T5 Timer	
0143:0142	R4 Register	
0141:0140	T4 Timer	
0128	ENUR Register	UART
0126	TBUF Register	
0124	RBUF Register	
0122	ENUI Register	
0120	ENU Register	
0104	Port D Input Register	
00F5:00F4	BFUN Register	PORTS A & B CONTROL
00F3:00F2	DIR B Register	
00F1:00F0	DIR A Register / IBUF	
00E6	UPIC Register	UPI CONTROL
00E3:00E2	Port B	PORTS A & B
00E1:00E0	Port A / OBUF	
00DE	Microcode ROM Dump	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00DD:00DC	HALT Enable Register	
00D8	Port I Input Register	
00D6	SIO Register	
00D4	IRCD Register	
00D2	IRPD Register	
00D0	ENIR Register	
00CF:00CE	X Register	HPC CORE REGISTERS
00CD:00CC	B Register	
00CB:00CA	K Register	
00C9:00C8	A Register	
00C7:00C6	PC Register	
00C5:00C4	SP Register	
00C3:00C2	(reserved)	
00C0	PSW Register	
00BF:00BE : : 0001:0000	On-Chip RAM	USER RAM

Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused outputs should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least 1 μF and bypass their outputs with a 10 μF to 50 μF tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a 10 μF to 20 μF tantalum electrolytic capacitor or a 50 μF to 100 μF aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.

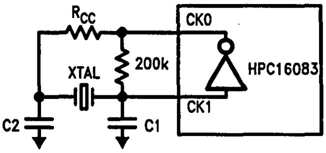
A recommended crystal oscillator circuit to be used with the HPC is shown below. See table for recommended component values. The recommended values given in the table below have yielded consistent results and are made to match a crystal with a 20 pF load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within 1″ distance. This is to reduce lead inductance from long PC traces, as well-as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A 1.0 μF, a 0.1 μF, and a 0.001 μF dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

HPC Oscillator Table

f_C (MHz)	R_{CC} (Ω)	C1 (pF)	C2 (pF)
2	50	82	100
4	50	62	75
6	50	50	56
8	50	47	50
10	50	39	50
12	0	39	39
14	0	33	39
16	0	33	39
18	0	33	33
20	0	33	33
22	0	27	39
24	0	27	39
26	0	27	33
28	0	27	33
30	0	27	27



TL/DD/8801-40

Crystal Specifications:

"AT" cut, parallel resonant crystals tuned to the desired frequency with the following specifications are recommended:

Series resistance < 65Ω

Loading capacitance C_L = 20 pF

HPC16083 CPU

The HPC16083 CPU has a 16-bit ALU and six 16-bit registers

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16083 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action	
ARITHMETIC INSTRUCTIONS			
ADD	Add	MA+Meml → MA	carry → C
ADC	Add with carry	MA+Meml+C → MA	carry → C
ADDS	Add short imm8	MA+imm8 → MA	carry → C
DADC	Decimal add with carry	MA+Meml+C → MA (Decimal)	carry → C
SUBC	Subtract with carry	MA−Meml+C → MA	carry → C
DSUBC	Decimal subtract w/carry	MA−Meml+C → MA (Decimal)	carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C	
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C	
DIVD	Divide Double Word (unsigned)	(X & MA)/Meml → MA, rem → X, 0 → K, carry → C	
IFEQ	If equal	Compare MA & Meml, Do next if equal	
IFGT	If greater than	Compare MA & Meml, Do next if MA > Meml	
AND	Logical and	MA and Meml → MA	
OR	Logical or	MA or Meml → MA	
XOR	Logical exclusive-or	MA xor Meml → MA	
MEMORY MODIFY INSTRUCTIONS			
INC	Increment	Mem + 1 → Mem	
DECSZ	Decrement, skip if 0	Mem −1 → Mem, Skip next if Mem = 0	

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	MemI → MA
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ↔ Mem
	Exchange, incr/decr X	A ↔ Mem(X), X ±1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP
POP	Pop Stack to Memory	SP−2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ↔ A,B±1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B,imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A − 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ↔ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → [SP],SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → [SP],SP+2 → SP,PC+ # → PC (#is +1025 to −1023)
JSRL	Jump subroutine long	PC → [SP],SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +257 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP−2 → SP,[SP] → PC
RETSK	Return then skip next	SP−2 → SP,[SP] → PC, & skip
RETI	Return from interrupt	SP−2 → SP,[SP] → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8 or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
MemI is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data
imm8 is 8-bit immediate data only

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir.		Direct	Indir.	Index	Immed.	Direct		Immed.	
	(B)	(X)					*	**	*	**
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16083 has been designed to be extremely code-efficient. The HPC16083 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16083, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16083 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16083 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16083 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16083 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

DEVELOPMENT SYSTEM

The Microcomputer On Line Emulator (MOLE) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC family of products. The development system consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the development system is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other development systems in a multi-development system environment.

The development system can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information system package contains:
DIAL-A-HELPER Users Manual
Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

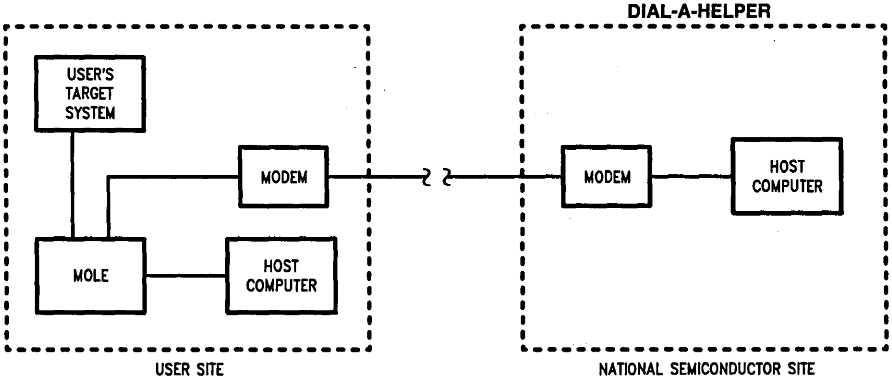
Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBMR	Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	4244105883001
	HPC-VMS	Assembler, Loader, Librarian for VAX/VMS	HPC Software User's Manual and 9 Track Tape	424410836-001
	HPC-VMS-C	C Compiler for VAX/VMS	HPC Software User's Manual and 9 Track Tape (Includes Assembler)	424410883-001
	424410897-001	Users Manual		424410897-001

VAX UNIX will be supported in the near future. Contact field sales for more information.

Development Support (Continued)

Voice: (408) 721-5582
Modem: (408) 739-1162
Baud: 300 or 1200 Baud
Set-Up: Length: 8-bit
Parity: None
Stop Bit: 1
Operation: 24 hrs, 7 days

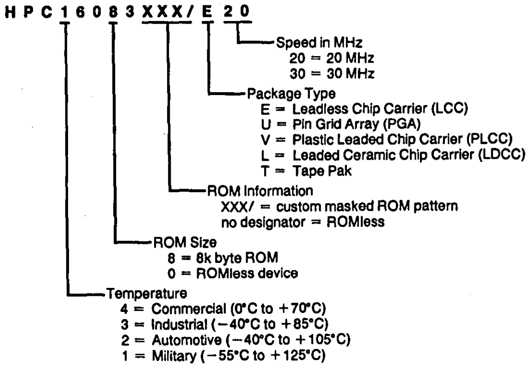


TL/DD/8801-32

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16083 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.


HPC16083XXX/E20
Speed in MHz
20 = 20 MHz
30 = 30 MHz
Package Type
E = Leadless Chip Carrier (LCC)
U = Pin Grid Array (PGA)
V = Plastic Leaded Chip Carrier (PLCC)
L = Leaded Ceramic Chip Carrier (LDCC)
T = Tape Pak
ROM Information
XXX/ = custom masked ROM pattern
no designator = ROMless
ROM Size
8 = 8k byte ROM
0 = ROMless device
Temperature
4 = Commercial (0°C to +70°C)
3 = Industrial (−40°C to +85°C)
2 = Automotive (−40°C to +105°C)
1 = Military (−55°C to +125°C)

TL/DD/8801-31

FIGURE 8. HPC Family Part Numbering Scheme

Examples

HPC46003E20 — ROMless, Commercial temp. (0°C to 70°C), DCC
HPC16083XXX/U20— 8k masked ROM, Military temp. (−55°C to +125°C), PGA
HPC26083XXX/V20— 8k masked ROM, Automotive temp. (−40°C to +105°C), PLCC

National Semiconductor

PRELIMINARY

HPC46083MH High-Performance microController Emulator

General Description

The HPC46083MH is the emulator device for the HPC16083 and is a member of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC46083MH has 8k bytes of on-chip EPROM. The HPC46083MH is a two chip system packaged in a dual cavity ceramic LDCC type package, with a lid on top, and a UV quartz window on bottom. Within the package is an HPC46083 and UV-erasable EPROM with port recreation logic. The EPROM die allows the HPC to function normally, while executing code out of the EPROM. The HPC46083MH may be programmed using a programming card to adapt the part to a normal 27C64 EPROM programmer. The part will function as the normal HPC, and the use of the EPROM should be transparent to the user. The only system design consideration is that pin 5 is V_{PP}, not V_{CC}, and should be tied to V_{CC} during normal operation. Pin 26 should also be tied to V_{CC}.

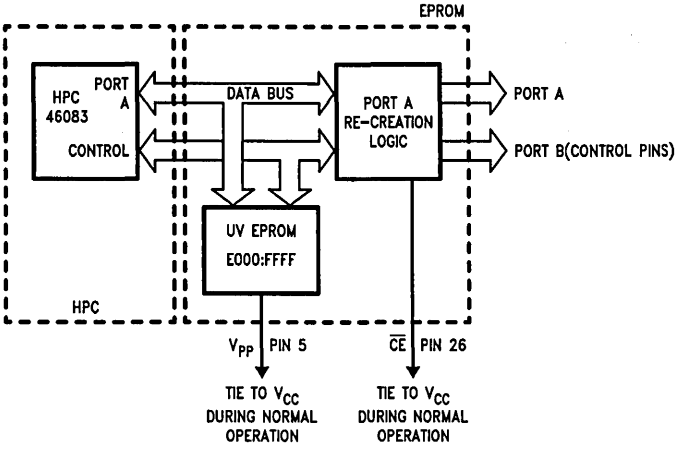
The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG™ logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips.

The HPC46083MH is available in 68-pin LDCC type packages.

Features

- HPC family—core features:
  - 16-bit architecture, both byte and word
  - 16-bit data bus, ALU, and registers
  - 64k bytes of external memory addressing
  - FAST—200 ns for fastest instruction when using 20.0 MHz clock
  - High code efficiency—most instructions are single byte
  - 16 x 16 multiply and 32 x 16 divide
  - Eight vectored interrupt sources
  - Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
  - MICROWIRE/PLUS serial I/O interface
  - CMOS—very low power with two power save modes: IDLE and HALT
- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 8k bytes of EPROM, 256 bytes of RAM on chip
- Commercial (0°C to +70°C)

Block Diagram (HPC46083 with 8k EPROM shown)



TL/DD/10105-2

20 MHz
Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current 54 + 4 × f_{in}	V_{CC} = 5.5V, f_{in} = 20.0 MHz (Note 1)		134	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		62	mA
I_{CC_2}	IDLE Mode Current 0.5 + 1.25 × f_{in}	V_{CC} = 5.5V, f_{in} = 20.0 MHz, (Note 1)		26	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		3	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		2	mA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		250	µA
INPUT VOLTAGE LEVELS $\overline{\text{RESET}}$, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	µA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 µA (Note 2)	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 µA (Note 2)		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	I_{OH} = −6 mA	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE Leakage Current			±5	µA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{\text{RESET}}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}, CKI driven to V_{IH1} and V_{IL1}, with rise and fall times less than 10 ns.

Note 2: This is guaranteed by design and not tested.

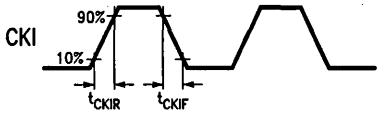
Note 3: Test duration is 100 ms.

20 MHz (1 Wait State Operation)
AC Electrical Characteristics V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency (Note 4)	2	20.0	MHz
$t_{C1} = 1/f_C$	Clock Period	50	500	ns
t_{CKIR} (Note 3)	CKI Rise Time		7	ns
t_{CKIF} (Note 3)	CKI Fall Tiime		7	ns
$[t_{CKIH}/(t_{CKIH} + t_{CKIL})]100$	Duty Cycle	45	55	%
$t_C = 2/f_C$	Timing Cycle	100	1000	ns
$t_{LL} = ½\,t_C - 12$	ALE Pulse Width	38		ns
t_{DC1C2R} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	10	50	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	10	50	ns
$t_{DC2ALER} = ¼\,tC + 35$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		65	ns
$t_{DC2ALEF} = ¼\,tC + 35$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		65	ns
$t_{ST} = ¼\,t_C - 20$	Address Valid to ALE Falling Edge	5		ns
$t_{VP} = ¼\,t_C - 5$	Address Hold from ALE Falling Edge	20		ns
$t_{WAIT} = t_C = WS$	Wait State Period	100		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.05	MHz
t_{XIN}	Pulse Width for Timer Inputs	100		ns
f_{MW}	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz

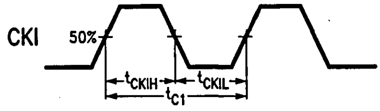
CKI Input Signal Characteristics

Rise/Fall Time



TL/DD/10105-3

Duty Cycle



TL/DD/10105-4

Read Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{\text{RD}}$ Falling Edge	20		ns
$t_{RW} = \frac{1}{2} t_C + WS - 10$	$\overline{\text{RD}}$ Pulse Width	140		ns
$t_{DR} = \frac{3}{4} t_C - 20$	Data Hold after Rising Edge of $\overline{\text{RD}}$	0	55	ns
$t_{ACC} = t_C + WS - 85$ (Note 2)	Address Valid to Input Data Valid		115	ns
$t_{RD} = \frac{1}{2} t_C + WS - 85$	$\overline{\text{RD}}$ Falling Edge to Input Data Valid		65	ns
$t_{RDA} = t_C - 5$	$\overline{\text{RD}}$ Rising Edge to Address Valid	95		ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 pF, other Outputs C_L = 80 pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurements will be skewed.

Note: WS = t_{WAIT} * number of pre-programmed wait states. Minimum and Maximum values are calculated from maximum operating frequency with one (1) wait state pre-programmed.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not directly tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a high confidence level.

Note 3: This is guaranteed by design and not tested.

Note 4: Maximum frequency with 0 wait states is 6 MHz.

Write Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{\text{WR}}$ Falling Edge	45		ns
$t_{WW} = \frac{3}{4} t_C + WS - 20$	$\overline{\text{WR}}$ Pulse Width	155		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{\text{WR}}$	20		ns
$t_V = \frac{1}{2} t_C + WS - 20$	Data Valid before Rising Edge of $\overline{\text{WR}}$	130		ns

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 80$	Falling Edge of ALE to Falling Edge of RDY		45	ns
$t_{RWP} = t_C + 10$	RDY Pulse Width	110		ns
$t_{SALE} = 1\frac{1}{4} t_C + 70$	Falling Edge of $\overline{\text{HLD}}$ to Rising Edge of ALE	145		ns
$t_{HWP} = t_C + 10$	$\overline{\text{HLD}}$ Pulse Width	110		ns
$t_{HAD} = 1\frac{1}{4} t_C + 70$	Rising Edge on $\overline{\text{HLD}}$ to Rising Edge on $\overline{\text{HLDA}}$		345	ns
$t_{HAE} = 2t_C + 130$	Falling Edge on $\overline{\text{HLD}}$ to Falling Edge on $\overline{\text{HLDA}}$		330*	ns
$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after Falling Edge on $\overline{\text{HLDA}}$		116	ns
$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable before Rising Edge of $\overline{\text{HLDA}}$	116		ns

***Note:** t_{HAE} may be as long as ($6t_C$ + 8ws + $144t_C$ + 180) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value tested is for the optimal case.

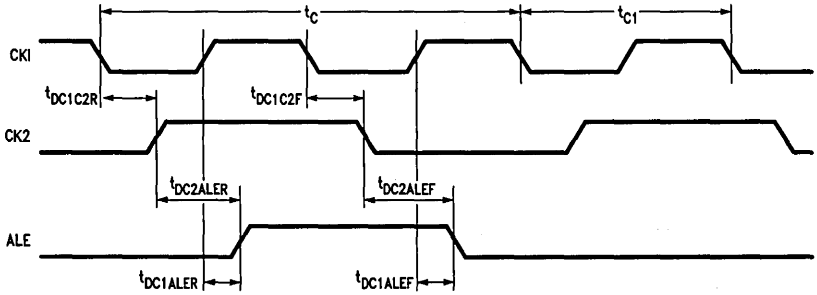
UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{\text{URD}}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{\text{URD}}$	10		ns
t_{RPW}	$\overline{\text{URD}}$ Pulse Width	100		ns
t_{OE}	$\overline{\text{URD}}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{\text{URD}}$ to Output Data Invalid	5	70	ns
t_{DRDY}	$\overline{\text{RDRDY}}$ Delay from Rising Edge of $\overline{\text{URD}}$		70	ns
t_{WDW}	$\overline{\text{UWR}}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{\text{UWR}}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{\text{UWR}}$	20		ns
t_A	$\overline{\text{WRRDY}}$ Delay from Rising Edge of $\overline{\text{UWR}}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

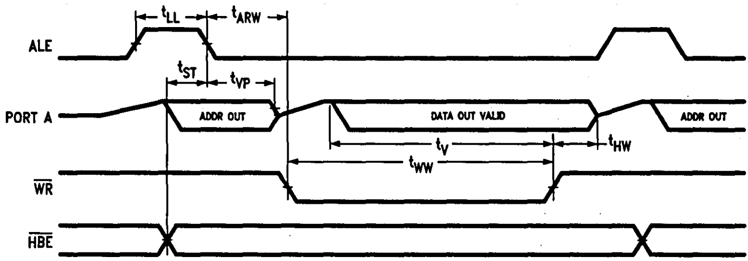
Timing Waveforms

CKI, CK2, ALE Timing Diagram



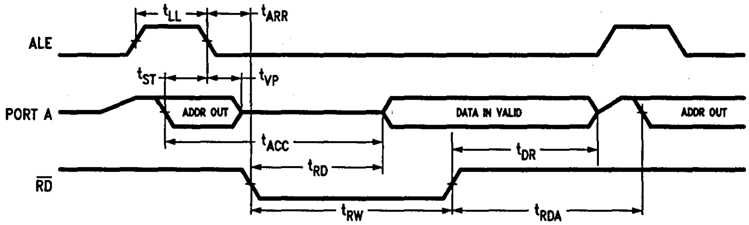
TL/DD/10105-5

Timing Waveforms (Continued)



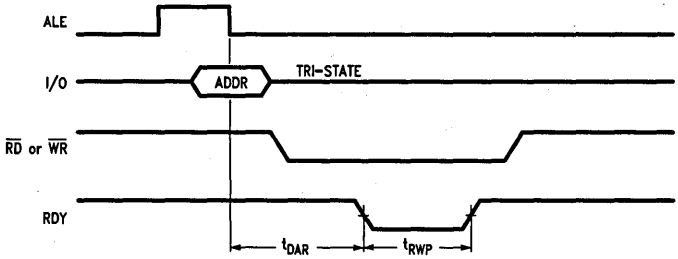
TL/DD/10105-6

FIGURE 1. Write Cycle



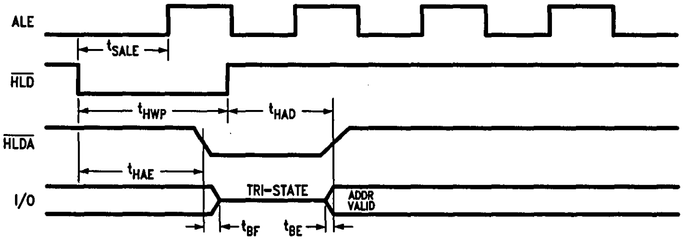
TL/DD/10105-7

FIGURE 2. Read Cycle



TL/DD/10105-8

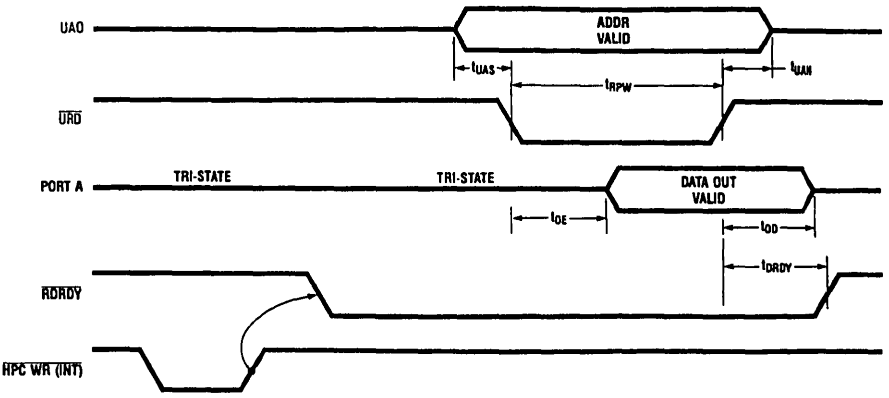
FIGURE 3. Ready Mode Timing



TL/DD/10105-9

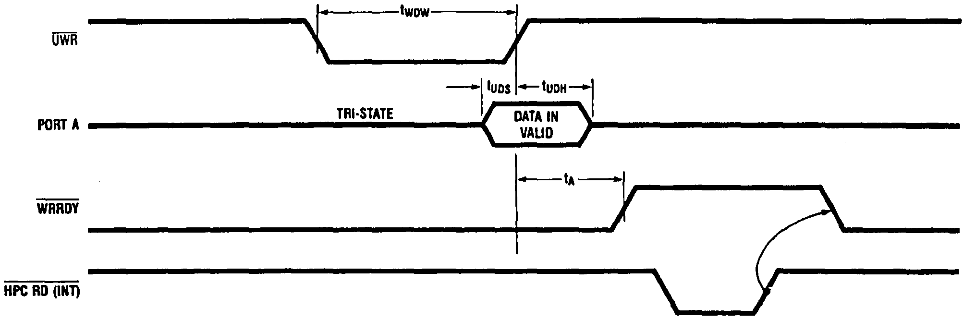
FIGURE 4. Hold Mode Timing

Timing Waveforms (Continued)



TL/DD/10105-10

FIGURE 5. UPI Read Timing



TL/DD/10105-11

FIGURE 6. UPI Write Timing

Pin Descriptions

The HPC46083MH is available in 68-pin LDCC packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T310	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{HLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	$\overline{\text{WRRDY}}$	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RDRDY}}$	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	$\overline{\text{WR}}$	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output/Input (sampled at reset)
B15:	$\overline{\text{RD}}$	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/$\overline{\text{URD}}$
I3:	INT3	Maskable Interrupt/Input Capture/$\overline{\text{UWR}}$
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4

Pin Descriptions (Continued)

through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V_{PP}	Programming Power
V_{CC1}	Positive Power Supply
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: GND and DGND are electrically connected in the package. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI The Chip System Clock Input

CKO The Chip System Clock Output (inversion of CKI)

Pins CKI and CKO are usually connected across an external crystal.

CK2 Clock Output (CKI divided by 2)

OTHER PINS

$\overline{\text{WO}}$ This is an active low open drain output that signals an illegal situation has been detected by the Watch Dog logic.

ST1 Bus Cycle Status Output: indicates first opcode fetch.

ST2 Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).

$\overline{\text{RESET}}$ is an active low input that forces the chip to restart and sets the ports in a TRI-STATE® mode.

RDY/$\overline{\text{HLD}}$ has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.

EXM External memory enable (active high) disables internal EPROM and maps it to external memory.

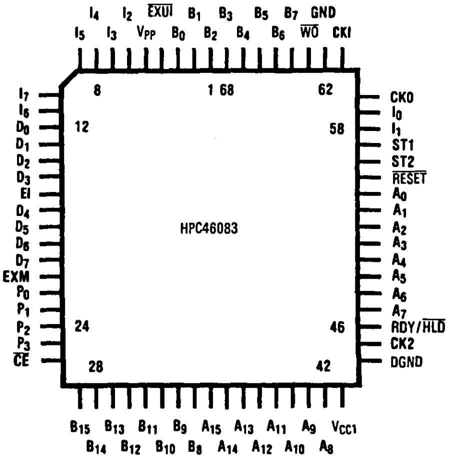
EI External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.

$\overline{\text{EXUI}}$ External interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 (Active Low).

$\overline{\text{CE}}$ Places part in EPROM Programming mode (Active Low).

Connection Diagrams

Leaded Chip Carriers



I_4 I_2 $\overline{\text{EXUI}}$ B_1 B_3 B_5 B_7 GND
I_5 I_3 V_{PP} B_0 B_2 B_4 B_6 $\overline{\text{WO}}$ CKI
8
1 68
62
12
58
24
46
28
42
HPC46083
I_7 I_6 D_0 D_1 D_2 D_3 EI D_4 D_5 D_6 D_7 EXM P_0 P_1 P_2 P_3 $\overline{\text{CE}}$
CKO I_0 I_1 ST1 ST2 $\overline{\text{RESET}}$ A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 RDY/$\overline{\text{HLD}}$ CK2 DGND
B_{15} B_{13} B_{11} B_9 A_{15} A_{13} A_{11} A_9 V_{CC1}
B_{14} B_{12} B_{10} B_8 A_{14} A_{12} A_{10} A_8

TL/DD/10105–15

Top View

The HPC46083MH is a two chip system packaged in a dual cavity ceramic LDCC package, with a UV quartz window on bottom. Within the package is an HPC46083 and a UV-erasable EPROM with port recreation logic. Code executes out of the EPROM. The HPC46083MH may be programmed using a programming card to adapt the part to a 27C64 EPROM programmer. The part functions as the normal HPC, and the use of the EPROM should be transparent to the user. The only system design consideration is that pin 5 is V_{PP}, not V_{CC}, and should be tied to V_{CC} during normal operation. Pin 26 should also be tied to V_{CC}. DGND is connected internally to V_{SS} via a ground plane internal to the package. This should not cause any functional problems to a normal user of the part. Please be careful when inserting the part that the polarity dot is on pin 1.

When programming the part, care should be taken to use only the NSC HPC-EMU-PRGM 980420174 Programming card. This is easily distinguished by the large Yamaichi socket on the top. When programming it in the NSC MOLE Brain Board programmer be sure to use the Chip 2764 option in the programming menu. Use 13.5V ±0.5V for V_{PP}. The part will program in most 27C64 EPROM programmers, however refer to the data sheet to ensure that all timing requirements are met in the programming algorithm. Normal erase times under a UV light run about 45 mins., but may vary depending on the intensity of the light.

Suggested sockets and extractor tool:

Socket #	AMP	PLCC	#821574-1 #6141749
	YAMAICHI	IC51-0684-390 IC120-0684-204	
	ENPLAS	PLCC-68-1.27-02	
Extractor Tool #	AMP	821566-1	

Programming Information

DC Electrical Characteristics $T_A = 25 \pm 5°C$, $V_{CC} = 5.50V \pm 5\%$, $V_{PP} = 13.5 \pm 0.5V$

Symbol	Parameter	Min	Max	Units
I_{PP}	V_{PP} Supply Current during Programming Pulse $\overline{RESET} = \overline{CE} = V_{IL}$		60	mA
I_{CC}	V_{PP} Supply Current		35	mA

Note 1: V_{CC} must be applied either coincidentally or before V_{PP} and removed either coincidentally or after V_{PP}.

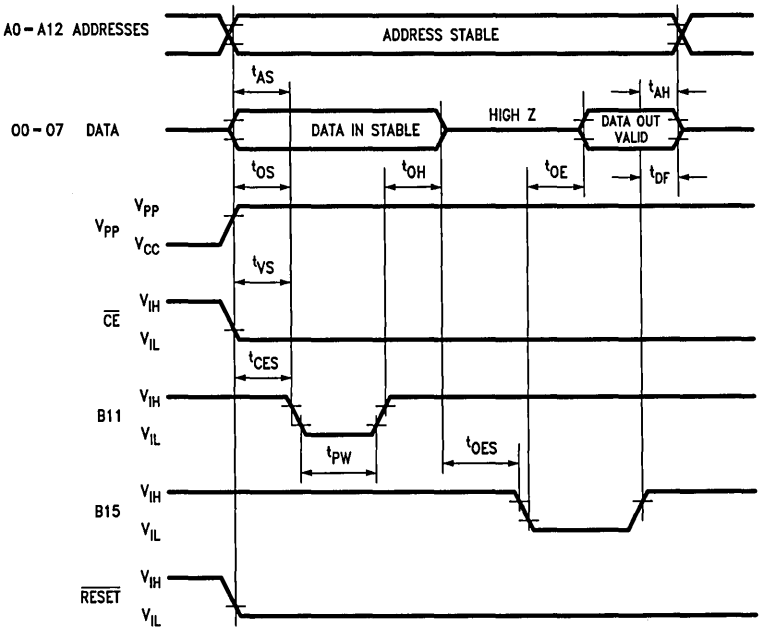
Note 2: V_{PP} must not be greater than 14V including overshoot. During $\overline{CE}$ = B11 = V_{IL}, V_{PP} must not be switched from 5V to 13.5V or vice-versa.

Note 3: During power up the B11 pin must be brought high ($\geq V_{IH}$) either coincident with or before power is applied to V_{PP}.

AC Electrical Characteristics $T_A = 25 \pm 5°C$, $V_{CC} = 5.50V \pm 5\%$, $V_{PP} = 13.5 \pm 0.5V$

Symbol	Parameter	Min	Typ	Max	Units
t_{AS}	Address Setup Time	2			μs
t_{CES}	$\overline{CE}$ Enable Setup Time	2			μs
t_{OES}	B_{11} Enable Setup Time	2			μs
t_{OS}	Data Setup Time	2			μs
t_{AH}	Address Hold Time	0			μs
t_{OH}	Data Hold Time	2			μs
t_{DF}	Chip Disable to Output Float Delay	0		130	ns
t_{OE}	Data Valid from B_{11} Enable			130	ns
t_{VS}	V_{PP} Setup Time	2			μs
t_{PW}	B_{15} Pulse Width	1	5	10	ms

Programming Waveform



TL/DD/10105-12

Programming Information (Continued)

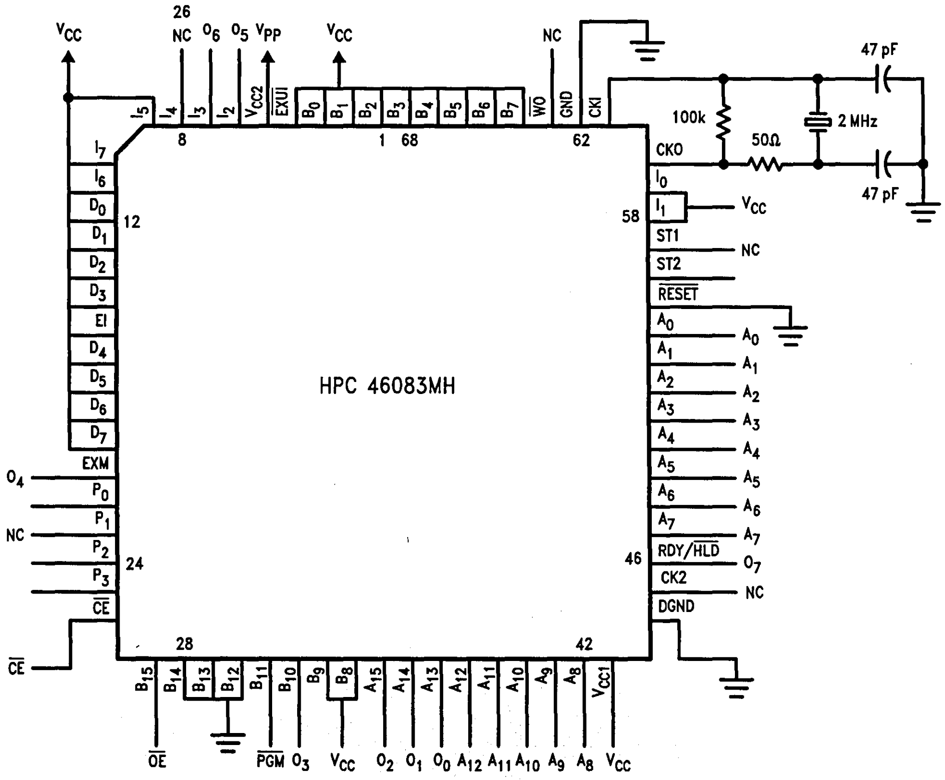
The following is the pin-connection list for programming the HPC Emulator.

HPC EMU		27C64	
Pin	Name	Pin	Name
1–4	B0–B2, EXUI	28	V_{CC}
5	V_{PP}	1	V_{PP}
6	I2	17	O5
7	I3	18	O6
8	I4	26	NC
9–20	I5–I7, D0–D7, EI	28	V_{CC}
21	EXM	16	O4
22–25	P0–P3	NC	
26	$\overline{\text{CE}}$	20	$\overline{\text{CE}}$
27	B15	22	$\overline{\text{OE}}$
28–30	B12–B14	14	GND
31	B11	27	$\overline{\text{PGM}}$
32	B10	15	O3
33,34	B9,B8	28	V_{CC}
35	A15	13	O2
36	A14	12	O1
37	A13	11	O0

HPC EMU		27C64	
Pin	Name	Pin	Name
38	A12	2	A12
39	A11	23	A11
40	A10	21	A10
41	A9	24	A9
42	A8	25	A8
43	V_{CC}	28	V_{CC}
44	DGND	14	GND
45	CK2	NC	
46	RDY/HLD	19	O7
47–54	A7–A0	3–10	A7–A0
55	RESET	14	GND
56,57	ST1,ST2	NC	
58,59	I0,I1	28	V_{CC}
60,61	CKO,CKI	Clock Circuit	NC
62	GND	14	GND
63	WO		NC
64–68	B3–B7	28	V_{CC}

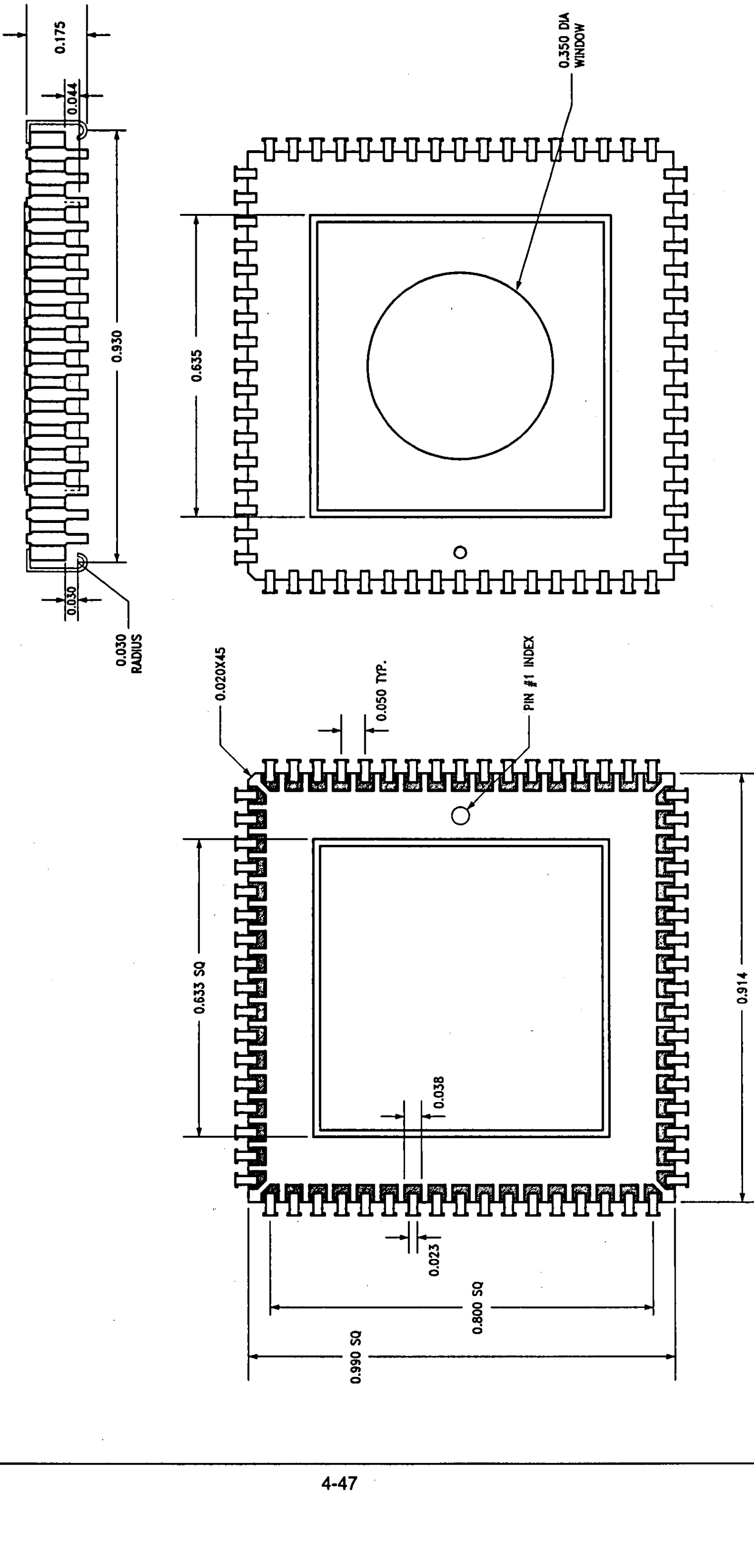
Attach a crystal circuit to CKI & CKO.

Programming Hookup to Program as 27C64



TL/DD/10105–13

Programming Information (Continued)



TL/DD/10105-14

National Semiconductor

PRELIMINARY

HPC16164/26164/36164/46164 HPC16104/26104/36104/46104 HPC16064/26064/36064/46064 HPC16004/26004/36004/46004 High-Performance microControllers with A/D

General Description

The HPC16164, HPC16104, HPC16064 and HPC16004 are members of the HPC™ family of High Performance microControllers. Each member of the family has the same core CPU with a unique memory and I/O configuration to suit specific applications. The HPC16164 and HPC16104 have 16k bytes of on-chip ROM. The HPC16104 and HPC16104 have no on-chip ROM and is intended for use with external memory. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC devices are complete microcomputers on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high performance applications. On-chip functions such as UART, up to eight 16-bit timers with 4 input capture registers, vectored interrupts, WATCHDOG logic and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64k bytes of external memory enables the HPC to be used in powerful applications typically performed by microprocessors and expensive peripheral chips. The term "HPC16164" is used throughout this datasheet to refer to the HPC16164, HPC16104, HPC16064 and HPC16004 devices unless otherwise specified.

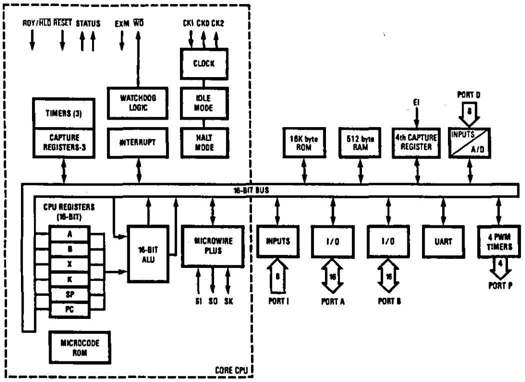
The HPC16164 and HPC16104 have, as an on-board peripheral, an 8-channel 8-bit Analog-to-Digital Converter. This A/D converter can operate in single-ended mode where the analog input voltage is applied across one of the eight input channels (D0–D7) and AGND. The A/D converter can also operate in differential mode where the analog input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel pairs in differential mode. The HPC16064 and HPC16004 do not have onboard A/D.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The IDLE and HALT modes provide further current savings. The HPC is available in 68-pin PLCC, LCC, LDCC, PGA and 84-pin TapePak® packages.

Features

- HPC family—core features:
  - — 16-bit architecture, both byte and word
  - — 16-bit data bus, ALU, and registers
  - — 64k bytes of external direct memory addressing
  - — FAST—200 ns for fastest instruction when using 20.0 MHz clock
  - — High code efficiency—most instructions are single byte
  - — 16 x 16 multiply and 32 x 16 divide
  - — Eight vectored interrupt sources
  - — Four 16-bit timer/counters with 4 synchronous outputs and WATCHDOG logic
  - — MICROWIRE/PLUS serial I/O interface
  - — CMOS—very low power with two power save modes: IDLE and HALT
- A/D—8-channel 8-bit analog-to-digital converter with conversion time minimum 6.6 μs for single conversion
- A/D—supports conversions in "quiet mode"

Block Diagram (HPC16164 with 16k ROM shown)



TL/DD/9682–1

Features (Continued)

- UART—full duplex, programmable baud rate
- Four additional 16-bit timer/counters with pulse width modulated outputs
- Four input capture registers
- 52 general purpose I/O lines (memory mapped)
- 16k bytes of ROM, 512 bytes of RAM on-chip
- ROMless version available (HPC16104)
- Commercial (0°C to +70°C), industrial (−40°C to +85°C), automotive (−40°C to +105°C) and military (−55°C to +125°C) temperature ranges

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD Rating	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

20 MHz

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, HPC46064/HPC46004, −40°C to +85°C for HPC36164/HPC36104, HPC36064/HPC36004, −40°C to +105°C for HPC26164/HPC26104, HPC26064/HPC26004, −55°C to +125°C for HPC16164/HPC16104, HPC16064/HPC16004

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz (Note 1)		40	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		15	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz, (Note 1)		3.5	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		1	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		300	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		100	μA
INPUT VOLTAGE LEVELS RESET, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	μA
I_{L2}	Input Leakage Current RDY/HLD, EXU1		−3	−50	μA
I_{L3}	Input Leakage Current B12		0.5	7	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA (Note 2)	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA (Note 2)		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	I_{OH} = −6 mA	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE® Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{RESET}$ = GND. I_{CC_3} is measured with NMI = V_{CC} and A/D inactive. CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns. V_{REF} = AGND = GND.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

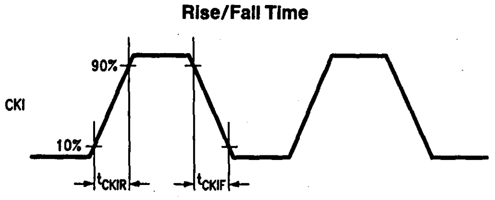
20 MHz

AC Electrical Characteristics

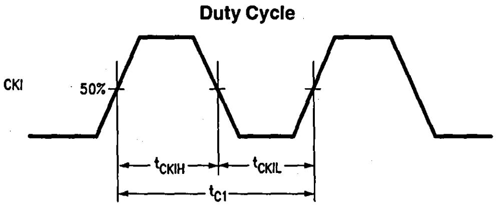
V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, HPC46064/HPC46004, −40°C to +85°C for HPC36164/HPC36104, HPC36064/HPC36004, −40°C to +105°C for HPC26164/HPC26104, HPC26064/HPC26004, −55°C to +125°C for HPC16164/HPC16104, HPC16064/HPC16004

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	20	MHz
$t_{C1} = 1/f_C$	Clock Period	50	500	ns
t_{CKIR} (Note 3)	CKI Rise Time		7	ns
t_{CKIF} (Note 3)	CKI Fall Time		7	ns
$[t_{CKIH}/(t_{CKIH} + t_{CKIL})]100$	Duty Cycle	45	55	%
$t_C = 2/f_C$	Timing Cycle	100		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns
t_{DC1C2R} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		45	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		45	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	18		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	20		ns
$t_{WAIT} = t_C$	Wait State Period	100		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.052	MHz
$t_{XIN} = t_C$	Pulse Width for Timer Inputs	100		ns
$f_{XOUT} = f_C/16$	Timer Output Frequency		1.25	MHz
$f_{MW} = f_C/19$	External MICROWIRE/PLUS Clock Input Frequency		1.052	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz

CKI Input Signal Characteristics



TL/DD/9682-34



TL/DD/9682-35

20 MHz

Read Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{\text{RD}}$ Falling Edge	20		ns
$t_{RW} = \frac{1}{2} t_C + WS - 10$	$\overline{\text{RD}}$ Pulse Width	140		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{\text{RD}}$	0	60	ns
$t_{ACC} = t_C + WS - 55$ (Note 2)	Address Valid to Input Data Valid		145	ns
$t_{RD} = \frac{1}{2} t_C + WS - 65$	$\overline{\text{RD}}$ Falling Edge to Input Data Valid		85	ns
$t_{RDA} = t_C - 5$	$\overline{\text{RD}}$ Rising Edge to Address Valid	95		ns

Write Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{\text{WR}}$ Falling Edge	45		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{\text{WR}}$ Pulse Width	160		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{\text{WR}}$	20		ns
$t_V = \frac{1}{2} t_C + WS - 5$	Data Valid before Rising Edge of $\overline{\text{WR}}$	145		ns

Note: Bus Output (Port A) $C_L = 100$ pF, CK2 Output $C_L = 50$ pF, other Outputs $C_L = 80$ pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC Specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurement will be skewed.

Note: WS = t_{WAIT} * number of pre-programmed wait states. Minimum and maximum values are calculated from maximum operating frequency with one (1) wait state pre-programmed.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO ***should not*** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not directly tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a very high confidence level.

Note 3: This is guaranteed by design and not tested.

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns
$t_{RWP} = t_C$	RDY Pulse Width	100		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{\text{HLD}}$ to Rising Edge of ALE	115		ns
$t_{HWP} = t_C + 10$	$\overline{\text{HLD}}$ Pulse Width	110		ns
$t_{HAD} = \frac{3}{4} t_C + 85$	Rising Edge on $\overline{\text{HLD}}$ to Rising Edge on $\overline{\text{HLDA}}$		160	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{\text{HLD}}$ to Falling Edge on $\overline{\text{HLDA}}$		200*	ns
$t_{BF} = \frac{1}{2} t_C + 66$	Data Valid after Falling Edge on $\overline{\text{HLDA}}$		116†	ns
$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable from Rising Edge of $\overline{\text{HLDA}}$	116†		ns

***Note:** t_{HAE} may be as long as ($3t_C + 4ws + 72t_C + 90$) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value is for the optimal case.

†**Note:** Due to emulation restrictions—actual limits will be better.

20 MHz

MICROWIRE/PLUS Timing

Symbol	Parameter	Min	Max	Units
t_{UWS}	MICROWIRE Setup			ns
Master	Time	100		
Slave		20		
t_{UWH}	MICROWIRE Hold			ns
Master	Time	20		
Slave		50		
t_{UWV}	MICROWIRE Output			ns
Master	Valid Time		50	
Slave			150	

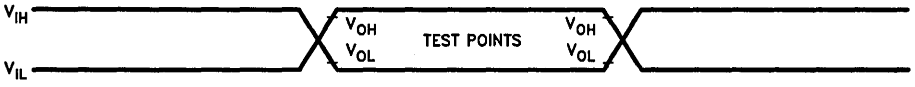
UPI Read/Write Timing

Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{URD}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{URD}$	10		ns
t_{RPW}	$\overline{URD}$ Pulse Width	100		ns
t_{OE}	$\overline{URD}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{URD}$ to Output Data Invalid (Note 4)	5	35	ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Rising Edge of $\overline{URD}$		70	ns
t_{WDW}	$\overline{UWR}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{UWR}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{UWR}$	15		ns
t_A	$\overline{WRRDY}$ Delay from Rising Edge of $\overline{UWR}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

Note 4: Guaranteed by design.

Input and Output for AC Tests



TL/DD/9682-40

Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

20 MHz

A/D Converter Specifications

V_{CC} = 5V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, −40°C to +85°C for HPC36164/HPC36104, −40°C to +105°C for HPC26164/HPC26104, −55°C to +125°C for HPC16164/HPC16104

Symbol	Parameter	Min	Max	Units
	Resolution		8	bits
f_{CCLK}	Clock Frequency (Note 4)	0.1	1.6	MHz
$t_{CON} = 8.5/f_{CCLK}$	Conversion Time (Note 3)	5.3		μs
V_{REF}	Reference Voltage Input (AGND = 0V)	3.0	V_{CC}	V
	Total Unadjusted Error (Note 1) (V_{REF} = 5.000V)		±½	LSB
R_{VREF}	Reference Input Resistance (Note 5)	1.6	4.8	kΩ
	DC Common Mode Error		±¼	LSB
	Power Supply Sensitivity (V_{REF} = 5.000V, V_{CC} = 5V ±10%)		±¼	LSB
	Voltage Reference Tolerance (V_{REF})		TBD	LSB
	Port D Input Capacitance (Note 5)		35	pF
	Analog Input Voltage Range (Note 2)	GND − 0.05	V_{CC} + 0.05	V
	On Channel Leakage		1	μA
	Off Channel Leakage		1	μA

Note 1: Total unadjusted error includes offset, full-scale, and multiplexer errors.

Note 2: 8 single-ended or 4 differential channels. Inherent sample and hold for single-ended inputs (GND = Pin 62).

Note 3: Conversion time does not include sample/hold time plus overhead. Sample and hold time is $2/f_{CCLK}$. Overhead is $1/f_{CCLK}$

Note 4: Clock supplied to A/D converter is derived from CKI. See A/D description for details.

Note 5: This is guaranteed by design and not tested.

30 MHz

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V
ESD Rating	2000V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

20 MHz

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, HPC46064/HPC46004, −40°C to +85°C for HPC36164/HPC36104, HPC36064/HPC36004, −40°C to +105°C for HPC26164/HPC26104, HPC26064/HPC26004, −55°C to +125°C for HPC16164/HPC16104, HPC16064/HPC16004

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 30.0 MHz (Note 1)		TBD	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		15	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 30.0 MHz, (Note 1)		TBD	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz, (Note 1)		2	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz, (Note 1)		400	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, (Note 1)		250	μA
INPUT VOLTAGE LEVELS RESET, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
ALL OTHER INPUTS					
V_{IH_2}	Logic High		0.7 V_{CC}		V
V_{IL_2}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	μA
I_{L2}	Input Leakage Current RDY/HLD, EXU1		−3	−50	μA
I_{L3}	Input Leakage Current B12		0.5	7	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA (Note 2)	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA (Note 2)		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (open drain) (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH4}	ST1 and ST2 Drive	I_{OH} = −6 mA	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5	V_{CC}	V
I_{OZ}	TRI-STATE® Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{RESET}$ = GND. I_{CC_3} is measured with NMI = V_{CC} and A/D inactive. CKI driven to V_{IH1} and V_{IL1} with rise and fall times less than 10 ns. V_{REF} = AGND = GND.

Note 2: This is guaranteed by design and not tested.

Note 3: Test duration is 100 ms.

30 MHz

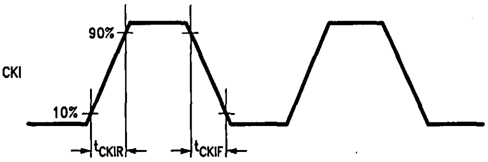
AC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, HPC46064/HPC46004, −40°C to +85°C for HPC36164/HPC36104, HPC36064/HPC36004, −40°C to +105°C for HPC26164/HPC26104, HPC26064/HPC26004, −55°C to +125°C for HPC16164/HPC16104, HPC16064/HPC16004

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2	30	MHz
$t_{C1} = 1/f_C$	Clock Period	33	500	ns
t_{CKIR} (Note 3)	CKI Rise Time		7	ns
t_{CKIF} (Note 3)	CKI Fall Time		7	ns
$[t_{CKIH}/(t_{CKIH} + t_{CKIL})]100$	Duty Cycle	45	55	%
$t_C = 2/f_C$	Timing Cycle	66		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	24		ns
t_{DC1C2R} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Rising Edge to ALE Rising Edge		37	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		37	ns
$t_{ST} = \frac{1}{4} t_C - 7$	Address Valid to ALE Falling Edge	9		ns
$t_{VP} = \frac{1}{4} t_C - 5$	Address Hold from ALE Falling Edge	11		ns
$t_{WAIT} = t_C$	Wait State Period	66		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1.579	MHz
$t_{XIN} = t_C$	Pulse Width for Timer Inputs	66		ns
$f_{XOUT} = f_C/16$	Timer Output Frequency		1.875	MHz
$f_{MW} = f_C/19$	External MICROWIRE/PLUS Clock Input Frequency		1.579	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		3.75	MHz

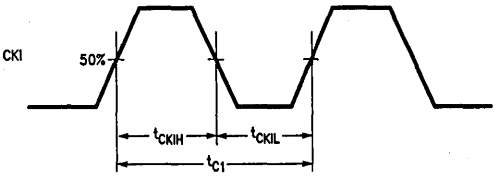
CKI Input Signal Characteristics

Rise/Fall Time



TL/DD/9682-34

Duty Cycle



TL/DD/9682-35

30 MHz

Read Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{4} t_C - 5$	ALE Falling Edge to $\overline{RD}$ Falling Edge	12		ns
$t_{RW} = \frac{1}{2} t_C + WS - 14$	$\overline{RD}$ Pulse Width	85		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold after Rising Edge of $\overline{RD}$	0	35	ns
$t_{ACC} = t_C + WS - 32$ (Note 2)	Address Valid to Input Data Valid		100	ns
$t_{RD} = \frac{1}{2} t_C + WS - 39$	$\overline{RD}$ Falling Edge to Input Data Valid		60	ns
$t_{RDA} = t_C - 5$	$\overline{RD}$ Rising Edge to Address Valid	61		ns

Write Cycle Timing

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 5$	ALE Falling Edge to $\overline{WR}$ Falling Edge	28		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{WR}$ Pulse Width	101		ns
$t_{HW} = \frac{1}{4} t_C - 10$	Data Hold after Rising Edge of $\overline{WR}$	7		ns
$t_V = \frac{1}{2} t_C + WS - 5$	Data Valid before Rising Edge of $\overline{WR}$	94		ns

Note: Bus Output (Port A) $C_L = 100$ pF, CK2 Output $C_L = 50$ pF, other Outputs $C_L = 80$ pF. AC parameters are tested using DC Characteristics Inputs and non CMOS Outputs. Measurement of AC Specifications is done with external clock driving CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF or AC measurement will be skewed.

Note: WS = t_{WAIT} * number of pre-programmed wait states. Minimum and maximum values are calculated from maximum operating frequency with one (1) wait state pre-programmed.

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO ***should not*** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

Note 2: These are not directly tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a very high confidence level.

Note 3: This is guaranteed by design and not tested.

Ready/Hold Timing

Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of RDY		75	ns
$t_{RWP} = t_C$	RDY Pulse Width	100		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	115		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	110		ns
$t_{HAD} = \frac{3}{4} t_C + 85$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		160	ns
$t_{HAE} = t_C + 85$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		151*	ns
t_{BF}	Data Valid after Falling Edge on $\overline{HLDA}$	0		ns
$t_{BE} = \frac{1}{2} t_C$	Bus Enable from Rising Edge of $\overline{HLDA}$	33		ns

***Note:** t_{HAE} may be as long as ($3t_C + 4ws + 72t_C + 90$) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value is for the optimal case.

30 MHz

MICROWIRE/PLUS Timing

Symbol	Parameter	Min	Max	Units
t_{UWS} Master Slave	MICROWIRE Setup Time	 100 20		ns
t_{UWH} Master Slave	MICROWIRE Hold Time	 20 50		ns
t_{UWV} Master Slave	MICROWIRE Output Valid Time		 50 150	ns

UPI Read/Write Timing

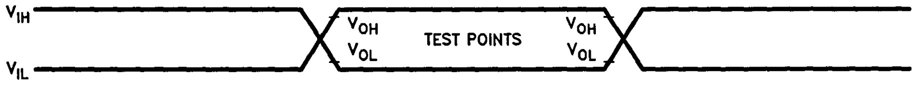
Symbol	Parameter	Min	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{\text{URD}}$	10		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{\text{URD}}$	10		ns
t_{RPW}	$\overline{\text{URD}}$ Pulse Width	100		ns
t_{OE}	$\overline{\text{URD}}$ Falling Edge to Output Data Valid	0	60	ns
t_{OD}	Rising Edge of $\overline{\text{URD}}$ to Output Data Invalid (Note 4)	5	35	ns
t_{DRDY}	$\overline{\text{RDRDY}}$ Delay from Rising Edge of $\overline{\text{URD}}$		70	ns
t_{WDW}	$\overline{\text{UWR}}$ Pulse Width	40		ns
t_{UDS}	Input Data Valid before Rising Edge of $\overline{\text{UWR}}$	10		ns
t_{UDH}	Input Data Hold after Rising Edge of $\overline{\text{UWR}}$	15		ns
t_A	$\overline{\text{WRRDY}}$ Delay from Rising Edge of $\overline{\text{UWR}}$		70	ns

Note: Bus Output (Port A) C_L = 100 pF, CK2 Output C_L = 50 F, other Outputs C_L = 80 pF.

Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{OL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

Note 4: Guaranteed by design.

Input and Output for AC Tests



TL/DD/9682-40

30 MHz

A/D Converter Specifications

V_{CC} = 5V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46164/HPC46104, −40°C to +85°C for HPC36164/HPC36104, −40°C to +105°C for HPC26164/HPC26104, −55°C to +125°C for HPC16164/HPC16104

Symbol	Parameter	Min	Max	Units
	Resolution		8	bits
f_{CCLK}	Clock Frequency (Note 4)	0.1	1.6	MHz
$t_{CON} = 8.5/f_{CCLK}$	Conversion Time (Note 3)	5.3		μs
V_{REF}	Reference Voltage Input (AGND = 0V)	3.0	V_{CC}	V
	Total Unadjusted Error (Note 1) (V_{REF} = 5.000V)		±½	LSB
R_{VREF}	Reference Input Resistance (Note 5)	1.6	4.8	kΩ
	DC Common Mode Error		±¼	LSB
	Power Supply Sensitivity (V_{REF} = 5.000V, V_{CC} = 5V ±10%)		±¼	LSB
	Voltage Reference Tolerance (V_{REF})		TBD	LSB
	Port D Input Capacitance (Note 5)		35	pF
	Analog Input Voltage Range (Note 2)	GND − 0.05	V_{CC} + 0.05	V
	On Channel Leakage		1	μA
	Off Channel Leakage		1	μA

Note 1: Total unadjusted error includes offset, full-scale, and multiplexer errors.

Note 2: 8 single-ended or 4 differential channels. Inherent sample and hold for single-ended inputs (GND = Pin 62).

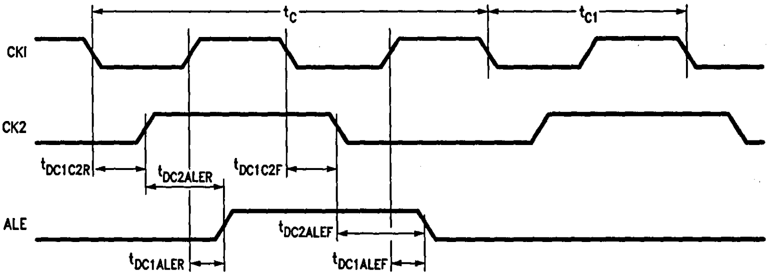
Note 3: Conversion time does not include sample/hold time. Sample and hold time is $2/f_{CCLK}$.

Note 4: Clock supplied to A/D converter is derived from CKI. See A/D description for details.

Note 5: This is guaranteed by design and not tested.

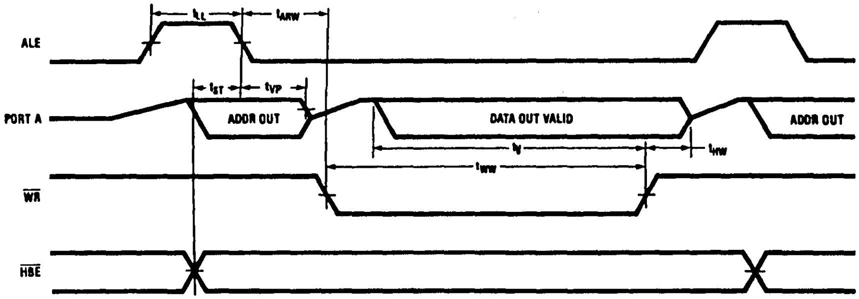
Timing Waveforms

CKI, CK2, ALE Timing Diagram



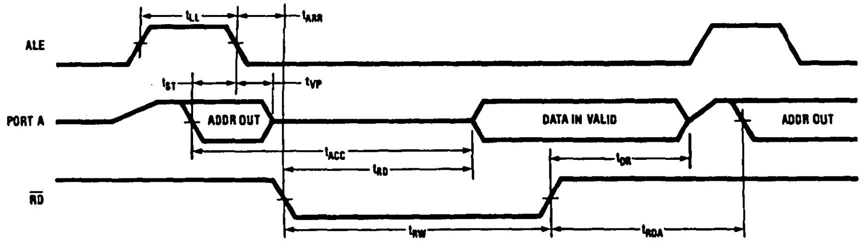
TL/DD/9682-2

Timing Waveforms (Continued)



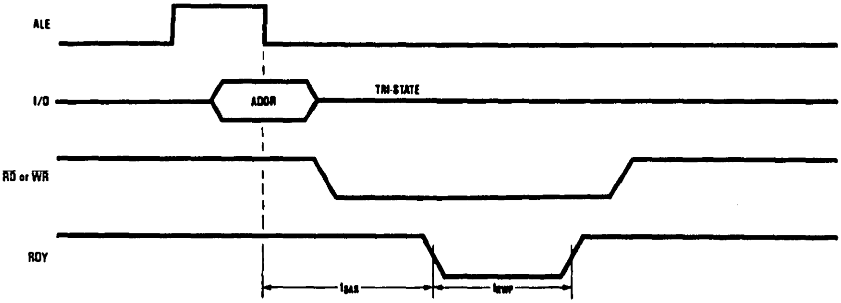
TL/DD/9682-3

FIGURE 1. Write Cycle



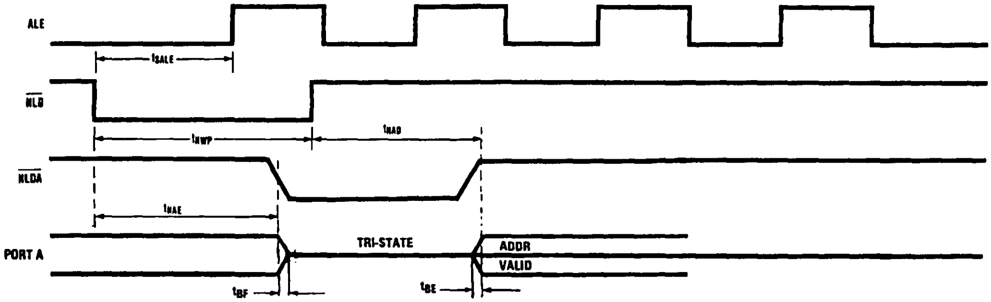
TL/DD/9682-4

FIGURE 2. Read Cycle



TL/DD/9682-5

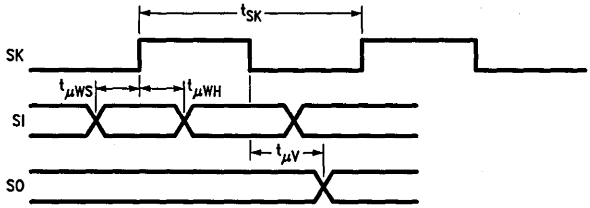
FIGURE 3. Ready Mode Timing



TL/DD/9682-6

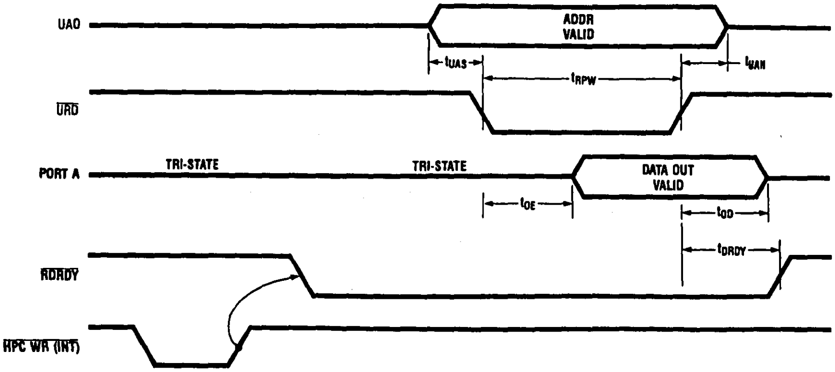
FIGURE 4. Hold Mode Timing

Timing Waveforms (Continued)



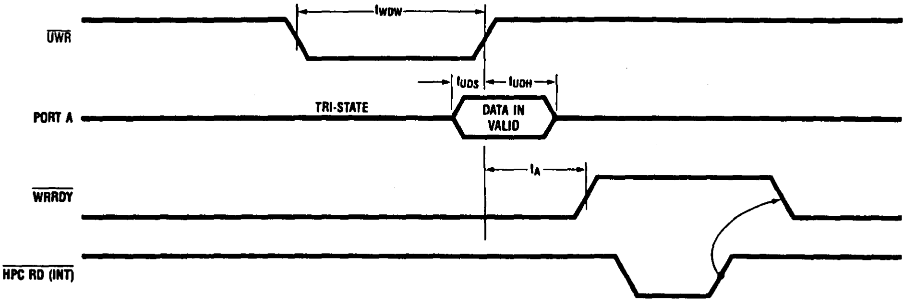
TL/DD/9682-39

FIGURE 5. MICROWIRE Setup/Hold Timing



TL/DD/9682-9

FIGURE 6. UPI Read Timing



TL/DD/9682-10

FIGURE 7. UPI Write Timing

Pin Descriptions

The HPC16164 is available in 68-pin PLCC, LCC, LDCC, PGA, and TapePak packages.

I/O PORTS

Port A is a 16-bit bidirectional I/O port with a data direction register to enable each separate pin to be individually defined as an input or output. When accessing external memory, port A is used as the multiplexed address/data bus.

Port B is a 16-bit port with 12 bits of bidirectional I/O similar in structure to Port A. Pins B10, B11, B12 and B15 are general purpose outputs only in this mode. Port B may also be configured via a 16-bit function register BFUN to individually allow each pin to have an alternate function.

B0:	TDX	UART Data Output
B1:		
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T310	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{HLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	UA0	Address 0 Input for UPI Mode
B11:	$\overline{\text{WRRDY}}$	Write Ready Output for UPI Mode
B12:		
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RDRDY}}$	Read Ready Output for UPI Mode

When accessing external memory, four bits of port B are used as follows:

B10:	ALE	Address Latch Enable Output
B11:	$\overline{\text{WR}}$	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output/Input (sampled at reset)
B15:	$\overline{\text{RD}}$	Read Output

Port I is an 8-bit input port that can be read as general purpose inputs and is also used for the following functions:

I0:		
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture/$\overline{\text{URD}}$
I3:	INT3	Maskable Interrupt/Input Capture/$\overline{\text{UWR}}$
I4:	INT4	Maskable Interrupt/Input Capture
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:		

Port D is an 8-bit input port that can be used as general purpose digital inputs or as analog channel inputs for the A/D converter. These functions of Port D are mutually exclusive and under the control of software.

Port P is a 4-bit output port that can be used as general purpose data, or selected to be controlled by timers 4 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V_{CC1} and V_{CC2}	Positive Power Supply
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI	The Chip System Clock Input
CKO	The Chip System Clock Output (inversion of CKI)

Pins CKI and CKO are usually connected across an external crystal.

CK2	Clock Output (CKI divided by 2)

OTHER PINS

$\overline{\text{WO}}$ This is an active low open drain output that signals an illegal situation has been detected by the Watch Dog logic.

ST1 Bus Cycle Status Output: indicates first opcode fetch.

ST2 Bus Cycle Status Output: indicates machine states (skip, interrupt and first instruction cycle).

$\overline{\text{RESET}}$ is an active low input that forces the chip to restart and sets the ports in a TRI-STATE® mode.

RDY/$\overline{\text{HLD}}$ has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.

VREF A/D converter reference voltage input.

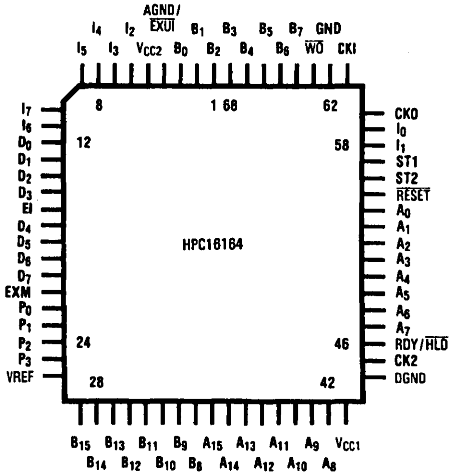
EXM External memory enable (active high) disables internal ROM and maps it to external memory.

EI External interrupt with vector address FFF1:FFF0. (Rising/falling edge or high/low level sensitive). Alternately can be configured as 4th input capture.

AGND/$\overline{\text{EXUI}}$ has two uses, selected by a software bit. It can be an external active low interrupt which is internally OR'ed with the UART interrupt with vector address FFF3:FFF2 or it can be the analog ground for the A/D converter.

Connection Diagrams

Plastic, Leadless and Leaded Chip Carriers

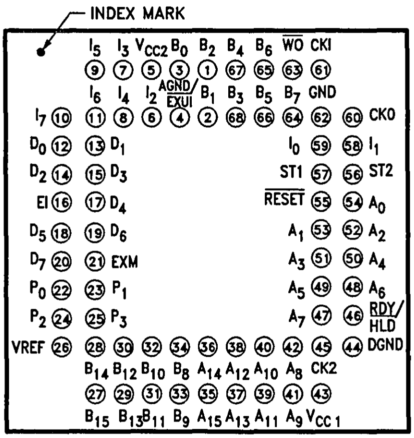

AGND/EXUI
I4 I2 B1 B3 B5 B7 GND
I5 I3 VCC2 B0 B2 B4 B6 WO CKI
8 1 68 62
12 58
HPC16164
24 46
28 42
I7 I6 D0 D1 D2 D3 EI D4 D5 D6 D7 EXM P0 P1 P2 P3 VREF
CKO I0 I1 ST1 ST2 RESET A0 A1 A2 A3 A4 A5 A6 A7 RDY/HLD CK2 DGND
B15 B13 B11 B9 A15 A13 A11 A9 VCC1
B14 B12 B10 B8 A14 A12 A10 A8

TL/DD/9682-11

Top View

Order Number HPC16164E, EL or V
See NS Package Number E68B, EL68A or V68A

Pin Grid Array Pinout

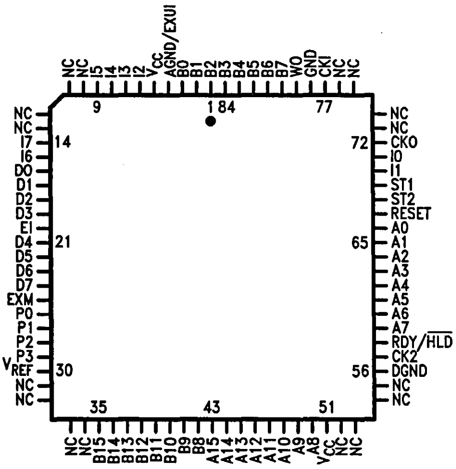

INDEX MARK
I5 I3 VCC2 B0 B2 B4 B6 WO CKI
9 7 5 3 1 67 65 63 61
I6 I4 I2 AGND/EXUI B1 B3 B5 B7 GND
I7 10 11 8 6 4 2 68 66 64 62 60 CKO
D0 12 13 D1 I0 59 58 I1
D2 14 15 D3 ST1 57 56 ST2
EI 16 17 D4 RESET 55 54 A0
D5 18 19 D6 A1 53 52 A2
D7 20 21 EXM A3 51 50 A4
P0 22 23 P1 A5 49 48 A6
P2 24 25 P3 A7 47 46 RDY/HLD
VREF 26 28 30 32 34 36 38 40 42 45 44 DGND
B14 B12 B10 B8 A14 A12 A10 A8 CK2
27 29 31 33 35 37 39 41 43
B15 B13 B11 B9 A15 A13 A11 A9 VCC1

TL/DD/9682-12

Top View
(looking down on component side of PC Board)

Order Number HPC16164U
See NS Package Number U68A

TapePak Package


NC NC I5 I4 I3 I2 VCC AGND/EXUI B0 B1 B2 B3 B4 B5 B6 B7 WO GND CKI NC NC
9 1 84 77
14 72
21 65
30 56
35 43 51
NC NC I7 I6 D0 D1 D2 D3 EI D4 D5 D6 D7 EXM P0 P1 P2 P3 VREF NC NC
NC NC CKO I0 I1 ST1 ST2 RESET A0 A1 A2 A3 A4 A5 A6 A7 RDY/HLD CK2 DGND NC NC
NC NC B15 B14 B13 B12 B11 B10 B9 B8 A15 A14 A13 A12 A11 A10 A9 A8 VCC NC NC

TL/DD/9682-36

Top View

Order Number HPC16164TP
Available in TapePak

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC16164 and HPC16104 have four operating modes. The ROMless HPC16104 has one mode of operation. The various modes of operation are determined by the state of both the EXM pin and the EA bit in the PSW register. The state of the EXM pin determines whether on-chip ROM will be accessed or external memory will be accessed within the address range of the on-chip ROM. The on-chip ROM range of the HPC16164 is C000 to FFFF (16k bytes). The HPC16104 has no on-chip ROM and is intended for use with external memory for program storage. A logic "0" state on the EXM pin will cause the HPC device to address on-chip ROM when the Program Counter (PC) contains addresses within the on-chip ROM address range. A logic "1" state on the EXM pin will cause the HPC device to address memory that is external to the HPC when the PC contains on-chip ROM addresses. The EXM pin should always be pulled high (logic "1") on the HPC16104 because no on-chip ROM is available. The function of the EA bit is to determine the legal addressing range of the HPC device. A logic "0" state in the EA bit of the PSW register does two things—addresses are limited to the on-chip ROM range and on-chip RAM and Register range, and the "illegal address detection" feature of the Watchdog logic is engaged. A logic "1" in the EA bit enables accesses to be made anywhere within the 64k byte address range and the "illegal address detection" feature of the Watchdog logic is disabled. The EA bit should be set to "1" by software when using the HPC16104 to disable the "illegal address detection" feature of Watchdog.

All HPC devices can be used with external memory. External memory may be any combination of RAM and ROM. Both 8-bit and 16-bit external data bus modes are available. Upon entering an operating mode in which external memory is used, port A becomes the Address/Data bus. Four pins of port B become the control lines ALE, $\overline{\text{RD}}$, $\overline{\text{WR}}$ and $\overline{\text{HBE}}$. The High Byte Enable pin ($\overline{\text{HBE}}$) is used in 16-bit mode to select high order memory bytes. The $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals are only generated if the selected address is off-chip. The 8-bit mode is selected by pulling $\overline{\text{HBE}}$ high at reset. If $\overline{\text{HBE}}$ is left floating or connected to a memory device chip select at reset, the 16-bit mode is entered. The following sections describe the operating modes of the HPC16164 and HPC16104.

Note: The HPC devices use 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.

HPC16164/HPC16064 Operating Modes

SINGLE CHIP NORMAL MODE

In this mode, the HPC16164/HPC16064 functions as a self-contained microcomputer (see *Figure 11*) with all memory (RAM and ROM) on-chip. It can address internal memory only, consisting of 16k bytes of ROM (C000 to FFFF) and 512 bytes of on-chip RAM and Registers (0000 to 02FF). The "illegal address detection" feature of the Watchdog is enabled in the Single-Chip Normal mode and a Watchdog Output ($\overline{\text{WO}}$) will occur if an attempt is made to access addresses that are outside of the on-chip ROM and RAM range of the device. Ports A and B are used for I/O functions and not for addressing external memory. The EXM pin and the EA bit of the PSW register must both be logic "0" to enter the Single-Chip Normal mode.

EXPANDED NORMAL MODE

The Expanded Normal mode of operation enables the HPC16164 to address external memory in addition to the on-chip ROM and RAM (see Table II). Watchdog illegal address detection is disabled and memory accesses may be made anywhere in the 64k byte address range without triggering an illegal address condition. The Expanded Normal mode is entered with the EXM pin pulled low (logic "0") and setting the EA bit in the PSW register to "1".

SINGLE-CHIP ROMLESS MODE

In this mode, the on-chip mask programmed ROM of the HPC16164 is not used. The address space corresponding to the on-chip ROM is mapped into external memory so 16k of external memory may be used with the HPC16164 (see Table II). The Watchdog circuitry detects illegal addresses (addresses not within the on-chip ROM and RAM range). The Single-Chip ROMless mode is entered when the EXM pin is pulled high (logic "1") and the EA bit is logic "0".

EXPANDED ROMLESS MODE

This mode of operation is similar to Single-Chip ROMless mode in that no on-chip ROM is used, however, a full 64k bytes of external memory may be used. The "illegal address detection" feature of Watchdog is disabled. The EXM pin must be pulled high (logic "1") and the EA bit in the PSW register set to "1" to enter this mode.

TABLE II. HPC16164 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Single-Chip Normal	0	0	C000:FFFF on-chip
Expanded Normal	0	1	C000:FFFF on-chip 0300:BFFF off-chip
Single-Chip ROMless	1	0	C000:FFFF off-chip
Expanded ROMless	1	1	0300:FFFF off-chip

Note: In all operating modes, the on-chip RAM and Registers (0000:02FF) may be accessed.

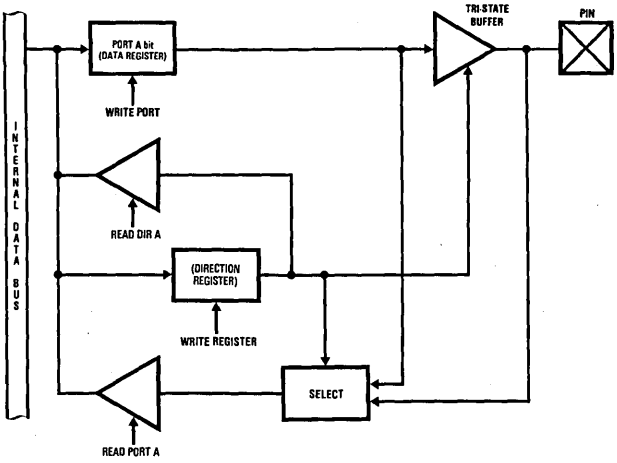
Ports A & B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 7*), consists of a data register and a direction register. Port B (see *Figures 8, 9* and *10*) has an alternate function register in addition to the data and direction registers. All the control registers are read/write registers.

The associated direction registers allow the port pins to be individually programmed as inputs or outputs. Port pins selected as inputs, are placed in a TRI-STATE mode by resetting corresponding bits in the direction register.

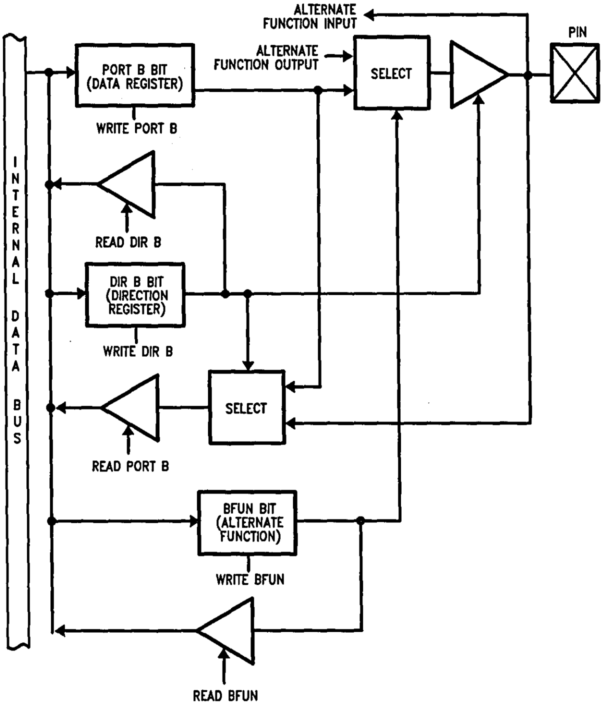
A write operation to a port pin configured as an input causes the value to be written into the data register, a read operation returns the value of the pin. Writing to port pins configured as outputs causes the pins to have the same value, reading the pins returns the value of the data register.

Primary and secondary functions are multiplexed onto Port B through the alternate function register (BFUN). The secondary functions are enabled by setting the corresponding bits in the BFUN register.



TL/DD/9682-13

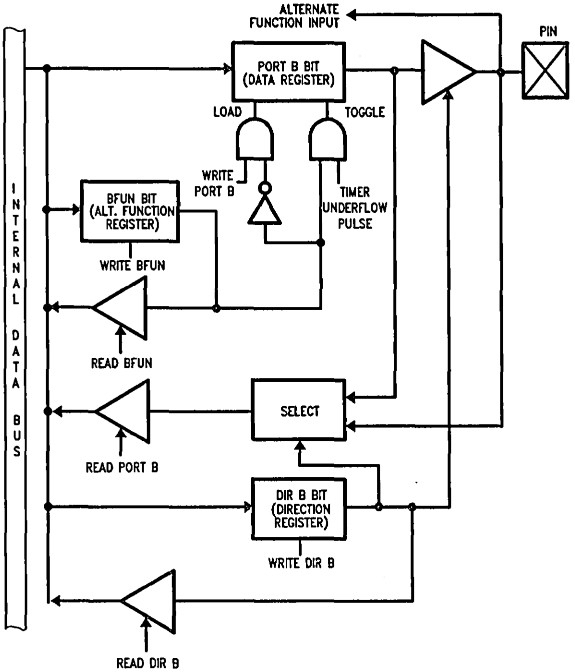
FIGURE 7. Port A: I/O Structure



TL/DD/9682-14

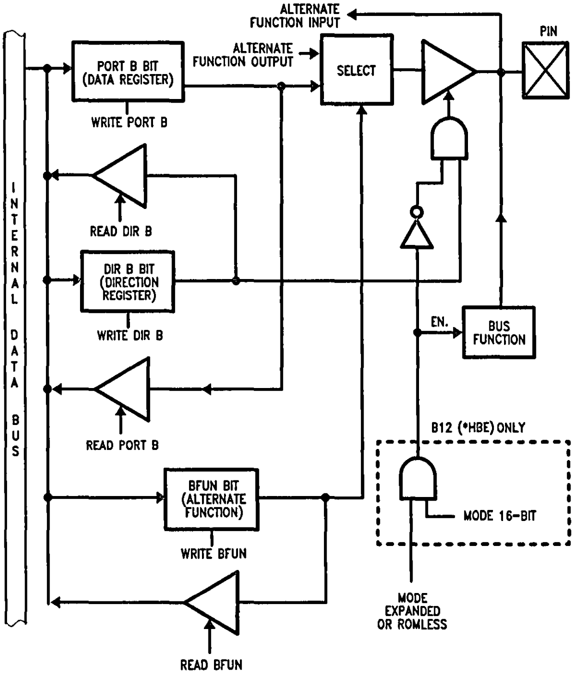
FIGURE 8. Structure of Port B Pins B0, B1, B2, B5, B6 and B7 (Typical Pins)

Ports A & B (Continued)



TL/DD/9682–15

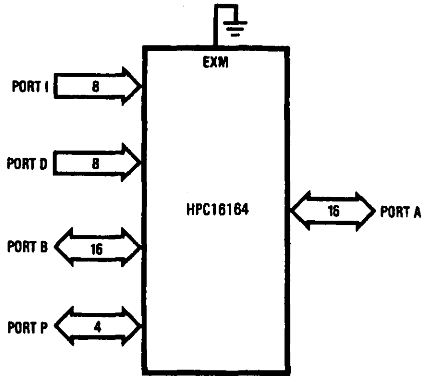
FIGURE 9. Structure of Port B Pins B3, B4, B8, B9, B13 and B14 (Timer Synchronous Pins)



TL/DD/9682–16

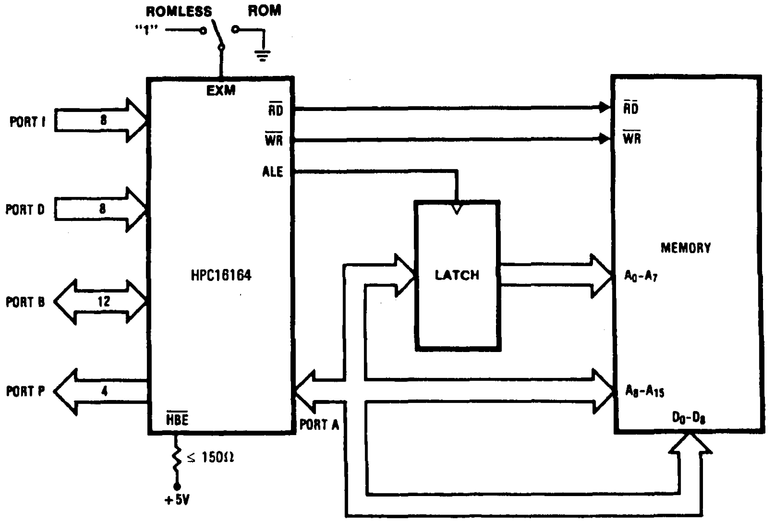
FIGURE 10. Structure of Port B Pins B10, B11, B12 and B15 (Pins with Bus Control Roles)

HPC16164 Operating Modes (Continued)



TL/DD/9682-17

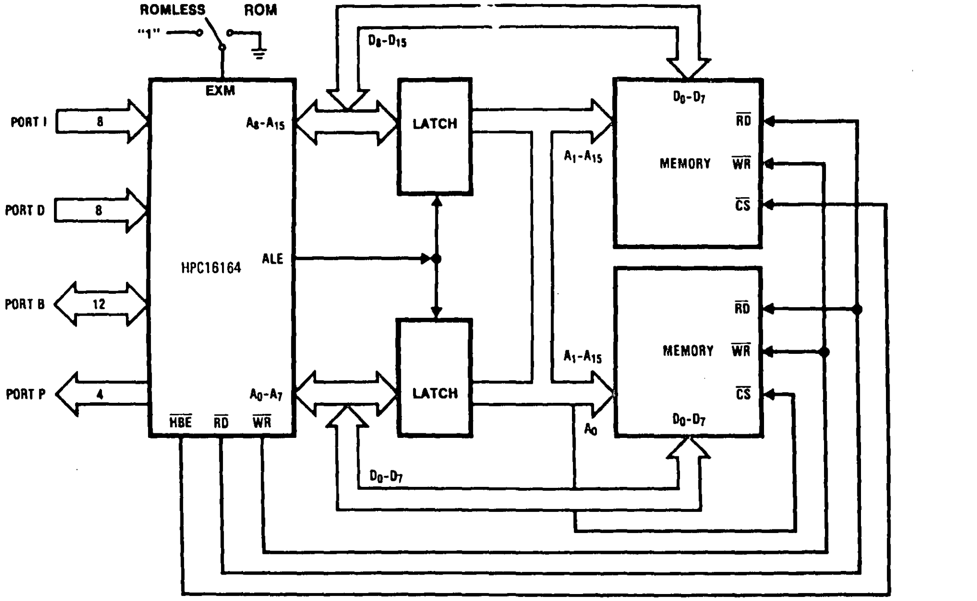
FIGURE 11. Single-Chip Mode



TL/DD/9682-18

FIGURE 12. 8-Bit External Memory

HPC16164 Operating Modes (Continued)



TL/DD/9682-19

FIGURE 13. 16-Bit External Memory

HPC16104/HPC16004 Operating Modes

EXPANDED ROMLESS MODE (HPC16104/HPC16004)

Because the HPC16104 has no on-chip ROM, it has only one mode of operation, the Expanded ROMless Mode. The EXM pin must be pulled high (logic "1") on power up, the EA bit in the PSW register should be set to a "1". The HPC16104/HPC16004 is a ROMless device and is intended for use with external memory. The external memory may be any combination of ROM and RAM. Up to 64k bytes of external memory may be accessed. It is necessary to vector on reset to an address between C000 and FFFF, therefore the user should have external memory at these addresses. The EA bit in the PSW register must immediately be set to "1" at the beginning of the user's program to disable illegal address detection in the Watchdog logic.

TABLE III. HPC16104 Operating Modes

Operating Mode	EXM Pin	EA Bit	Memory Configuration
Expanded ROMless	1	1	0300:FFFF off-chip

Note: The on-chip RAM and Registers (0000:02FF) of the HPC16104 may be accessed at all times.

Wait States

The internal ROM can be accessed at the maximum operating frequency with one wait state. With 0 wait states, internal ROM accesses are limited to ⅔ f_C max.

The HPC16164 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals.

Power Save Modes

Two power saving modes are available on the HPC16164: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, all on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16164 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16164 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the $\overline{\text{RESET}}$ or the NMI. The $\overline{\text{RESET}}$ input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16164 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. As with the HALT

Power Save Modes (Continued)

mode, the processor is returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization. A timer T0 overflow will also cause the HPC16164 to resume normal operation.

HPC16164 Interrupts

Complex interrupt handling is easily accomplished by the HPC16164's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table IV.

TABLE IV. Interrupts

Vector Address	Interrupt Source	Arbitration Ranking
$FFFF:FFFE	RESET	0
$FFFD:FFFC	Nonmaskable external on rising edge of I1 pin	1
$FFFB:FFFA	External interrupt on I2 pin	2
$FFF9:FFF8	External interrupt on I3 pin	3
$FFF7:FFF6	External interrupt on I4 pin	4
$FFF5:FFF4	Overflow on internal timers	5
$FFF3:FFF2	Internal by on-board peripherals or external on EXUI	6
$FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC16164 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. The arbitration ranking is given in Table IV. The interrupt on Reset has the highest rank and is serviced first.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately.

RESET and EXUI are level-LOW-sensitive interrupts and EI is programmable for edge-(RISING or FALLING) or level-(HIGH or LOW) sensitivity. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 and I4 can be software selected to be rising or falling edge. External interrupt (EXUI) is shared with the on-board peripherals, UART and A/D. The EXUI interrupt is level-LOW-sensitive. To select this interrupt, disable the ERI and ETI UART interrupts by resetting these enable bits in the ENUI register and disable the A/D function by resetting the ADEN bit in the A/D control register #3 (CR3). To select the on-board peripherals interrupt, leave this pin floating or tie it high if the A/D function is disabled. If the A/D function is enabled, this pin becomes the analog ground (AGND).

Interrupt Control Registers

The HPC16164 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16164 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

The NMI bit is read only and I2, I3, and I4 are designed as to only allow a zero to be written to the pending bit (writing a one has no affect). A LOAD IMMEDIATE instruction is to be the only instruction used to clear a bit or bits in the IRPD register. This allows a mask to be used, thus ensuring that the other pending bits are not affected.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable bit (GIE) is copied into the CGIE bit of the PSW register; it is then reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack and re-enable interrupts if the CGIE bit is set, or RET to just pop the stack if the CGIE bit is clear, and then returns to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 14* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A and B in the TRI-STATE condition and Port P in the LOW state. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location (which must correspond to an on board location). The Reset vector address must be between C000 and FFFF when using the HPC16104.

Servicing the Interrupts (Continued)

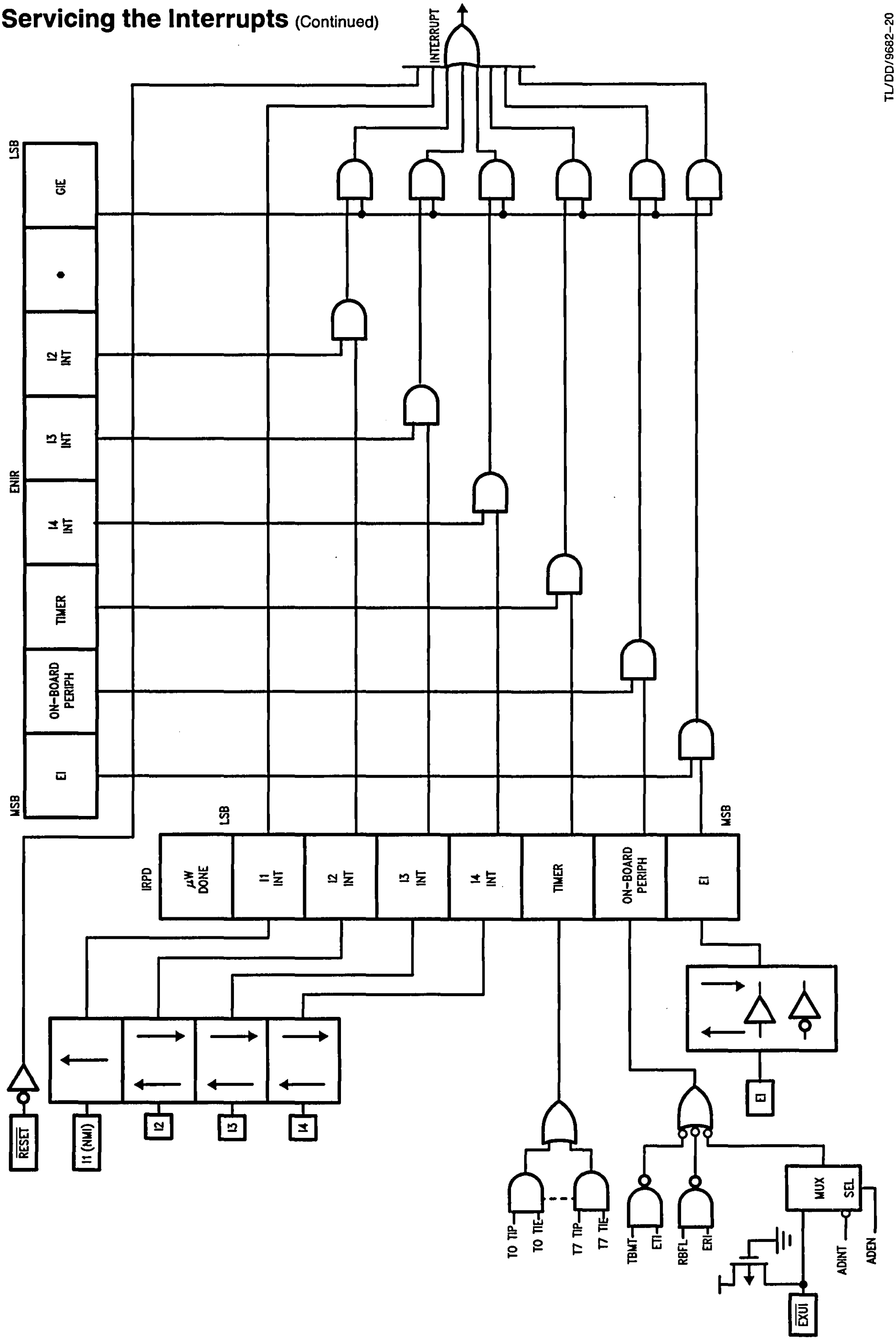


FIGURE 14. Block Diagram of Interrupt Logic

Timer Overview

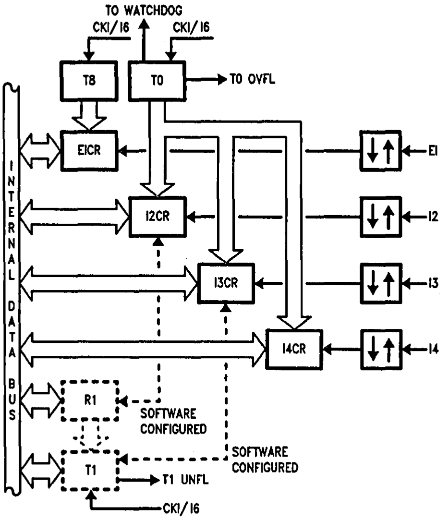
The HPC16164 contains a powerful set of flexible timers enabling the HPC16164 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16164 contains nine 16-bit timers. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watchdog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register TMMODE configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 15*).

The HPC16164 provides an additional 16-bit free running timer, T8, with associated input capture register EICR (External Interrupt Capture Register) and Configuration Register, EICON. EICON is used to select the mode and edge of the EI pin. EICR is a 16-bit capture register which records the value of T8 (which is identical to T0) when a specific event occurs on the EI pin.

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 16*).

The timers T1 through T7 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

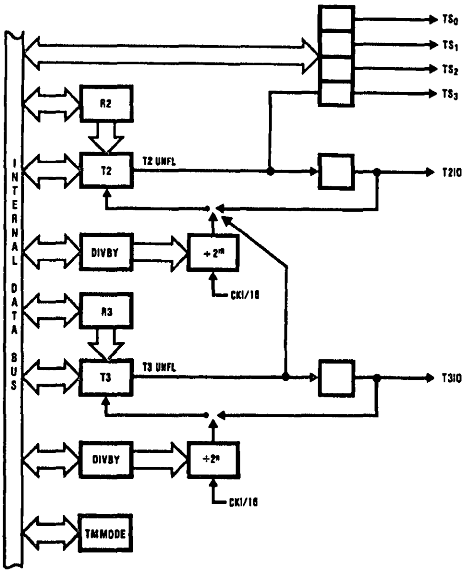


TL/DD/9682-21

FIGURE 15. Timers T0, T1 and T8 with Four Input Capture Registers

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16164 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 16*).

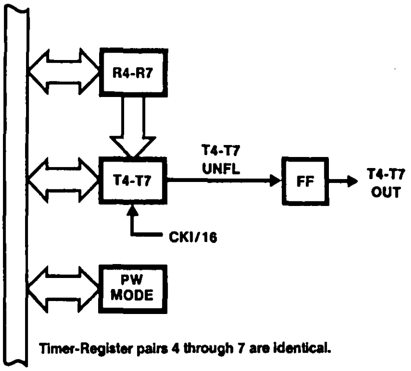


TL/DD/9682-22

FIGURE 16. Timers T2-T3 Block

Timer Overview (Continued)

Timer/register pairs 4–7 form four identical units which can generate synchronous outputs on port P (see *Figure 17*). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to ½ the frequency of the source used for clocking the timer.



Timer-Register pairs 4 through 7 are identical.

TL/DD/9682-23

FIGURE 17. Timers T4–T7 Block

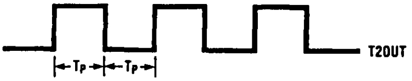
Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3. The control register PWMODE similarly programs the pulse width timers T4 through T7 by allowing them to be started, stopped, and to latch and enable interrupts on underflows. The PORTP register contains bits to preset the outputs and enable the synchronous timer output functions.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16164.

Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.



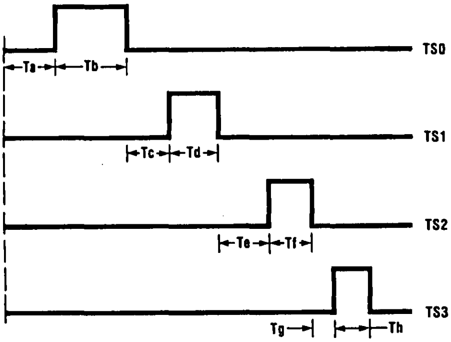
TL/DD/9682-24

FIGURE 18. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 19* is an example of synchronous pulse train generation.

Watchdog Logic

The Watchdog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watchdog logic are potentially infinite loops and illegal addresses. Should the Watchdog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. An illegal condition also occurs when the processor generates an illegal address when in the Single-Chip modes.* Any illegal condition forces the Watchdog Output ($\overline{\text{WO}}$) pin low. The $\overline{\text{WO}}$ pin is an open drain output and can be connected to the $\overline{\text{RESET}}$ or NMI inputs or to the users external logic.



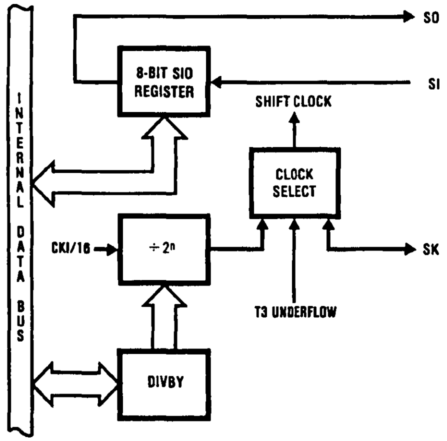
TL/DD/9682-25

FIGURE 19. Synchronous Pulse Generation

*Note: See Operating Modes for details.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 20*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.



TL/DD/9682-26

FIGURE 20. MICROWIRE/PLUS

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., A/D converters, display drivers, EEPROMs).

MICROWIRE/PLUS Operation

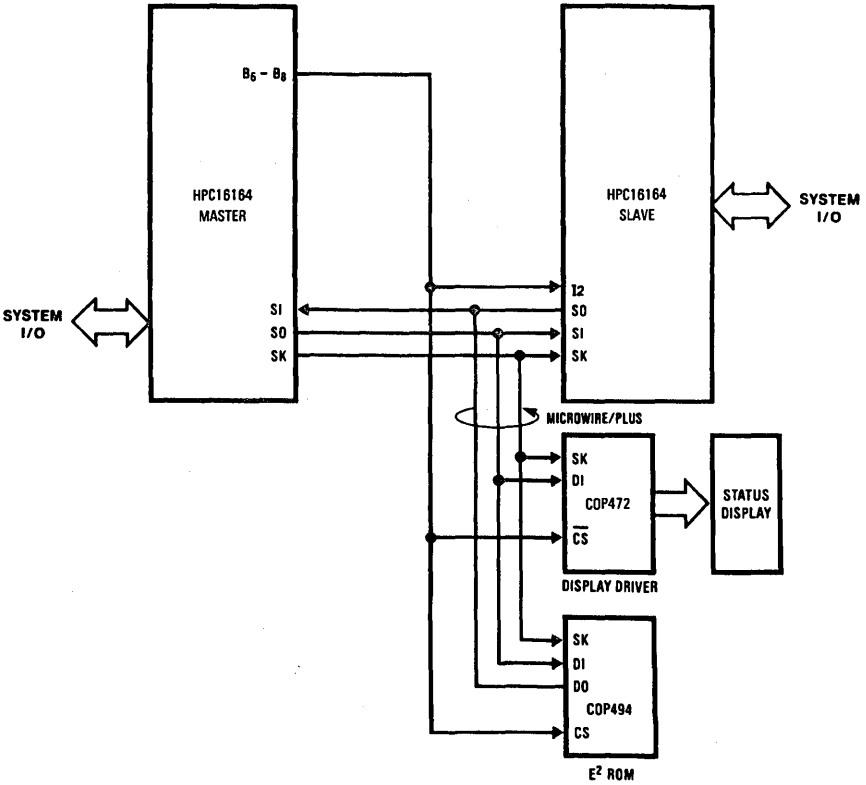
The HPC16164 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16164 is the master or slave. The shift clock is generated when the HPC16164 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16164 is configured as a slave. When the HPC16164 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 15 selectable steps from 64 Hz to 1 MHz with CKI at 16.0 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is clocked out on the falling edge of the SK clock. Serial data on the SI pin is clocked in on the rising edge of the SK clock.

MICROWIRE/PLUS Application

Figure 21 illustrates a MICROWIRE/PLUS arrangement for an automotive application. The microcontroller-based system could be used to interface to an instrument cluster and various parts of the automobile. The diagram shows two HPC16164 microcontrollers interconnected to other MICROWIRE peripherals. HPC16164 #1 is set up as the master and initiates all data transfers. HPC16164 #2 is set up as a slave answering to the master.

The master microcontroller interfaces the operator with the system and could also manage the instrument cluster in an automotive application. Information is visually presented to the operator by means of an LCD display controlled by the COP472 display driver. The data to be displayed is sent serially to the COP472 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC16164 could be used as a fuel injection processor and generate timing signals required to operate the fuel valves. The master processor could be used to periodically send updated values to the slave via the MICROWIRE/PLUS link. To speed up the response, chip select logic is implemented by connecting an output from the master to the external interrupt input on the slave.



TL/DD/9682-27

FIGURE 21. MICROWIRE/PLUS Application

HPC16164 UART

The HPC16164 contains a software programmable UART. The UART (see *Figure 22*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

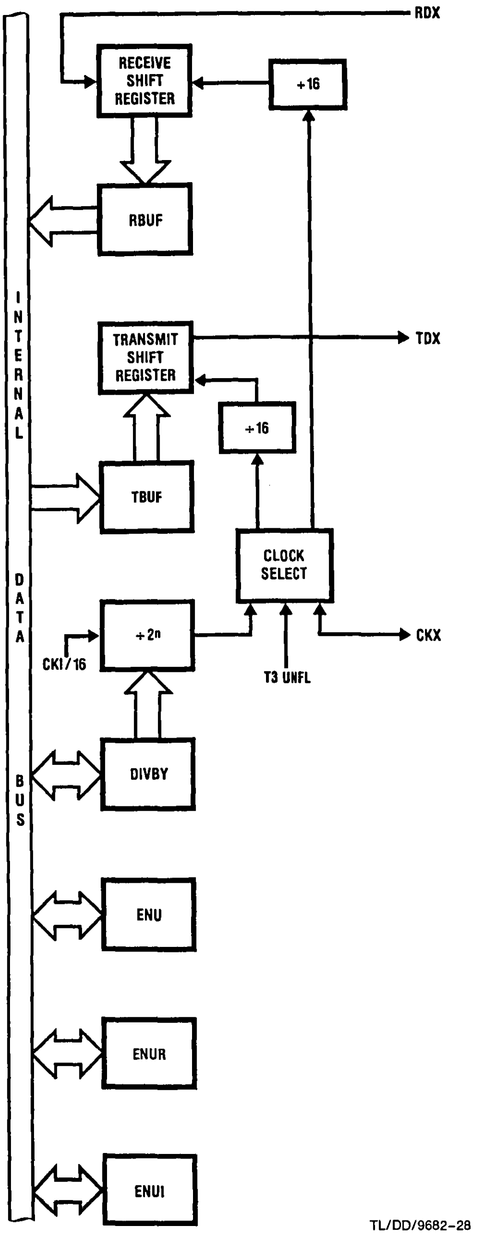
The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 128 kHz in binary steps or T3 underflow. By selecting a 9.83 MHz crystal, all standard baud rates from 75 baud to 38.4 kBaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source.

The HPC16164 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16164 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16164 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.

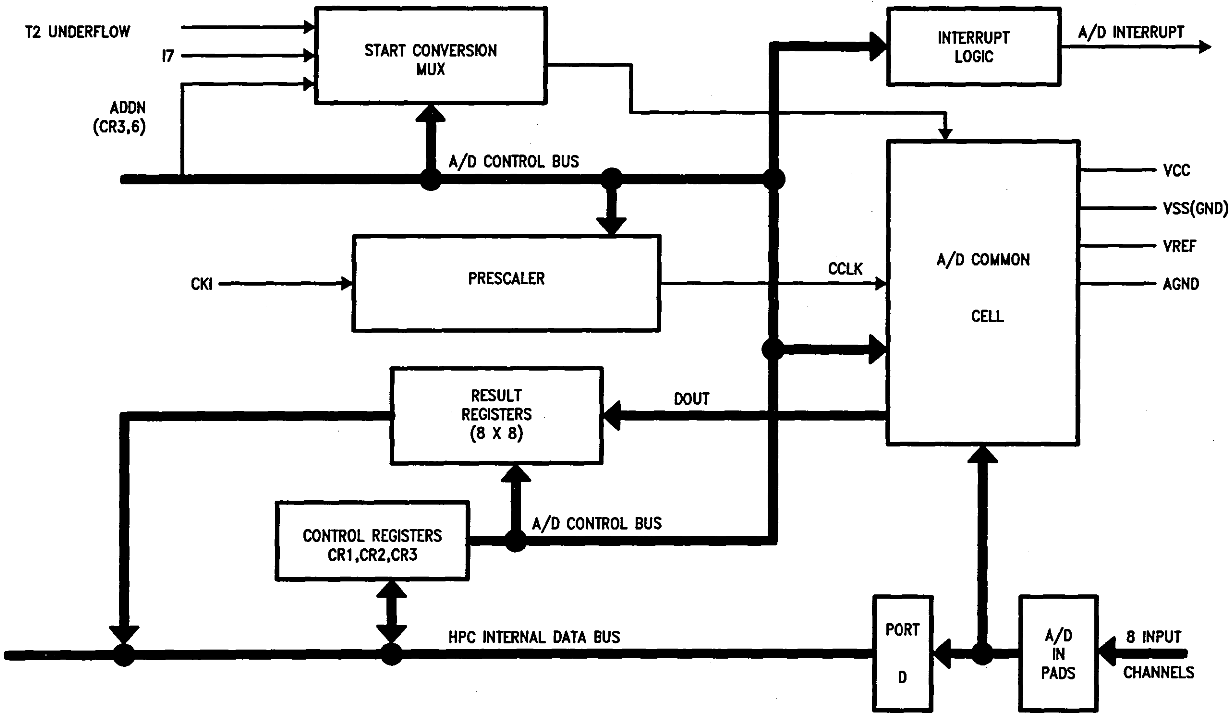
The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16164 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.



TL/DD/9682-28

FIGURE 22. UART Block Diagram

A/D Converter Operation



TL/DD/9682–29

FIGURE 23. A/D Block Diagram

A/D Converter Operation (Continued)

The HPC16164 has an on-board eight-channel 8-bit Analog to Digital converter. Conversion is performed using a successive approximation technique. The A/D converter cell can operate in single-ended mode where the input voltage is applied across one of the eight input channels (D0–D7) and AGND or in differential mode where the input voltage is applied across two adjacent input channels. The A/D converter will convert up to eight channels in single-ended mode and up to four channel-pairs in differential mode.

OPERATING MODES

The operating modes of the converter are selected by 4 bits called ADMODE (CR2.4–7). Associated with the eight input channels in single-ended mode are eight result registers, one for each channel. The A/D converter can be programmed by software to convert on any specific channel storing the result in the result register associated with that channel. It can also be programmed to stop after one conversion or to convert continuously. If a brief history of the signal on any specific input channel is required, the converter can be programmed to convert on that channel and store the consecutive results in each of the result registers before stopping. As a final configuration in single-ended mode, the converter can be programmed to convert the signal on each input channel and store the result in its associated result register continuously.

Associated with each even-odd pair of input channels in differential mode of operation are four result register-pairs. The A/D converter performs two conversions on the selected pair of input channels. One conversion is performed assuming the positive connection is made to the even channel and the negative connection is made to the following odd channel. This result is stored in the result register associated with the even channel. Another conversion is performed assuming the positive connection is made to the odd channel and the negative connection is made to the preceding even channel. This result is stored in the result register associated with the odd channel. This technique does not require that the programmer know the polarity of the input signal. If the even channel result register is non-zero (meaning the odd channel result register is zero), then the input signal is positive with respect to the odd channel. If the odd channel result register is non-zero (meaning the even channel result register is zero), then the input signal is positive with respect to the even channel.

The same operating modes for single-ended operation also apply when the inputs are taken from channel-pairs in differential mode. The programmer can configure the A/D to convert on any selected channel-pair and store the result in its associated result register-pair then stop. The A/D can also be programmed to do this continuously. Conversion can also be done any channel-pair storing the result into four result register-pairs for a history of the differential input. Finally, all input channel-pairs can be converted continuously.

The final mode of operation suppresses the external address/data bus activity during the single conversion modes. These quiet modes of operation utilize the RDY function of the HPC Core to insert wait states in the instruction being executed in order to limit digital noise in the environment due to external bus activity when addressing external memory. The overall effect is to increase the accuracy of the A/D.

CONTROL

The conversion clock supplied to the A/D converter can be selected by three bits in CR1 used as a prescaler on CKI. These bits can be used to ensure that the A/D is clocked as fast as possible when different external crystal frequencies are used. Controlling the starting of conversion cycles in each of the operating modes can be done by four different methods. The method is selected by two bits called SC (CR3.0–1). Conversion cycles can be initiated through software by resetting a bit in a control register, through hardware by an underflow of Timer T2, or externally by a rising or falling edge of a signal input on I7.

INTERRUPTS

The A/D converter can interrupt the HPC when it completes a conversion cycle if one of the non-continuous modes has been selected. If one of the cycle modes was selected, then the converter will request an interrupt after eight conversions. If one of the one-shot modes was selected, then the converter will request an interrupt after every conversion. When this interrupt is generated, the HPC vectors to the on-board peripheral interrupt vector location at address FFF2. The service routine must then determine if the A/D converter requested the interrupt by checking the A/D done flag which doubles as the A/D interrupt pending flag.

REGISTER MAP

The A/D converter status and control registers and the result registers are detailed as follows:

Control Register #1 (CR1)

	Prescaler(3)	Result Reg pntr(4)
msb		lsb

byte at location 0100

Result Register pointer—These four bits are read/only by the software. In all the operating modes that are single channel or single channel-pair, this pointer gets the value of the Channel Select bits (CR2.0–3) and remains constant. In the operating modes that work on multiple channels or multiple channel-pairs, this pointer gets initialized to zero and will change to reflect the current channel that is being converted (default value on power-up is 0000).

Prescaler—These three bits are used to select the clock (CCLK) supplied to the SAR in the A/D converter cell. The maximum clock that can be supplied is 1.67 MHz and the minimum is 100 kHz. Therefore, these bits can be used to ensure that the A/D is clocked as fast as posible at different external crystal frequencies.

000 = stop the clock (CCLK) to the A/D cell (default value on power-up)

011 = use CKI/4 to allow max CKI of 6.66 MHz

010 = use CKI/8 to allow max CKI of 13.33 MHz

111 = use CKI/12 to allow max CKI of 20 MHz

101 = use CKI/16 to allow max CKI of 26.66 MHz

001 = use CKI/20 to allow max CKI of 33.33 MHz

110 = use CKI/24 to allow max CKI of 40 MHz

100 = use CKI/32 to allow max CKI of 53.4 MHz

Note: All remaining unused bits in this control register are UNDEFINED and not available for use by the program.

A/D Converter Operation (Continued)

Control Register #2 (CR2)

ADMODE(4)	Channel Select(4)
msb	lsb

byte at location 0102

ADMODE—These four bits are used to select the mode of operation for the A/D converter as described in OPERATING MODES.

0000 = single-ended, single channel, single result register, one-shot (default value on power-up)

0001 = single-ended, single channel, single result register, continuous

0010 = single-ended, single channel, multiple result registers, stop after 8

0011 = single-ended, multiple channel, multiple result registers, continuous

0100 = differential, single channel-pair, single result register-pair, one-shot

0101 = differential, single channel-pair, single result register-pair, continuous

0110 = differential, single channel-pair, multiple result register-pairs, stop after 4 pairs

0111 = differential, multiple channel-pair, multiple result register-pairs, continuous

Channel Select—These four bits are used to select the channel on which to initiate conversions.

Single-ended

x000 = Convert on Channel 0 (Input Port D.0)

x001 = Convert on Channel 1 (Input Port D.1)

x010 = Convert on Channel 2 (Input Port D.2)

x011 = Convert on Channel 3 (Input Port D.3)

x100 = Convert on Channel 4 (Input Port D.4)

x101 = Convert on Channel 5 (Input Port D.5)

x110 = Convert on Channel 6 (Input Port D.6)

x111 = Convert on Channel 7 (Input Port D.7)

Differential

x000 = Convert on Channel-Pair 0,1

x010 = Convert on Channel-Pair 2,3

x100 = Convert on Channel-Pair 4,5

x110 = Convert on Channel-Pair 6,7

Control Register #3 (CR3)

	ADDN			ADIE	ADEN	SC Mode (2)
msb						lsb

byte at location 0106

SC mode—These two bits are used to select the mode for starting a conversion cycle.

00 = A conversion cycle is initiated by resetting the A/D done flag (ADDN) (default value on power-up).

01 = A conversion cycle is initiated by an underflow of Timer T2.

10 = A conversion cycle is initiated by the falling edge of the signal on input I7.

11 = A conversion cycle is initiated by the rising edge of the signal on input I7.

ADEN—Setting this bit enables pin 4 to be the analog ground, AGND. Resetting this bit returns pin 4 as $\overline{\text{EXUI}}$ (reset on power-up).

ADIE—This is the A/D interrupt enable bit. (reset on power-up).

ADDN—This bit is the A/D done flag and doubles as the A/D interrupt pending flag. If one of the one-shot modes was selected using ADMODE(=xx00) and control was selected as SC = 00, then this bit must be reset by software to initiate the conversion and is set by the hardware at the end of one conversion. If one of the cycle modes was selected using ADMODE(=xx10) and control was selected as SC = 00, then this bit must be reset by software to initiate the conversion cycle and is not set by the hardware until the end of one conversion cycle. If any of the continuous modes were selected and control was selected as SC = 00, then this bit must be reset by software to initiate the conversions and is not set by the hardware until the clock to the A/D cell is stopped by selecting the value 000 for the prescaler. In all other control selections, this bit has no effect on the initiation of conversions but is still necessary for proper interrupt operation. The ADDN flag must also be reset for the quiet modes to work properly (set on power-up).

Note: All remaining unused bits in this control register are UNDEFINED and not available for use by the program. Also, all result register contents are UNDEFINED on power-up.

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC16164 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC16164's and set up systems with very high data exchange rates. Another area of application could be where a HPC16164 is programmed as an intelligent peripheral to a host system such as the Series 32000® microprocessor. *Figure 24* illustrates how a HPC16164 could be used as an intelligent peripherial for a Series 32000-based application.

The interface consists of a Data Bus (port A), a Read Strobe ($\overline{\text{URD}}$), a Write Strobe ($\overline{\text{UWR}}$), a Read Ready Line ($\overline{\text{RDRDY}}$), a Write Ready Line ($\overline{\text{WRRDY}}$) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The $\overline{\text{URD}}$ and $\overline{\text{UWR}}$ inputs may be used to interrupt the HPC16164. The $\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$ outputs may be used to interrupt the host processor.

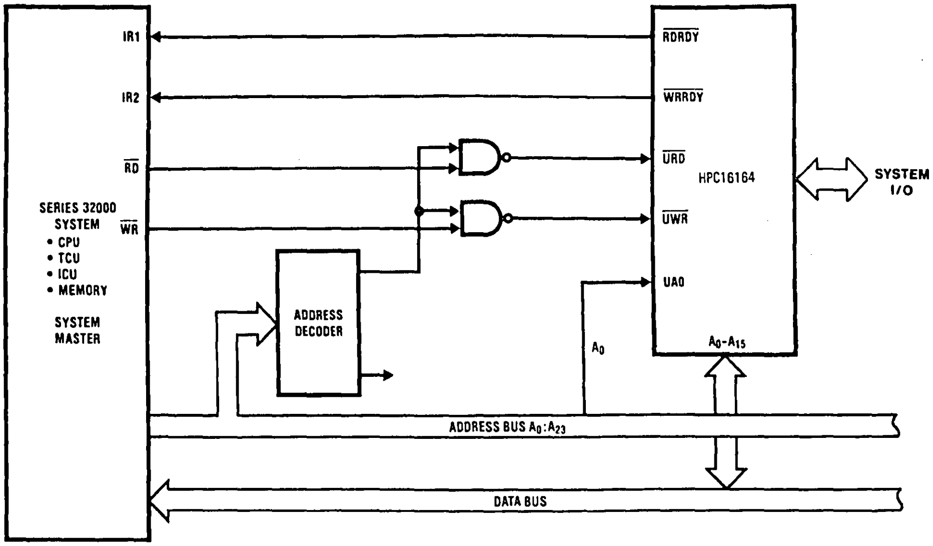
The UPI contains an Input Buffer (IBUF), an Output Buffer (OBUF) and a Control Register (UPIC). In the UPI mode, port A on the HPC16164 is the data bus. UPI can only be used if the HPC16164 is in the Single-Chip mode.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16164 supports shared memory access with two pins. The pins are the RDY/$\overline{\text{HLD}}$ input pin and the $\overline{\text{HLDA}}$ output pin. The user can software select either the Hold or Ready function by the state of a control bit. The $\overline{\text{HLDA}}$ output is multiplexed onto port B.

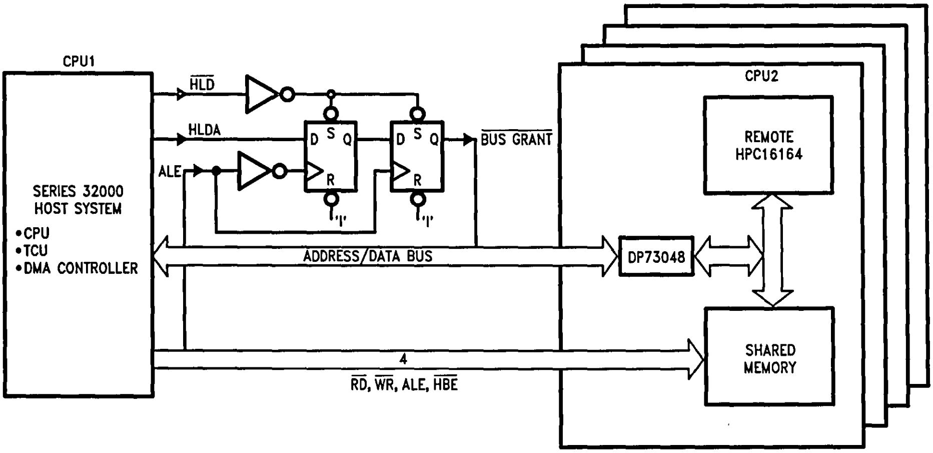
The host uses DMA to interface with the HPC16164. The host initiates a data transfer by activating the $\overline{\text{HLD}}$ input of the HPC16164. In response, the HPC16164 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{\text{HLDA}}$) from the HPC16164 indicating that the sytem bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16164 resumes normal operations.

To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus. *Figure 25* illustrates an application of the shared memory interface between the HPC16164 and a Series 32000 system.



TL/DD/9682-30

FIGURE 24. HPC16164 as a Peripheral: (UPI Interface to Series 32000 Application)



TL/DD/9682-31

FIGURE 25. Shared Memory Application: HPC16164 Interface to Series 32000 System

Memory

The HPC16164 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 16 kbytes of ROM and 512 bytes of RAM available on the chip itself. The ROM may contain program instructions, constants or data. The ROM and RAM share the same address space allowing instructions to be executed out of RAM.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16164 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC16164 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table V.

TABLE V. HPC16164 Memory Map

FFFF:FFF0	Interrupt Vectors	USER MEMORY
FFEF:FFD0	JSRP Vectors	
FFCF:FFCE		
: :	On-Chip ROM*	
E001:C000		
BFFF:BFFE		
: :	External Expansion	
0301:0300	Memory	
02FF:02FE	On-Chip RAM	USER RAM
: :		
01C1:01C0		
0195:0194	Watchdog Address	Watchdog Logic
0192	T0CON Register	Timer Block T0:T3
0191:0190	TMMODE Register	
018F:018E	DIVBY Register	
018D:018C	T3 Timer	
018B:018A	R3 Register	
0189:0188	T2 Timer	
0187:0186	R2 Register	
0185:0184	I2CR Register/ R1	
0183:0182	I3CR Register/ T1	
0181:0180	I4CR Register	
015E:015F	EICR	Timer Block T4:T7
015C	EICON	
0153:0152	Port P Register	
0151:0150	PWMODE Register	
014F:014E	R7 Register	
014D:014C	T7 Timer	
014B:014A	R6 Register	
0149:0148	T6 Timer	
0147:0146	R5 Register	
0145:0144	T5 Timer	
0143:0142	R4 Register	
0141:0140	T4 Timer	
0128	ENUR Register	UART
0126	TBUF Register	
0124	RBUF Register	
0122	ENUI Register	
0120	ENU Register	

011F:011E	A/D Result Register 7	A to D Registers†
011D:011C	A/D Result Register 6	
011B:011A	A/D Result Register 5	
0119:0118	A/D Result Register 4	
0117:0116	A/D Result Register 3	
0115:0114	A/D Result Register 2	
0113:0112	A/D Result Register 1	
0111:0110	A/D Result Register 0	
0106	A/D Control Register #3	
0104	Port D Input Register	
0102	A/D Control Register #2	A to D Registers†
0100	A/D Control Register #1	
00F5:00F4	BFUN Register	PORTS A & B CONTROL
00F3:00F2	DIR B Register	
00F1:00F0	DIR A Register / IBUF	
00E6	UPIC Register	UPI CONTROL
00E3:00E2	Port B	PORTS A & B
00E1:00E0	Port A / OBUF	
00DE	Microcode ROM Dump	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00DD:00DC	HALT Enable Register	
00D8	Port I Input Register	
00D6	SIO Register	
00D4	IRCD Register	
00D2	IRPD Register	
00D0	ENIR Register	
00CF:00CE	X Register	HPC CORE REGISTERS
00CD:00CC	B Register	
00CB:00CA	K Register	
00C9:00C8	A Register	
00C7:00C6	PC Register	
00C5:00C4	SP Register	
00C3:00C2	(reserved)	
00C0	PSW Register	
00BF:00BE	On-Chip	USER RAM
: :	RAM	
0001:0000		

***Note:** The HPC16164 and HPC16064 On-Chip ROM is on addresses C000:FFFF and the External Expansion Memory is 0300:BFFF. The HPC16104 and HPC16004 have no On-Chip ROM, External Memory is 0300:FFFF.

†**Note:** Only one HPC16164 and HPC16104 have on-board A/D.

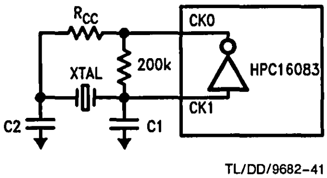
Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused output should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- If you use local regulators, bypass their inputs with a tantalum capacitor of at least 1 µF and bypass their outputs with a 10 µF to 50 µF tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a 10 µF to 20 µF tantalum electrolytic capacitor or a 50 µF to 100 µF aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.



TL/DD/9682-41

A recommended crystal oscillator circuit to be used with the HPC is shown below. See table for recommended component values. The recommended values given in the table below have yielded consistent results and are made to match a crystal with a 20 pF load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within "1" distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout contains a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A 1.0 µF, a 0.1 µF, and a 0.001 µF dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

HPC Oscillator Table

f_C (MHz)	R_{CC} (Ω)	C1 (pF)	C2 (pF)
2	50	82	100
4	50	62	75
6	50	50	56
8	50	47	50
10	50	39	50
12	0	39	39
14	0	33	39
16	0	33	39
18	0	33	33
20	0	33	33
22	0	27	39
24	0	27	39
26	0	27	33
28	0	27	33
30	0	27	27

Crystal Specifications:

"AT" cut, parallel resonant crystals tuned to the desired frequency with the following specifications are recommended:
Series resistance < 65Ω
Loading capacitance: C_L = 20 pF

HPC16164 CPU

The HPC16164 CPU has a 16-bit ALU and six 16-bit registers

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16164 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect Using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action
ARITHMETIC INSTRUCTIONS		
ADD	Add	MA+Meml → MA carry → C
ADC	Add with carry	MA+Meml+C → MA carry → C
ADDS	Add short imm8	A+imm8 → A carry → C
DADC	Decimal add with carry	MA+Meml+C → MA (Decimal) carry → C
SUBC	Subtract with carry	MA−Meml+C → MA carry → C
DSUBC	Decimal subtract w/carry	MA−Meml+C → MA (Decimal) carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C
DIVD	Divide Double Word (unsigned)	X & MA/Meml → MA, rem → X, 0 → K, Carry → C
IFEQ	If equal	Compare MA & Meml, Do next if equal
IFGT	If greater than	Compare MA & Meml, Do next if MA > Meml
AND	Logical and	MA and Meml → MA
OR	Logical or	MA or Meml → MA
XOR	Logical exclusive-or	MA xor Meml → MA
MEMORY MODIFY INSTRUCTIONS		
INC	Increment	Mem + 1 → Mem
DECSZ	Decrement, skip if 0	Mem −1 → Mem, Skip next if Mem = 0

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	Meml → MA
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ←→ Mem
	Exchange, incr/decr X	A ←→ Mem(X), X ±1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP
POP	Pop Stack to Memory	SP−2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ←→ A,B±1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B,imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A − 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ←→ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP),SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP),SP+2 → SP,PC+ # → PC (#is +1025 to −1023)
JSRL	Jump subroutine long	PC → W(SP),SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +257 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP−2 → SP,W(SP) → PC
RETSK	Return then skip next	SP−2 → SP,W(SP) → PC, & skip
RETI	Return from interrupt	SP−2 → SP,W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8 or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
Meml is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data
imm8 is 8-bit immediate data only

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir.		Direct	Indir	Index	Immed.	Direct		Immed.	
	(B)	(X)					*	**	*	**
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADDS	—	—	—	—	—	2	—	—	—	—
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIVD	1	2	3(4)	3	4(5)	—	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC		1
RC		1
IFC		1
IFNC		1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETSK	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16164 has been designed to be extremely code-efficient. The HPC16164 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16164, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16164 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16164 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16164 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16164 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Development Support

DEVELOPMENT SYSTEM

The Microcomputer On Line Emulator (MOLE) is a low cost development system and emulator for all microcontroller products. These include COPS™ microcontrollers and the HPC family of products. The development system consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the development system is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or a modem, or to connect to other development systems in a multi-development system environment.

The development system can be used in either a stand alone mode or in conjunction with a selected host system using PC-DOS communicating via a RS-232 port.

How to Order

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications group. Dial-A-Helper is an Electronic Bulletin Board Information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

Information System Package Contains:
Dial-A-Helper Users Manual
Public Domain Communications Software

Development Support (Continued)

Development Tools Selection Table

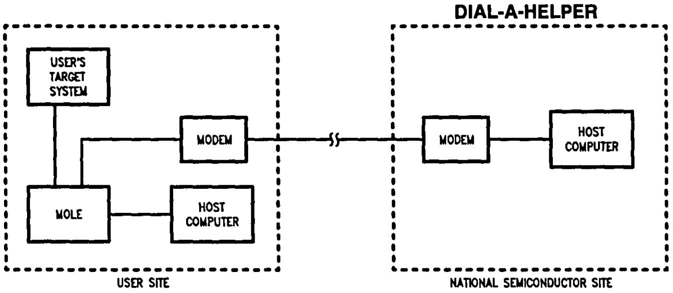
Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBM-R	Relocatable Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001
	MOLE-HPC-VMS	Assembler, Loader, Librarian for VAX/VMS	HPC Software User's Manual and 9 Track Tape	424410836-001
	MOLE-HPC-VMS-C	C Compiler for VAX/VMS	HPC Software User's Manual and 9 Track Tape (Includes Assembler)	424410883-001
	424410897-001	Users Manual		424410897-001

Development Support (Continued)

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances, he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582		
Modem:	(408) 739-1162		
	Baud:	300 or 1200 baud	
	Set-Up:	Length:	8-Bit
		Parity:	None
		Stop Bit:	1
	Operation:	24 Hrs. 7 Days	

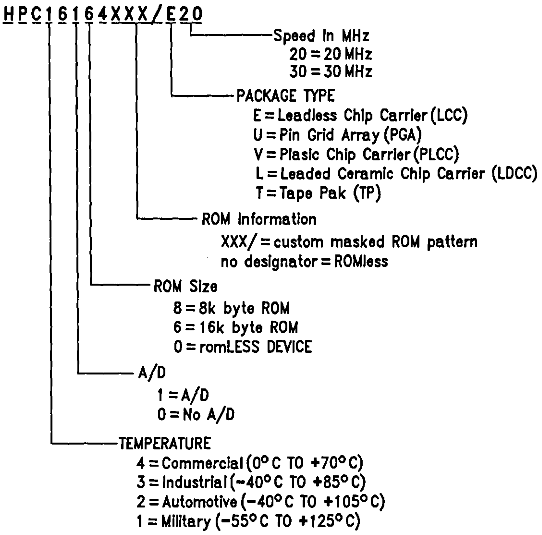


TL/DD/9682-37

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16164 has been generically used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.


HPC16164XXX/E20
Speed In MHz
20 = 20 MHz
30 = 30 MHz
PACKAGE TYPE
E = Leadless Chip Carrier (LCC)
U = Pin Grid Array (PGA)
V = Plasic Chip Carrier (PLCC)
L = Leaded Ceramic Chip Carrier (LDCC)
T = Tape Pak (TP)
ROM Information
XXX/ = custom masked ROM pattern
no designator = ROMless
ROM Size
8 = 8k byte ROM
6 = 16k byte ROM
0 = romLESS DEVICE
A/D
1 = A/D
0 = No A/D
TEMPERATURE
4 = Commercial (0° C TO +70° C)
3 = Industrial (−40° C TO +85° C)
2 = Automotive (−40° C TO +105° C)
1 = Military (−55° C TO +125° C)

TL/DD/9682-33

FIGURE 8. HPC Family Part Numbering Scheme

Examples

HPC46104E20	— ROMless, Commercial temp. (0°C to 70°C), LCC
HPC16164XXX/U20	— 16k masked ROM, Military temp. (−55°C to +125°C), PGA
HPC26104XXX/V20	— ROMless, Automotive temp. (−40°C to +105°C), PLCC

National Semiconductor

PRELIMINARY

HPC16400/HPC36400/HPC46400 High-Performance Communications microController

General Description

The HPC16400 is a member of the HPC™ family of High Performance microControllers. Each member of the family has the same identical core CPU with a unique memory and I/O configuration to suit specific applications. Each part is fabricated in National's advanced microCMOS technology. This process combined with an advanced architecture provides fast, flexible I/O control, efficient data manipulation, and high speed computation.

The HPC16400 has 4 functional blocks to support a wide range of communication application-2 HDLC channels, 4 channel DMA controller to facilitate data flow for the HDLC channels, programmable serial interface and UART.

The serial interface decoder allows the 2 HDLC channels to be used with devices using interchip serial link for point-to-point & multipoint data exchanges. The decoder generates enable signals for the HDLC channels allowing multiplexed D and B channel data to be accessed.

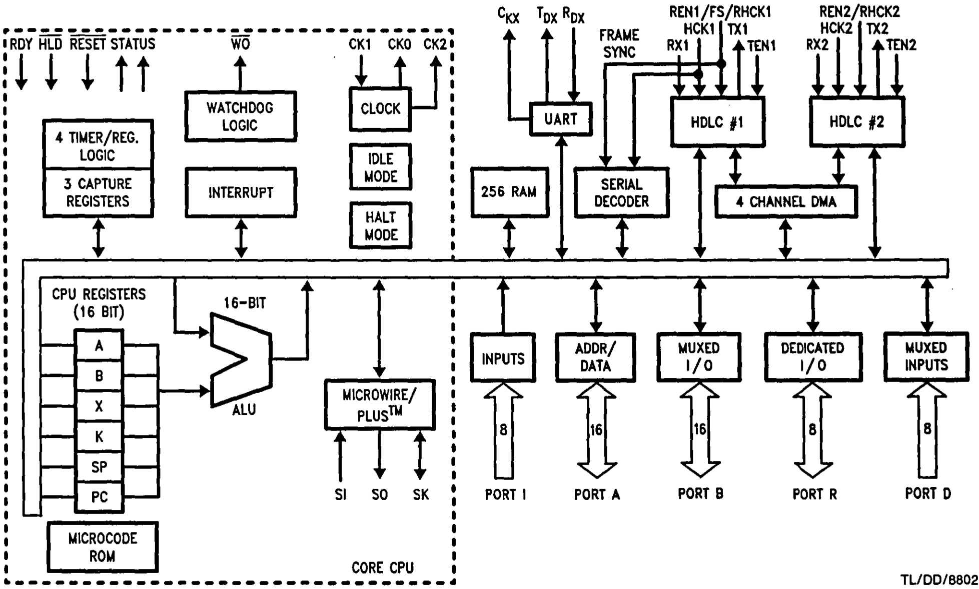
The HDLC channels manage the link by providing sequencing using the HDLC framing along with error control based upon a cyclic redundancy check (CRC). Multiple address recognition modes, and both bit and byte modes of operation are supported.

The HPC16400 is available in 68-pin PLCC, LCC, LDCC and 84-pin TapePak® packages.

Features

- HPC family—core features:
  - — 16-bit data bus, ALU, and registers
  - — 64 kbytes of external direct memory addressing
  - — FAST!—20.0 MHz system clock
  - — High code efficiency
  - — 16 x 16 multiply and 32 x 16 divide
  - — Eight vectored interrupt sources
  - — Four 16-bit timer/counters with WATCHDOG logic
  - — MICROWIRE/PLUS serial I/O interface
  - — CMOS—low power with two power save modes
- Two full duplex HDLC channels
  - — Optimized for X.25 and LAPD applications
  - — Programmable frame address recognition
  - — Up to 4.65 Mbps serial data rate
  - — Built in diagnostics
  - — Synchronous bypass mode
- Programmable interchip serial data decoder
- Four channel DMA controller
- UART—full duplex, programmable baud rate (up to 208.3 kBaud)
- 544 kbytes of extended addressing
- Easy interface to National's DASL, 'U' and 'S' transceivers—TP3400, TP3410 and TP3420
- Commercial (0°C to 70°C) Industrial (−40°C to +85°C) and military (−55°C to +125°C) temperature ranges

Block Diagram



TL/DD/8802-1

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Total Allowable Source or Sink Current	100 mA
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)	300°C
ESD Rating	2000V
V_{CC} with Respect to GND	−0.5V to 7.0V
All Other Pins	(V_{CC} + 0.5)V to (GND − 0.5)V

Note: *Absolute maximum ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications are not ensured when operating the device at absolute maximum ratings.*

DC Electrical Characteristics

V_{CC} = 5.0V ±10% unless otherwise specified, T_A = 0°C to +70°C for HPC46400, −40°C to +85°C for HPC36400, −55°C to +125°C for HPC16400

Symbol	Parameter	Test Conditions	Min	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz (Note 1)		70	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		10	mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.5V, f_{in} = 20.0 MHz (Note 1)		10	mA
		V_{CC} = 5.5V, f_{in} = 2.0 MHz (Note 1)		2	mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.5V, f_{in} = 0 kHz (Note 1)		500	μA
		V_{CC} = 2.5V, f_{in} = 0 kHz (Note 1)		150	μA
INPUT VOLTAGE LEVELS RESET, NMI, CKI AND WO (SCHMITT TRIGGERED)					
V_{IH_1}	Logic High		0.9 V_{CC}		V
V_{IL_1}	Logic Low			0.1 V_{CC}	V
PORT A					
V_{IH_2}	Logic High		2.0		V
V_{IL_2}	Logic Low			0.8	V
ALL OTHER INPUTS					
V_{IH_3}	Logic High		0.7 V_{CC}		V
V_{IL_3}	Logic Low			0.2 V_{CC}	V
I_{LI}	Input Leakage Current			±1	μA
C_I	Input Capacitance	(Note 2)		10	pF
C_{IO}	I/O Capacitance	(Note 2)		20	pF
OUTPUT VOLTAGE LEVELS					
V_{OH_1}	Logic High (CMOS)	I_{OH} = −10 μA (Note 2)	V_{CC} − 0.1		V
V_{OL_1}	Logic Low (CMOS)	I_{OH} = 10 μA (Note 2)		0.1	V
V_{OH_2}	Port A/B Drive, CK2 (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA	2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4	V
V_{OH_3}	Other Port Pin Drive, WO (B_0–B_9, B_{13}, B_{14}, R_0–R_7, D_5, D_7)	I_{OH} = −1.6 mA	2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4	V
V_{OH_4}	ST1 and ST2 Drive	I_{OH} = −6 mA	2.4		V
V_{OL_4}		I_{OL} = 1.6 mA		0.4	V
V_{RAM}	RAM Keep-Alive Voltage	(Note 3)	2.5		V
I_{OZ}	TRI-STATE Leakage Current			±5	μA

Note 1: I_{CC_1}, I_{CC_2}, I_{CC_3} measured with no external drive (I_{OH} and I_{OL} = 0, I_{IH} and I_{IL} = 0). I_{CC_1} is measured with $\overline{RESET}$ = V_{SS}. I_{CC_3} is measured with NMI = V_{CC}. CKI driven to V_{IH_1} and V_{IL_1} with rise and fall times less than 10 ns.

Note 2: These parameters are guaranteed by design and are not tested.

Note 3: Test duration is 100 ms.

AC Electrical Characteristics V_{CC} = 5.0V ±10%, T_A = 0°C to +70°C for HPC46400, −40°C to +85°C for HPC36400, −55°C to +125°C for HPC16400

Symbol	Parameter	Min	Max	Units
f_C = CKI freq.	Operating Frequency	2.0	20	MHz
$t_{C1} = 1/f_C$	Clock Period	50	500	ns
t_{CKIR} (Note 3)	CKI Rise Time		7	ns
t_{CKIF} (Note 3)	CKI Fall Time		7	ns
$[t_{CKIH}/(t_{CKIH} + t_{CKIL})]100$	Duty Cycle	45	55	%
$t_C = 2/f_C$	Timing Cycle	100		ns
$t_{LL} = \frac{1}{2} t_C - 9$	ALE Pulse Width	41		ns
t_{DC1C2R} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Rising Edge	0	55	ns
t_{DC1C2F} (Notes 1, 2)	Delay from CKI Falling Edge to CK2 Falling Edge	0	55	ns
$t_{DC1ALER}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Rising Edge for CPU Cycles and CKI Falling Edge for DMA Cycles	0	35	ns
$t_{DC1ALEF}$ (Notes 1, 2)	Delay from CKI Rising Edge to ALE Falling Edge for CPU Cycles and CKI Falling Edge for DMA Cycles	0	35	ns
$t_{DC2ALER} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CKI Rising Edge to ALE Rising Edge		45	ns
$t_{DC2ALEF} = \frac{1}{4} t_C + 20$ (Note 2)	Delay from CK2 Falling Edge to ALE Falling Edge		45	ns
$t_{ST} = \frac{1}{4} t_C - 16$	Address Valid to ALE Falling Edge	9		ns
$t_{WAIT} = t_C$	Wait State Period	100		ns
$f_{XIN} = f_C/19$	External Timer Input Frequency		1052	kHz
$t_{XIN} = t_C$	Pulse Width for Timer Inputs	100		ns
$f_{XOUT} = f_C/16$	Timer Output Frequency		1.25	MHz
$f_{MW} = f_C/19$	External MICROWIRE/PLUS Clock Input Frequency		1.25	MHz
$f_U = f_C/8$	External UART Clock Input Frequency		2.5	MHz

Note 1: Do not design with this parameter unless CKI is driven with an active signal. When using a passive crystal circuit, CKI or CKO **should not** be connected to any external logic since any load (besides the passive components in the crystal circuit) will affect the stability of the crystal unpredictably.

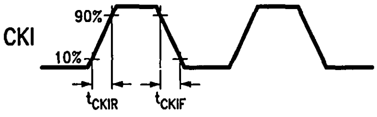
Note 2: These are not directly tested parameters. Therefore the given min/max value cannot be guaranteed. It is, however, derived from measured parameters, and may be used for system design with a very high confidence level.

Note 3: These parameters are guaranteed by design and are not tested.

Note: Measurement of AC specifications is done with external clock drive on CKI with 50% duty cycle. The capacitive load on CKO must be kept below 15 pF else AC measurements will be skewed.

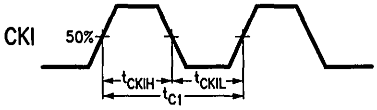
CKI Input Signal Characteristics

Rise/Fall Time



TL/DD/8802-25

Duty Cycle



TL/DD/8802-26

CPU Read Cycle Timing (See *Figure 1*)

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{2} t_C - 20$	ALE Falling Edge to $\overline{\text{RD}}$ Falling Edge	30		ns
$t_{RW} = \frac{1}{4} t_C + WS - 10$	$\overline{\text{RD}}$ Pulse Width	115		ns
$t_{DR} = t_C - 15$	Data Hold after Rising Edge of $\overline{\text{RD}}$	0	85	ns
$t_{ACC} = t_C + WS - 55$ (Note 2)	Address Valid to Input Data Valid		145	ns
$t_{RD} = \frac{1}{4} t_C + WS - 35$	$\overline{\text{RD}}$ Falling Edge to Input Data Valid		90	ns
$t_{RDA} = t_C - 5$	$\overline{\text{RD}}$ Rising Edge to Address Valid	95		ns
$t_{VP} = \frac{1}{4} t_C - 10$	Address Hold from ALE Falling Edge	15		ns

Note: WS = t_{WAIT} * number of pre-programmed wait states. Minimum and Maximum values are calculated from maximum operating frequency with one (1) wait state pre-programmed.

CPU Write Cycle Timing (See *Figure 2*)

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 20$	ALE Falling Edge to $\overline{\text{WR}}$ Falling Edge	30		ns
$t_{WW} = \frac{3}{4} t_C + WS - 15$	$\overline{\text{WR}}$ Pulse Width	160		ns
$t_{HW} = \frac{1}{4} t_C - 5$	Data Hold after Rising Edge of $\overline{\text{WR}}$	20		ns
$t_V = \frac{1}{2} t_C + WS - 40$	Data Valid before Rising Edge of $\overline{\text{WR}}$	110		ns
$t_{VP} = \frac{1}{4} t_C - 10$	Address Hold from ALE Falling Edge	15		ns

Note: Bus output (Port A) C_L = 100 pF, CK2 output C_L = 50 pF, other outputs C_L = 80 pF. AC Parameters are tested using DC Characteristics and non CMOS outputs.

DMA Read Cycle Timing f_C = 20 MHz (See *Figure 1*)

Symbol	Parameter	Min	Max	Units
$t_{ARR} = \frac{1}{2} t_C - 20$	ALE Falling Edge to $\overline{\text{RD}}$ Falling Edge	30		ns
$t_{RW} = \frac{3}{2} t_C - 15$	$\overline{\text{RD}}$ Pulse Width	135		ns
$t_{DR} = \frac{3}{4} t_C - 15$	Data Hold After Rising Edge of $\overline{\text{RD}}$	0	60	ns
$t_{ACC} = \frac{9}{4} t_C - 75$	Address Valid to Input Data Valid		150	ns
$t_{RD} = \frac{3}{2} t_C - 35$	$\overline{\text{RD}}$ Falling Edge to Input Data Valid		115	ns
$t_{RDA} = \frac{3}{4} t_C - 5$	$\overline{\text{RD}}$ Rising Edge to Address Valid	95		ns
$t_{VP} = \frac{1}{2} t_C - 10$	Address Hold from ALE Falling Edge	40		ns

Note: Minimum and Maximum values are calculated for maximum operating frequency.

DMA Write Cycle Timing f_C = 20 MHz (See *Figure 2*)

Symbol	Parameter	Min	Max	Units
$t_{ARW} = \frac{1}{2} t_C - 20$	ALE Falling Edge to $\overline{\text{WR}}$ Falling Edge	30		ns
$t_{WW} = \frac{3}{2} t_C - 15$	$\overline{\text{WR}}$ Pulse Width	135		ns
$t_{HW} = \frac{1}{2} t_C - 15$	Data Hold After Rising Edge of $\overline{\text{WR}}$	35		ns
$t_V = \frac{3}{2} t_C - 50$	Data Valid before Rising Edge of $\overline{\text{WR}}$	100		ns
$t_{VP} = \frac{1}{2} t_C - 10$	Address Hold from ALE Falling Edge	40		ns

Note: Bus output (Port A) C_L = 100 pF, CK2 output C_L = 50 pF, other outputs C_L = 80 pF. AC Parameters are tested using DC Characteristics and non CMOS outputs.

Ready/Hold Timing

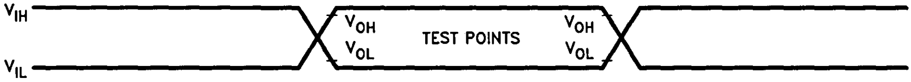
Symbol	Parameter	Min	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of $\overline{RDY}$		75	ns
$t_{RWP} = t_C$	$\overline{RDY}$ Pulse Width	100		ns
$t_{SALE} = \frac{3}{4} t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE	115		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width	110		ns
$t_{HAD} = \frac{3}{4} t_C + 85$	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		160	ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		200*	ns
$t_{BF} = \frac{1}{2} t_C + 66$	Bus Float after Falling Edge of $\overline{HLDA}$		116 (Due to Emulation)	ns
$t_{BE} = \frac{1}{2} t_C + 66$	Bus Enable before Rising Edge of $\overline{HLDA}$	116		ns

***Note:** t_{HAE} may be as long as ($3t_C + 4ws + 72t_C + 90$) depending on which instruction is being executed, the addressing mode and number of wait states. t_{HAE} maximum value tested is for the optimal case.

MICROWIRE/PLUS Timing

Symbol	Parameter	Min	Max	Units
t_{UWS}	MICROWIRE Setup Time			
Master		100		ns
Slave		20		
t_{UWH}	MICROWIRE Hold Time			
Master		20		ns
Slave		50		
t_{UWV}	MICROWIRE Output Valid Time			
Master			50	ns
Slave			150	

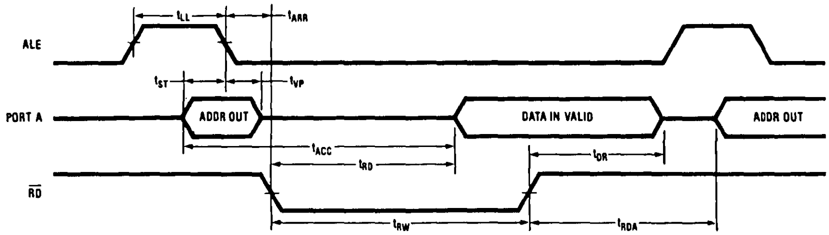
Input and Output for AC Tests



TL/DD/8802-34

Note: AC testing inputs are driven at V_{IH} for a logic "1" and V_{IL} for a logic "0". Output timing measurements are made at V_{OH} for a logic "1" and V_{OL} for a logic "0".

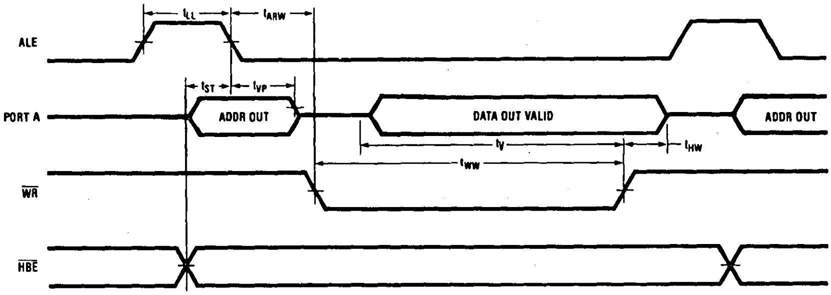
Timing Waveforms



TL/DD/8802-22

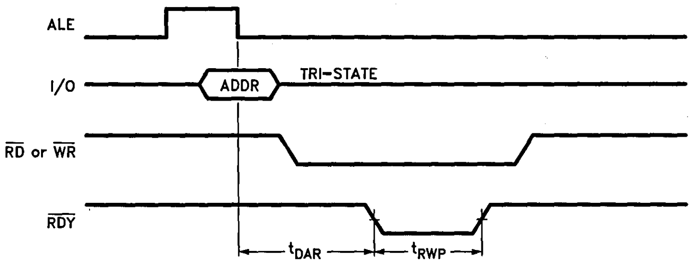
FIGURE 1. CPU and DMA Read Cycles

Timing Waveforms (Continued)



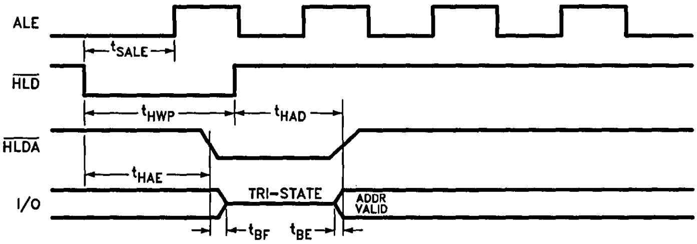
TL/DD/8802-21

FIGURE 2. CPU and DMA Write Cycles



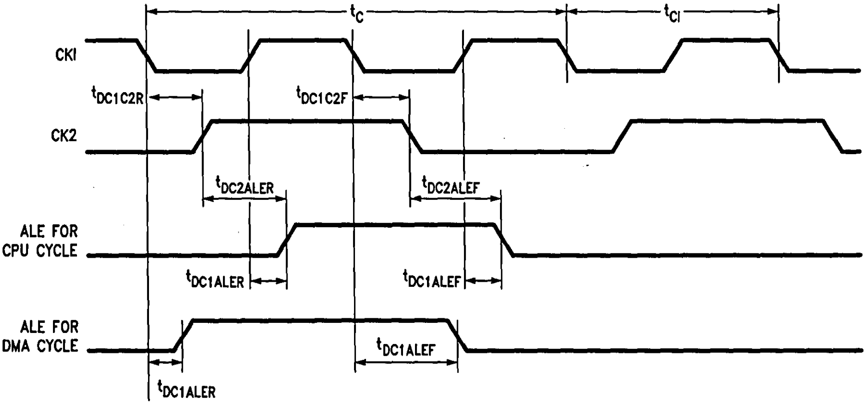
TL/DD/8802-4

FIGURE 3. Ready Mode Timing



TL/DD/8802-5

FIGURE 4. Hold Mode Timing

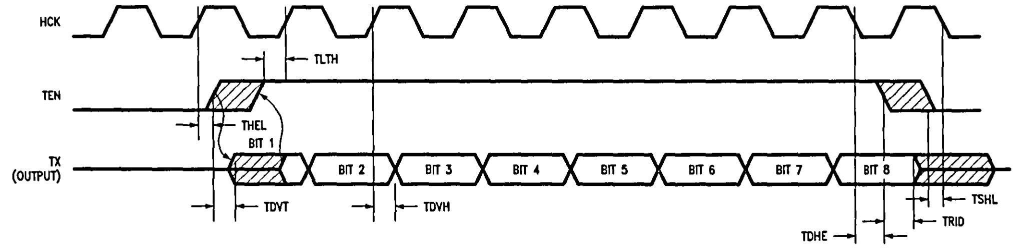


TL/DD/8802-23

FIGURE 5. CKI, CK2 ALE Timing Diagram

Timing Waveforms (Continued)

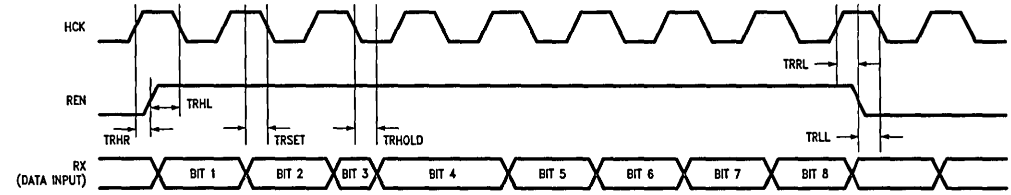
Timing Diagrams for TX using External Enable



TL/DD/8802-28

Symbol	Parameter	Min	Max	Units
THEL	Hold Time of Enable Low to Rising Edge of HCK	5		ns
TDVT	Data Valid Time from Rising Edge of TEN		40	ns
TDVH	Data Valid Time from Rising Edge of HCK		45	ns
TDHE	Hold Time of TEN Falling Edge to HCK Falling Edge	60		ns
TRID	Time From TEN Falling Edge to TRI-STATE® of TX Output		40	ns
TSHL	Setup Time of TEN Falling Edge to HCK Falling Edge	20		ns
TLTH	Setup Time Required for TEN Rising Edge of HCK Rising Edge	80		ns

Timing Diagrams for RX using External Enable



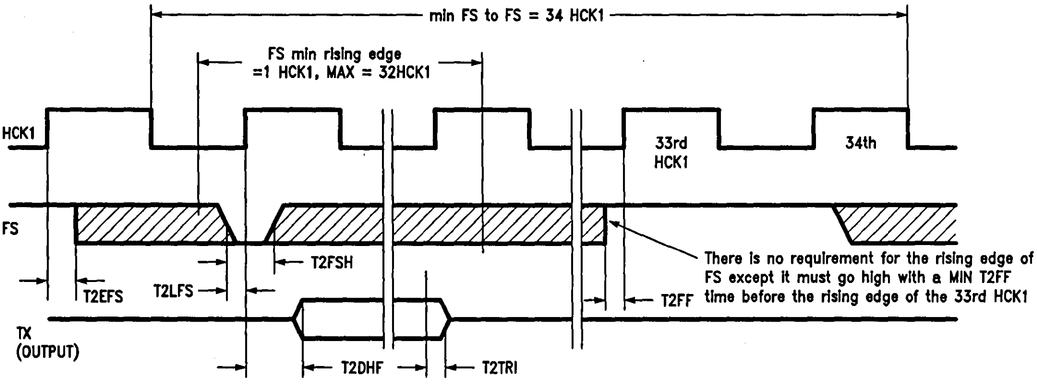
TL/DD/8802-29

Symbol	Parameter	Min	Max	Comments	Units
TRHR	Hold Time of Enable Low to Rising Edge of HCK	5		50% to 50%	ns
TRHL	Setup Time of Rising Edge of REN to Falling Edge of HCK	30		50% to 50%	ns
TRSET	Data Setup Time of RX Data to Falling Edge of HCK	20		50% to 50%	ns
TRHOLD	Data Hold Time of RX Data to Falling Edge of HCK	20		50% to 50%	ns
TRRL	Hold Time of REN High to Rising Edge of HCK	5		50% to 50%	ns
TRLL	Setup Time of REN Low of Falling Edge of HCK	30		50% to 50%	ns

Note: When using RX with normal serial decoder the timings for TRSET and TRHOLD would apply and will have the same values as above.

Timing Waveforms (Continued)

Serial Decoder Timing Diagrams (Mode 2)

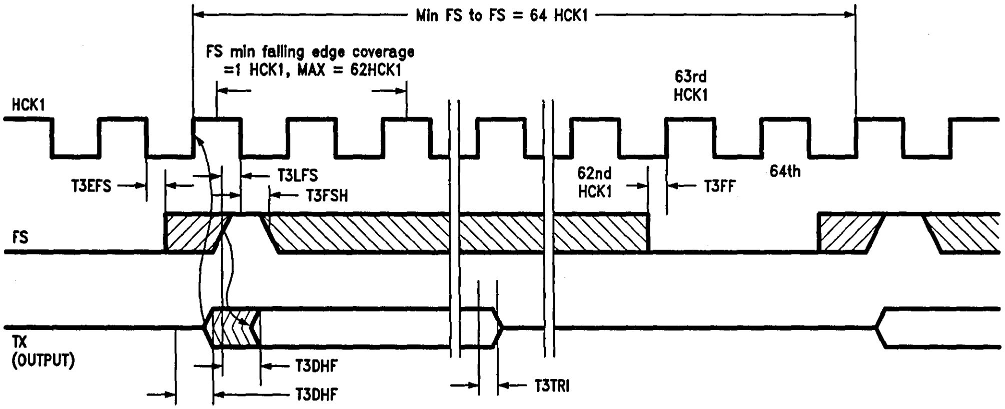

min FS to FS = 34 HCK1
FS min rising edge
=1 HCK1, MAX = 32HCK1
HCK1
33rd
HCK1
34th
FS
T2FSH
There is no requirement for the rising edge of
FS except it must go high with a MIN T2FF
time before the rising edge of the 33rd HCK1
T2EFS
T2LFS
T2FF
TX
(OUTPUT)
T2DHF
T2TRI

TL/DD/8802-30

Symbol	Parameter	Min	Max	Comments	Units
T2EFS	Delay Time of FS High to Rising Edge of HCK1	10		50% to 50%	ns
T2LFS	Setup Time of FS Low to Rising Edge of HCK1	20		50% to 50%	ns
T2FF	Setup Time of FS High to Rising Edge of the 33rd HCK1	20		50% to 50%	ns
T2FSH	Hold Time of FS Low to Rising Edge of HCK1	20		50% to 50%	ns
T2DHF	Delay From Rising Edge of HCK1 to TX Data Out Valid (First Bit after Valid FS)		50	50% to 0.1 or 0.9 V_{CC}	ns
T2TRI	Delay From Rising Edge of HCK to TRI-STATE of TX Output		40	50% to 0.1 V_{CC} (T0H) 50% to 0.9 V_{CC} (T0H)	ns

Note: Receiver operation is guaranteed when min. TRSET and TRHOLD specs are met.

Serial Decoder Timing Diagrams (Mode 3)



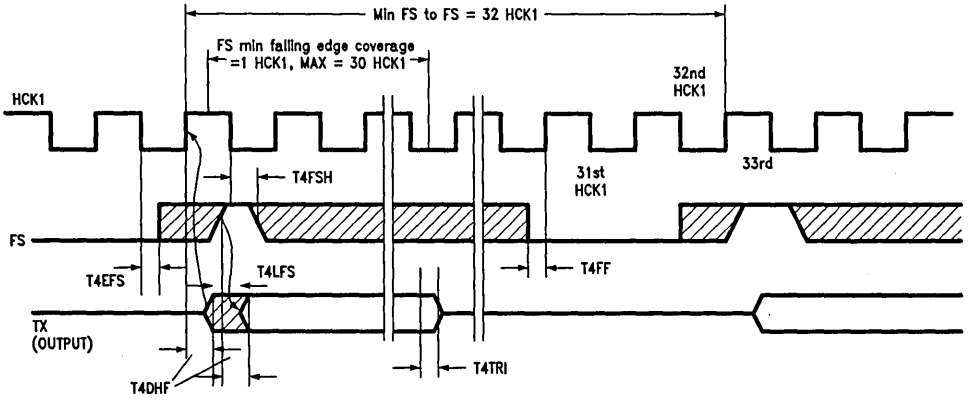
TL/DD/8802-31

Symbol	Parameter	Min	Max	Comments	Units
T3EFS	Delay of Falling Edge of HCK1 to Rising Edge of FS (Early Frame Sync.)	10		50% to 50%	ns
T3LFS	Min Setup Time of FS High to HCK1 Falling Edge to Guarantee RX Operation (Late Frame Sync.)	45		50% to 50%	ns
T3DHF	Delay Time From Rising HCK1 or FS to TX Output Data Valid		50	50% to 0.1 or 0.9 V_{CC}	ns
T3FSH	Hold Time of FS High to Falling Edge of HCK1 (Frame Sync. Hold)	20		50% to 50%	ns
T3FF	Setup Time of FS Low to Rising Edge of 63rd HCK1	20		50% to 50%	ns
T3TRI	Delay From Rising Edge of HCK1 to TRI-STATE of TX Output		40	50% to 0.1 V_{CC} (T0H) 50% to 0.9 V_{CC} (T1H)	ns

Note: Receiver operation is guaranteed when TRSET and TRHOLD specs are met.

Timing Waveforms (Continued)

Serial Decoder Timing Diagrams (Mode 4)

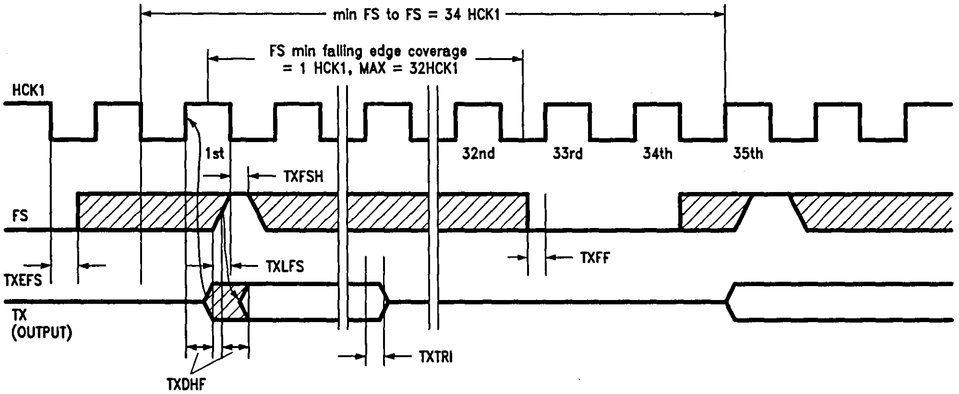


TL/DD/8802-32

Symbol	Parameter	Min	Max	Comments	Units
T4EFS	Delay of Falling Edge of HCK1 to Rising Edge of FS	10		50% to 50%	ns
T4LFS	Min Setup Time FS High to HCK1 Falling Edge to Guarantee RX Operation	45		50% to 50%	ns
T4DHF	Delay Time From Active Edge to TX Output Data Valid		50	50% to 0.1 or 0.9 V_{CC}	ns
T4FSH	Hold Time of FS High to Falling Edge of HCK1	20		50% to 50%	ns
T4FF	Setup Time of FS Low to Rising Edge of 31st HCK1	20		50% to 50%	ns
T4TRI	Delay From Rising Edge of HCK to TRI-STATE of TX Output		40	50% to 0.1 V_{CC} (T0H) 50% to 0.9 V_{CC} (T1H)	ns

Note: Receiver operation is guaranteed when TRSET & TRHOLD specs are met.

Serial Decoder Timing Diagrams (Mode 5,6,7)



X = 5, 6, 7

TL/DD/8802-33

Symbol	Parameter	Min	Max	Comments	Units
TXEFS	Delay of Falling Edge of HCK1 to Rising Edge of FS	10		50% to 50%	ns
TXLFS	Min Setup Time of FS High to HCK1 Falling Edge to Guarantee RX Operation	45		50% to 50%	ns
TXDHF	Delay Time From Active Edge to TX Output Data Valid		50	50% to 0.1 or 0.9 V_{CC}	ns
TXFSH	Hold Time of FS High to Falling Edge of HCK1	20		50% to 50%	ns
TXFF	Setup Time of FS Low to Rising Edge of 33rd HCK1		20	50% to 50%	ns
TXTRI	Delay From Rising Edge of HCK to TRI-STATE of TX Output		40	50% to 0.1 V_{CC} (T0H) 50% to 0.9 V_{CC} (T1H)	ns

Note: Receiver operation is guaranteed when TRSET and TRHOLD specs are met.

Pin Descriptions

I/O PORTS

Port A is a 16-bit multiplexed address/data bus used for accessing external program and data memory. Four associated bus control signals are available on port B. The Address Latch Enable (ALE) signal is used to provide timing to demultiplex the bus. Reading from and writing to external memory are signalled by $\overline{\text{RD}}$ and $\overline{\text{WR}}$ respectively. External memory can be addressed as either bytes or words with the decoding controlled by two lines, Bus High Byte enable ($\overline{\text{HBE}}$) and Address/Data Line 0 (A0).

Port B is a 16-bit port, with 12 bits of bidirectional I/O. Pins B10, B11, B12 and B15 are the control bus signals for the address/data bus. Port B may also be configured via a function register BFUN to individually allow each bidirectional I/O pin to have an alternate function.

B0:	TDX	UART Data Output
B1:	CFLG1	Closing Flag Output for HDLC #1 Transmitter
B2:	CKX	UART Clock (Input or Output)
B3:	T2IO	Timer2 I/O Pin
B4:	T3IO	Timer3 I/O Pin
B5:	SO	MICROWIRE/PLUS Output
B6:	SK	MICROWIRE/PLUS Clock (Input or Output)
B7:	$\overline{\text{HLDA}}$	Hold Acknowledge Output
B8:	TS0	Timer Synchronous Output
B9:	TS1	Timer Synchronous Output
B10:	ALE	Address Latch Enable Output for Address/Data Bus
B11:	$\overline{\text{WR}}$	Address/Data Bus Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output for Address/Data Bus
B13:	TS2	Timer Synchronous Output
B14:	TS3	Timer Synchronous Output
B15:	$\overline{\text{RD}}$	Address/Data Bus Read Output

When operating in the extended memory addressing mode, four bits of port B can are used as follows—

B8:	BS0	Memory bank switch output 0 (LSB)
B9:	BS1	Memory bank switch output 1
B13:	BS2	Memory bank switch output 2
B14:	BS3	Memory bank switch output 3 (MSB)

Port I is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

I0:	HCK2	HLDC #2 Clock Input
I1:	NMI	Nonmaskable Interrupt Input
I2:	INT2	Maskable Interrupt/Input Capture
I3:	INT3	Maskable Interrupt/Input Capture
I4:	INT4/RDY	Maskable Interrupt/Input Capture/ Ready Input
I5:	SI	MICROWIRE/PLUS Data Input
I6:	RDX	UART Data Input
I7:	HCK1	HDLC #1 Clock/Serial Decoder Clock Input

Port D is an 8-bit input port that can be read as general purpose inputs and can also be used for the following functions:

D0:	REN1/FS/ RHCK1	Receiver #1 Enable/Serial Decoder Frame Sync Input/Receiver #1 Clock Input
D1:	TEN1	Transmitter #1 Enable Input
D2:	REN2/ RHCK2	Receiver #2 Enable Input/Receiver #2 Clock Input
D3:	TEN2	Transmitter #2 Enable Input
D4:	RX1	Receiver #1 Data Input
D5:	TX1	Transmitter #1 Data Output
D6:	RX2	Receiver #2 Data Input
D7:	TX2	Transmitter #2 Data Output

Note: Any of these pins can be read by software. Therefore, unused functions can be used as general purpose inputs, notably external enable lines when the internal serial decoder is used (see SERIAL DECODER/ENABLE CONFIGURATION REGISTER).

Port R is an 8-bit bidirectional I/O port available for general purpose I/O operations. Port R has a direction register to enable each separate pin to be individually defined as an input or output. It has a data register which contains the value to be output. In addition, the Port R pins can be read directly using the Port R pins address.

Pin Descriptions (Continued)

POWER SUPPLIES

$V_{CC1} + V_{CC2}$ Positive Power Supply (two pins)

GND Ground for On-Chip Logic

DGND Ground for Output Buffers

Note: There are two electrically connected V_{CC} pins on the chip, GND and DGND are electrically isolated. Both V_{CC} pins and both ground pins must be used.

CLOCK PINS

CKI The System Clock Input

CKO The System Clock Output (Inversion of CKI)

Pins CKI and CKO are usually connected across an external crystal.

CK2 Clock Output (CKI divided by 2)

OTHER PINS

$\overline{\text{WO}}$ This is an active low open drain output which signals an illegal situation has been detected by the Watch Dog logic.

ST1 Bus Cycle Status Output indicates first opcode fetch.

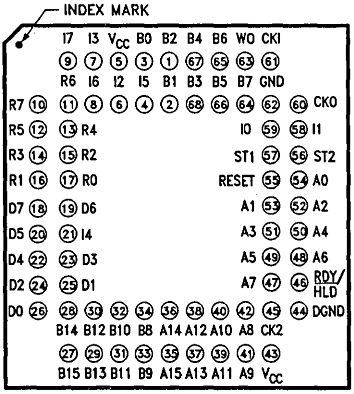
ST2 Bus Cycle Status Output indicates machine states (skip and interrupt).

$\overline{\text{RESET}}$ Active low input that forces the chip to restart and sets the ports in a TRI-STATE mode.

RDY/$\overline{\text{HLD}}$ Has two uses, selected by a software bit. This pin is either a READY input to extend the bus cycle for slower memories or a HOLD-REQUEST input to put the bus in a high impedance state for external DMA purposes. In the second case the I4 pin becomes the READY input.

Connection Diagram

Pin Grid Array

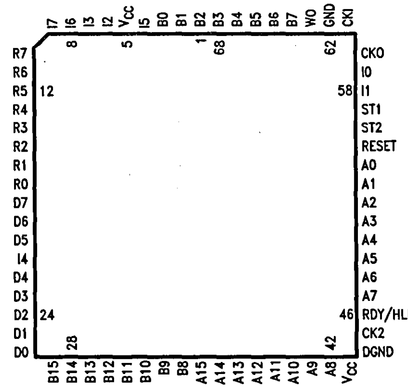

INDEX MARK
I7 I3 VCC B0 B2 B4 B6 WO CKI
9 7 5 3 1 67 65 63 61
R6 I6 I2 I5 B1 B3 B5 B7 GND
R7 10 11 8 6 4 2 68 66 64 62 60 CKO
R5 12 13 R4 I0 59 58 I1
R3 14 15 R2 ST1 57 56 ST2
R1 16 17 R0 RESET 55 54 A0
D7 18 19 D6 A1 53 52 A2
D5 20 21 I4 A3 51 50 A4
D4 22 23 D3 A5 49 48 A6
D2 24 25 D1 A7 47 46 RDY/HLD
D0 26 28 30 32 34 36 38 40 42 45 44 DGND
B14 B12 B10 B8 A14 A12 A10 A8 CK2
27 29 31 33 35 37 39 41 43
B15 B13 B11 B9 A15 A13 A11 A9 VCC

TL/DD/8802-24

Top View

Order Number HPC16400U
See NS Package Number U68A

Plastic, Leadless and Leaded Chip Carriers


I7 I6 I3 I2 VCC I5 B0 B1 B2 B3 B4 B5 B6 B7 WO GND CKI
8 5 1 68 62
R7 R6 R5 R4 R3 R2 R1 R0 D7 D6 D5 I4 D4 D3 D2 D1 D0
12 24
58 46
CKO I0 I1 ST1 ST2 RESET A0 A1 A2 A3 A4 A5 A6 A7 RDY/HLD CK2 DGND
28 42
B15 B14 B13 B12 B11 B10 B9 B8 A15 A14 A13 A12 A11 A10 A9 A8 VCC

TL/DD/8802-17

Top View

Order Number HPC16400E,
HPC16400V or HPC16400L
See NS Package Number E68B, EL68A or V68A

Wait States

The HPC16400 provides four software selectable Wait States that allow access to slower memories. The Wait States are selected by the state of two bits in the PSW register. Additionally, the RDY input may be used to extend the instruction cycle, allowing the user to interface with slow memories and peripherals. The DMA always uses one Wait State, independent of the value selected in the PSW.

Power Save Modes

Two power saving modes are available on the HPC16400: HALT and IDLE. In the HALT mode, all processor activities are stopped. In the IDLE mode, the on-board oscillator and timer T0 are active but all other processor activities are stopped. In either mode, on-board RAM, registers and I/O are unaffected.

HALT MODE

The HPC16400 is placed in the HALT mode under software control by setting bits in the PSW. All processor activities, including the clock and timers, are stopped. In the HALT mode, power requirements for the HPC16400 are minimal and the applied voltage (V_{CC}) may be decreased without altering the state of the machine. There are two ways of exiting the HALT mode: via the $\overline{\text{RESET}}$ or the NMI. The $\overline{\text{RESET}}$ input reinitializes the processor. Use of the NMI input will generate a vectored interrupt and resume operation from that point with no initialization. The HALT mode can be enabled or disabled by means of a control register HALT enable. To prevent accidental use of the HALT mode the HALT enable register can be modified only once.

IDLE MODE

The HPC16400 is placed in the IDLE mode through the PSW. In this mode, all processor activity, except the on-board oscillator and Timer T0, is stopped. The HPC16400 resumes normal operation upon timer T0 overflow. As with the HALT mode, the processor is also returned to full operation by the $\overline{\text{RESET}}$ or NMI inputs, but without waiting for oscillator stabilization.

HPC16400 Interrupts

Complex interrupt handling is easily accomplished by the HPC16400's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table I.

TABLE I. Interrupts

Vector/ Address	Interrupt Source	Arbitration Ranking
FFFF|FFFE	Reset	0
FFFD|FFFC	Nonmaskable Ext (NMI)	1
FFFB|FFFA	External on I2	2
FFF9|FFF8	External on I3	3
FFF7|FFF6	I4 + HDLC/DMA Error	4
FFF5|FFF4	Internal on Timers	5
FFF3|FFF2	Internal on UART	6
FFF1|FFF0	End of Message (EOM)	7

The 16400 contains arbitration logic to determine which interrupt will be serviced first if two or more interrupts occur simultaneously. Interrupts are serviced after the current instruction is completed except for the RESET which is serviced immediately.

The NMI interrupt will immediately stop DMA activity. Byte transfers in progress will finish thereby allowing an orderly transition to the interrupt service vector (see DMA description). The HDLC channels continue to operate, and the user must service data errors that might have occurred during the NMI service routine.

Interrupt Processing

Interrupts are serviced after the current instruction is completed except for the RESET, which is serviced immediately.

RESET is a level-sensitive interrupt. All other interrupts are edge-sensitive. NMI is positive-edge sensitive. The external interrupts on I2, I3 can be software selected to be rising or falling edge.

Interrupt Control Registers

The HPC16400 allows the various interrupt sources and conditions to be programmed. This is done through the various control registers. A brief description of the different control registers is given below.

INTERRUPT ENABLE REGISTER (ENIR)

RESET and the External Interrupt on I1 are non-maskable interrupts. The other interrupts can be individually enabled or disabled. Additionally, a Global Interrupt Enable Bit in the ENIR Register allows the Maskable interrupts to be collectively enabled or disabled. Thus, in order for a particular interrupt to request service, both the individual enable bit and the Global Interrupt bit (GIE) have to be set.

INTERRUPT PENDING REGISTER (IRPD)

The IRPD register contains a bit allocated for each interrupt vector. The occurrence of specified interrupt trigger conditions causes the appropriate bit to be set. There is no indication of the order in which the interrupts have been received. The bits are set independently of the fact that the interrupts may be disabled. IRPD is a Read/Write register. The bits corresponding to the maskable, external interrupts are normally cleared by the HPC16400 after servicing the interrupts.

For the interrupts from the on-board peripherals, the user has the responsibility of resetting the interrupt pending flags through software.

INTERRUPT CONDITION REGISTER (IRCD)

Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

Servicing the Interrupts

The Interrupt, once acknowledged, pushes the program counter (PC) onto the stack thus incrementing the stack pointer (SP) twice. The Global Interrupt Enable (GIE) bit is reset, thus disabling further interrupts. The program counter is loaded with the contents of the memory at the vector address and the processor resumes operation at this point. At the end of the interrupt service routine, the user does a RETI instruction to pop the stack, set the GIE bit and return to the main program. The GIE bit can be set in the interrupt service routine to nest interrupts if desired. *Figure 6* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A, B (except B12), D and R in the TRI-STATE condition. RESET is an active-low Schmitt trigger input. The processor vectors to FFFF:FFFE and resumes operation at the address contained at that memory location.

Timer Overview

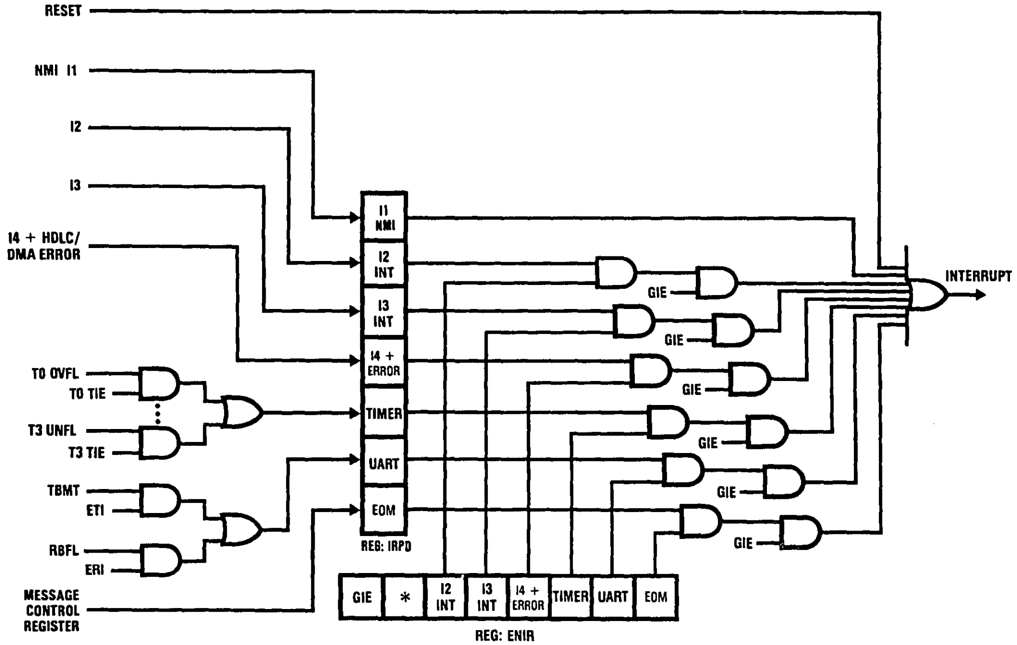
The HPC16400 contains a powerful set of flexible timers enabling the HPC16400 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16400 contains four 16-bit timers. Three of the timers have an associated 16-bit register. Timer T0 is a free-running timer, counting up at a fixed CKI/16 (Clock Input/16) rate. It is used for Watch Dog logic, high speed event capture, and to exit from the IDLE mode. Consequently, it cannot be stopped or written to under software control. Timer T0 permits precise measurements by means of the capture registers I2CR, I3CR, and I4CR. A control bit in the register T0CON configures timer T1 and its associated register R1 as capture registers I3CR and I2CR. The capture registers I2CR, I3CR, and I4CR respectively, record the value of timer T0 when specific events occur on the interrupt pins I2, I3, and I4. The control register IRCD programs the capture registers to trigger on either a rising edge or a falling edge of its respective input. The specified edge can also be programmed to generate an interrupt (see *Figure 7*).

The timers T2 and T3 have selectable clock rates. The clock input to these two timers may be selected from the following two sources: an external pin, or derived internally by dividing the clock input. Timer T2 has additional capability of being clocked by the timer T3 underflow. This allows the user to cascade timers T3 and T2 into a 32-bit timer/counter. The control register DIVBY programs the clock input to timers T2 and T3 (see *Figure 8*).

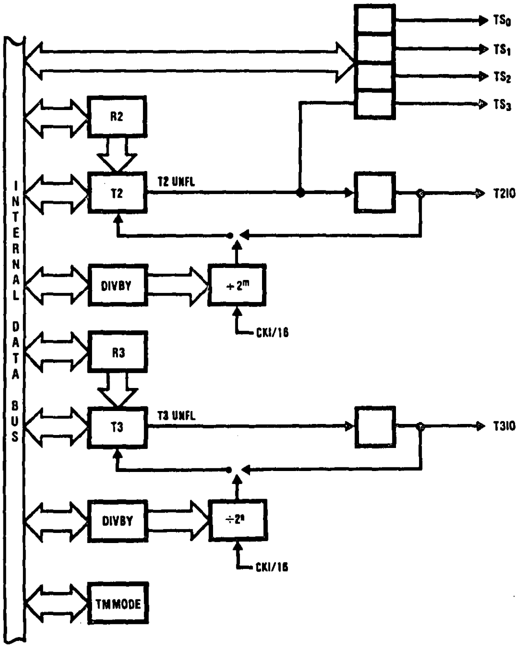
The timers T1 through T3 in conjunction with their registers form Timer-Register pairs. The registers hold the pulse duration values. All the Timer-Register pairs can be read from or written to. Each timer can be started or stopped under software control. Once enabled, the timers count down, and upon underflow, the contents of its associated register are automatically loaded into the timer.

Timer Overview (Continued)



TL/DD/8802-8

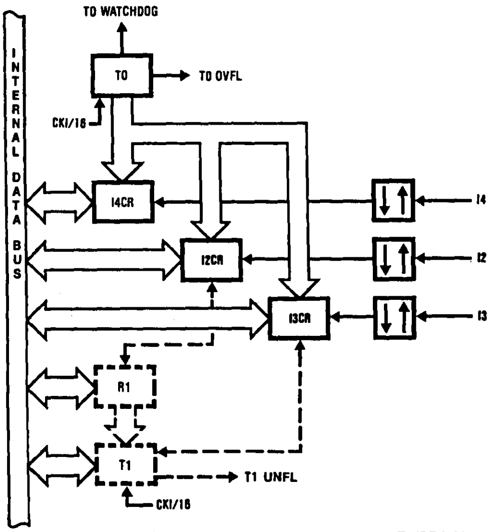
FIGURE 6. Interrupt Enable Logic



TL/DD/8802-10

FIGURE 8. Timers T2-T3 Block

Timer Overview (Continued)



TL/DD/8802-9

FIGURE 7. Timers T0–T1 Block

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16400 simplifies pulse generation and measurement. There are four synchronous timer outputs (TS0 through TS3) that work in conjunction with the timer T2. The synchronous timer outputs can be used either as regular outputs or individually programmed to toggle on timer T2 underflows (see *Figure 8*). Maximum output frequency for any timer output can be obtained by setting timer/register pair to zero. This then will produce an output frequency equal to ½ the frequency of the source used for clocking the timer.

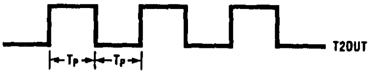
Timer Registers

There are four control registers that program the timers. The divide by (DIVBY) register programs the clock input to timers T2 and T3. The timer mode register (TMMODE) contains control bits to start and stop timers T1 through T3. It also contains bits to latch, acknowledge and enable interrupts from timers T0 through T3.

Timer Applications

The use of Pulse Width Timers for the generation of various waveforms is easily accomplished by the HPC16400.

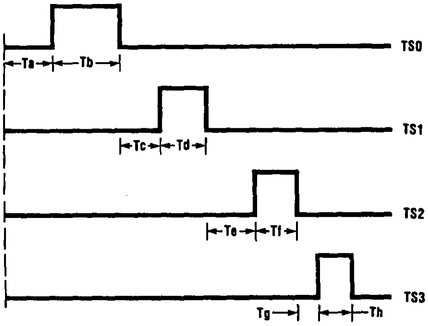
Frequencies can be generated by using the timer/register pairs. A square wave is generated when the register value is a constant. The duty cycle can be controlled simply by changing the register value.



TL/DD/8802-12

FIGURE 9. Square Wave Frequency Generation

Synchronous outputs based on Timer T2 can be generated on the 4 outputs TS0–TS3. Each output can be individually programmed to toggle on T2 underflow. Register R2 contains the time delay between events. *Figure 10* is an example of synchronous pulse train generation.



TL/DD/8802-13

FIGURE 10. Synchronous Pulse Generation

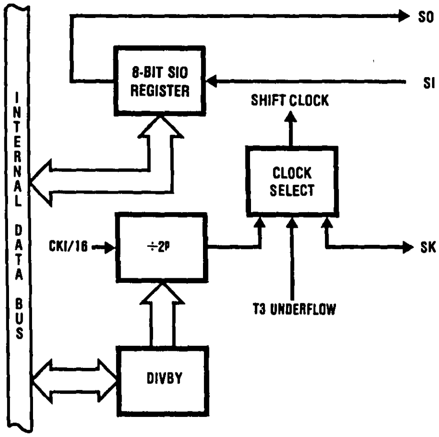
Watch Dog Logic

The Watch Dog Logic monitors the operations taking place and signals upon the occurrence of any illegal activity. The illegal conditions that trigger the Watch Dog logic are potentially infinite loops. Should the Watch Dog register not be written to before Timer T0 overflows twice, or more often than once every 4096 counts, an infinite loop condition is assumed to have occurred. The illegal condition forces the Watch Out (WO) pin low. The WO pin is an open drain output and can be connected to the RESET or NMI inputs or to the users external logic.

MICROWIRE/PLUS

MICROWIRE/PLUS is used for synchronous serial data communications (see *Figure 11*). MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register using SI as the input and SO as the output. SK is the clock for the serial shift register (SIO). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. A DONE flag indicates when the data shift is completed.

The MICROWIRE/PLUS capability enables it to interface with any of National Semiconductor's MICROWIRE peripherals (i.e., ISDN Transceivers, A/D converters, display drivers, EEPROMs).



TL/DD/8802-14

FIGURE 11. MICROWIRE/PLUS

MICROWIRE/PLUS Operation

The HPC16400 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16400 is the master or slave. The shift clock is generated when the HPC16400 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16400 is configured as a slave. When the HPC16400 is a master, the DIVBY register programs the frequency of the SK clock. The DIVBY register allows the SK clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz with CKI at 16 MHz.

The contents of the SIO register may be accessed through any of the memory access instructions. Data waiting to be transmitted in the SIO register is shifted out on the falling edge of the SK clock. Serial data on the SI pin is latched in on the rising edge of the SK clock.

HPC16400 UART

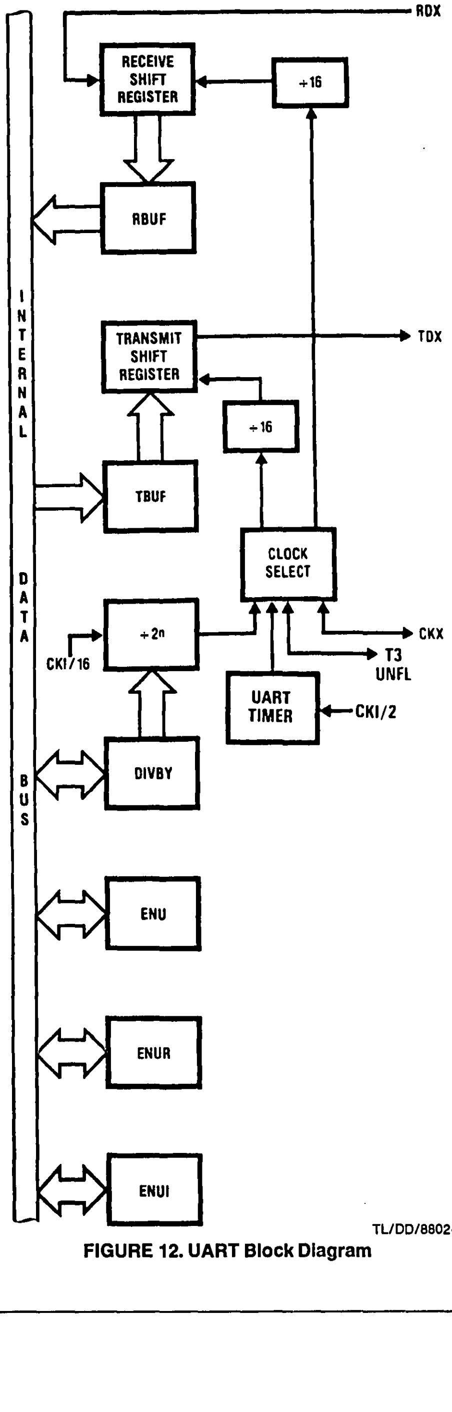
The HPC16400 contains a software programmable UART. The UART (see *Figure 12*) consists of a transmit shift register, a receiver shift register and five addressable registers, as follows: a transmit buffer register (TBUF), a receiver buffer register (RBUF), a UART control and status register (ENU), a UART receive control and status register (ENUR) and a UART interrupt and clock source register (ENUI). The ENU register contains flags for transmit and receive functions; this register also determines the length of the data frame (8 or 9 bits) and the value of the ninth bit in transmission. The ENUR register flags framing and data overrun errors while the UART is receiving. Other functions of the ENUR register include saving the ninth bit received in the data frame and enabling or disabling the UART's Wake-up Mode of operation. The determination of an internal or external clock source is done by the ENUI register, as well as selecting the number of stop bits and enabling or disabling transmit and receive interrupts.

The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register, or a special dedicated timer. The baud rate may be selected from a range of 8 baud to 208.3 kbaud. Without having to select a special baud rate crystal, all standard baud rates from 75 baud to 38.4 kbaud can be generated. The external baud clock source comes from the CKX pin. The Transmitter and Receiver can be run at different rates by selecting one to operate from the internal clock and the other from an external source. The special dedicated timer is selected as an external source.

The HPC16400 UART supports two data formats. The first format for data transmission consists of one start bit, eight data bits and one or two stop bits. The second data format for transmission consists of one start bit, nine data bits, and one or two stop bits. Receiving formats differ from transmission only in that the Receiver always requires only one stop bit in a data frame.

UART Wake-up Mode

The HPC16400 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16400 to be networked with other processors. Typically in such environments, the messages consist of addresses and actual data. Addresses are specified by having the ninth bit in the data frame set to 1. Data in the message is specified by having the ninth bit in the data frame reset to 0.



TL/DD/8802–15

FIGURE 12. UART Block Diagram

UART Wake-up Mode (Continued)

The UART monitors the communication stream looking for addresses. When the data word with the ninth bit set is received, the UART signals the HPC16400 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accept subsequent data.

Programmable Serial Decoder Interface

The programmable serial decoder interface allows the two HDLC channels to be used with devices employing several popular serial protocols for point-to-point and multipoint data exchanges. These protocols combine the 'B' and 'D' channels onto common pins—received data, transmit data, clock and Sync, which normally occurs at an 8 KHz rate and provides framing for the particular protocol.

The decoder uses the serial link clock and Sync signals to generate internal enables for the 'D' and 'B' channels, thereby allowing the HDLC channels to access the appropriate channel data from the multiplexed link.

HDLC Channel Description

HDLC/DMA Structure

HDLC 1		HDLC 2	
HDLC1 Receive	HDLC1 Transmit	HDLC2 Receive	HDLC2 Transmit
DMAR1	DMAT1	DMAR2	DMAT2

GENERAL INFORMATION

Both HDLC channels on the HPC16400 are identical and operate up to 4.65 Mbps. When used in an ISDN basic access application, HDLC channel #1 has been designated for use with the 16 Kbps D-channel or either B channel and HDLC #2 can be used with either of the 64 Kbps B-channels. If the 'D' and 'B' channels are present on a common serial link, the programmable serial decoder interface generates the necessary enable signals needed to access the D and B channel data.

There are two sources for the receive and transmit channel enable signals. They can be internally generated from the serial decoder interface or they can be externally enabled.

LAPD, the Link Access Protocol for the D channel is derived from the X.25 packet switching LAPB protocol. LAPD specifies the procedure for a terminal to use the D channel for the transfer of call control or user-data information. The procedure is used in both point-to-point and point-to-multipoint configurations. On the 16400, the HDLC controller contains user programmable features that allow for the efficient processing of LAPD Information.

HDLC Channel Pin Description

Each HDLC channel has the following pins associated with it.

HCK — HDLC Channel Clock Input Signal.

RX — Receive Serial Data Input. Data latched on the negative HCK edge.

REN/RHCK — HDLC Channel Receiver Enable Input/Receiver Clock Input.

TEN — HDLC Channel Transmitter Enable Input.

TX — Transmit Serial Data Output. Data clocked out on the positive HCK edge. Data (not including CRC) is sent LSB first. TRI-STATE when transmitter not enabled.

HDLC Functional Description

TRANSMITTER DESCRIPTION

Data information is transferred from external memory through the DMA controller into the transmit buffer register from where it is loaded into a 8-bit serial shift register. The CRC is computed and appended to the frame prior to the closing flag being transmitted. Data is output at the TX output pin. If no further transmit commands are given the transmitter sends out continous flags, aborts, or the idle pattern as selected by the control register.

An interrupt is generated when the DMA has transferred the last byte from RAM to the HDLC channel for a particular message or on a transmit error condition. An associated transmit status register will contain the status information indicating the specific interrupt source.

TRANSMITTER FEATURES

Interframe fill: the transmitter can send either continuous '1's or repeated flags or aborts between the closing flag of one packet and the opening flag of the next. When the CPU commands the transmitter to open a new frame, the interframe fill is terminated immediately.

Abort: the 7 '1's abort sequence will be immediately sent on command from the CPU or on an underrun condition in the DMA. If required, it may be followed by a new opening flag to send another packet.

Bit/Byte boundaries: The message length between packet headers may have any number of bits and is not confined to an integral number of bytes. Three bits in the control register are used to indicate the number of valid bits in the last byte. These bits are loaded by the users software.

RECEIVER DESCRIPTION

Data is input to the receiver on the RX pin. The receive clock can be externally input at either the HCK pin or the REN/RHCK pin.

Incoming data is routed through one of several paths depending on whether it is the flag, data, or CRC.

Once the receiver is enabled it waits for the opening flag of the incoming frame, then starts the zero bit deletion, addressing handling and CRC checking. All data between the flags is shifted through two 8-bit serial shift registers before being loaded into the buffer register. The user programmable address register values are compared to the incoming data while it resides in the shift registers. If an address match occurs or if operating in the transparent address recognition mode, the DMA channel is signaled that attention is required and the data is transferred by it to external memory. Appropriate interrupts are generated to the CPU on the reception of a complete frame, or on the occurance of a frame error.

The receive interrupt, in conjunction with status data in the control registers allows interrupts to be generated on the following conditions—frame length error, CRC error, receive error, abort and receive complete.

RECEIVER FEATURES

Flag sharing: the closing flag of one packet may be shared as the opening flag of the next. Receiver will also be able to share a zero between flags—011111101111110 is a valid two flag sequence for receive (not transmit).

HDLC Functional Description (Continued)

Interframe fill: the receiver automatically accepts either repeated flags, repeated aborts, or all '1's as the interframe fill.

Idle: Reception of successive flags as the interframe fill sequence to be signaled to the user by setting the Flag bit in the Receiver Status register.

Short Frame Rejection: Reception of greater than 2 bytes but less than 4 bytes between flags will generate a frame error, terminating reception of the current frame and setting the Frame Error (FER) status bit in the Receive Control and Status register. Reception of less than 2 bytes will be ignored.

Abort: the 7 '1's abort sequence (received with no zero insertion) will be immediately recognized and will cause the receiver to reinitialize and return to searching the incoming data for an opening flag. Reception of the abort will cause the abort status bit in the Interrupt Error Status register to be set and will signal an End of Message (EOMR).

Bit/Byte boundaries: The message length between packet headers may have any number of bits and it is not confined to an integral number of bytes. Three bits in the status register are used to indicate the number of valid bits in the last byte.

Addressing: Two user programmable bytes are available to allow frame address recognition on the two bytes immediately following the opening flag. When the received address matches the programmed value(s), the frame is passed through to the DMA channel. If no match occurs, the received frame address information is disregarded and the receiver returns to searching for the next opening flag and the address recognition process starts anew.

Support is provided to allow recognition of the broadcast address sequence of seven consecutive 1's. Additionally, a transparent mode of operation is available where no address decoding is done.

HDLC INTERRUPT CONDITIONS

The end of message interrupt (EOM) indicates that a complete frame has been received or transmitted by the HDLC controller. Thus, there are four separate sources for this interrupt, two each from each HDLC channel. The Message Control Register contains the pending bits for each source.

The HDLC/DMA error interrupt groups several related error conditions. Error conditions from both transmit/receiver channels can cause this interrupt, and the possible sources each have a status bit in the error status register that is set on the occurrence of an error. The bit must then be serviced by the user.

HDLC CHANNEL CLOCK

Each HDLC channel uses the falling edge of the clock to sample the receive data. Outgoing transmit data is shifted out on the rising edge of the external clock. The maximum data rate when using the externally provided clocks is 4.65 Mb/s.

CYCLIC REDUNDACY CHECK

There are two standard CRC codes used in generating the 16-bit Frame Check Sequence (FCS) that is appended to the end of the data frame. Both codes are supported and the user selects the error checking code to be used through software control (HDLC control reg). The two error checking polynomials available are:

(1) CRC—16 (x16 + x15 + x2 +1)

(2) CCITT CRC (x16 + x12 + x5 + 1)

SYNCHRONOUS BYPASS MODE

When the BYPAS bit is set in the HDLC control register, all HDLC framing/formatting functions for the specified HDLC channel are disabled.

This allows byte-oriented data to be transmitted and received synchronously thus "bypassing" the HDLC functions.

LOOP BACK OPERATIONAL MODE

The user has the ability, by setting the appropriate bit in the register to internally route the transmitter output to the receiver input, and to internally route the RX pin to the TX pin.

DMA Controller*

GENERAL INFORMATION

The HPC16400 uses Direct Memory Access (DMA) logic to facilitate data transfer between the 2 full Duplex HDLC channels and external packet RAM. There are four DMA channels to support the four individual HDLC channels. Control of the DMA channels is accomplished through registers which are configured by the CPU. These control registers define specific operation of each channel and changes are immediately reflected in DMA operation. In addition to individual control registers, global control bits (MSS and MSSC in Message Control Register) are available so that the HDLC channels may be globally controlled.

The DMA issues a bus request to the CPU when one or more of the individual HDLC channels request service. Upon receiving a bus acknowledge from the CPU, the DMA completes all requests pending and any requests that may have occurred during DMA operation before returning control to the CPU. If no further DMA transfers are pending, the DMA relinquishes the bus and the CPU can again initiate a bus cycle.

Four memory expansion bits have been added for each of the four channels to support data transfers into the expanded memory bank areas.

The DMA has priority logic for a DMA requesting service. The priorities are:

1st priority	Receiver channel 1
2nd priority	Transmit channel 1
3rd priority	Receive channel 2
4th priority	Transmit channel 2

RECEIVER DMA OPERATION

The receiver DMA consists of a shift register and two buffers. A receiver DMA operation is initiated by the buffer registers. Once a byte has been placed in a buffer register from the HDLC, it generates a request and upon obtaining control of the bus, the DMA places the byte in external memory.

RECEIVER REGISTERS

All the following registers are Read/Write

A. Frame Length Register

This user programmable 16-bit register contains the maximum number of bytes to be placed in a data "block". If this number is exceeded, a Frame Too Long (FTLR1, FTLR2) error is generated. This register is decremented by one each Receiver DMA cycle.

*The specific registers and/or register names may have changed. Please contact the factory for updated information.

DMA Controller (Continued)

B. CNTRL ADDR 1
DATA ADDR 1
CNTRL ADDR 2
DATA ADDR 2

For split frame operation, the CNTRL ADDR register contains the external memory address where the Frame Header (Control & Address fields) are to be stored and the DATA ADDR register contains an equivalent address for the Information field.

For non-split frame operation, the CNTRL and DATA ADDR registers each contain the external memory address for entire frames.

TRANSMITTER DMA OPERATION

The transmitter DMA consists of a shift register and two buffers. A transmitter DMA cycle is initiated by the TX data buffers. The TX data buffers generate a request when either one is empty and the DMA responds by placing a byte in the buffer. The HDLC transmitter can then accept the byte to send when needed, upon which the DMA will issue another request, resulting in a subsequent DMA cycle.

TRANSMITTER REGISTERS

The following registers are Read/Write:

A. Field Address 1 (FA1)
# Bytes Field 1 (NBF1)
Field Address (FA2)
Bytes Field 2 (NBF2)

FA1 and FA2 are starting addresses of blocks of information to transmitter.

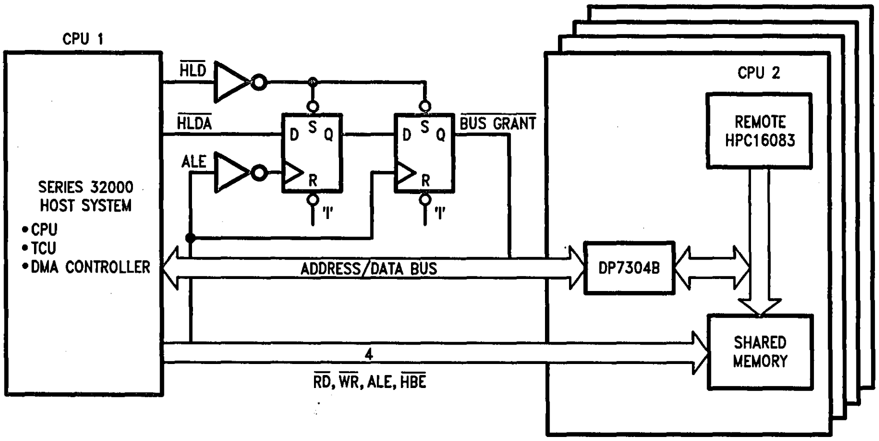
NBF1 and NBF2 are the number of bytes in the block to be transmitted.

Shared Memory Support

Shared memory access provides a rapid technique to exchange data. It is effective when data is moved from a peripheral to memory or when data is moved between blocks of memory. A related area where shared memory access proves effective is in multiprocessing applications where two CPUs share a common memory block. The HPC16400 supports shared memory access with two pins. The pins are the RDY/$\overline{\text{HLD}}$ input pin and the $\overline{\text{HLDA}}$ output pin. The user can software select either the Hold or Ready function on the RDY/$\overline{\text{HLD}}$ pin by the state of a control bit. The $\overline{\text{HLDA}}$ output is multiplexed onto port B.

The host uses DMA to interface with the HPC16400. The host initiates a data transfer by activating the $\overline{\text{HLD}}$ input of the HPC16400. In response, the HPC16400 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{\text{HLDA}}$) from the HPC16400 indicating that the sytem bus is free. On receiving the acknowledge, the host can rapidly transfer data into, or out of, the shared memory by using a conventional DMA controller. Upon completion of the message transfer, the host removes the HOLD request and the HPC16400 resumes normal operations.

Figure 13 illustrates an application of the shared memory interface between the HPC16400 and a Series 32000 system. To insure proper operation, the interface logic shown is recommended as the means for enabling and disabling the user's bus.



TL/DD/8802-16

FIGURE 13. Shared Memory Application: HPC16400 Interface to Series 32000 System

Memory

The HPC16400 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 256 bytes of RAM available on the chip itself.

Program memory addressing is accomplished by the 16-bit program counter on a byte basis. Memory can be addressed directly by instructions or indirectly through the B, X and SP registers. Memory can be addressed as words or bytes. Words are always addressed on even-byte boundaries. The HPC16400 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The HPC16400 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table II.

Extended Memory Addressing

If more than 64k of addressing is desired in a HPC16400 system, on board bank select circuitry is available that allows four I/O lines of Port B (B8, B9, B13, B14) to be used in extending the address range. This gives the user a main routine area of 32k and 16 banks of 32k each for subroutine and data, thus getting a total of 544k of memory.

Note: If all four lines are not needed for memory expansion, the unused lines can be used as general purpose inputs.

The Extended Memory Addressing mode is entered by setting the EMA control bit in the Message Control Register. If this bit is not set, the port B lines (B8, B9, B13, B14) are available as general purpose I/O or synchronous outputs as selected by the BFUN register.

The main memory area contains the interrupt vectors & service routines, stack memory, and common memory for the bank subroutines to use. The 16 banks of memory can contain program or data memory (note: since the on chip resources are mapped into addresses 0000-01FF, the first 512 bytes of each bank are not usable, actual available memory is 536.5k).

TABLE II. Memory Map

FFFF-FFF0	Interrupt Vectors	
FFEF-FFD0	JSRP Vectors	
FFCF-FFCE : : 0201-0200	External Expansion	USER MEMORY
01FF-01FE : : 01C1-01C0	On Chip RAM	USER RAM
01B8	Error Status	HDLC # 2
01B6	Receiver Status	
01B4	HDLC Cntrl	
01B2	Recr Addr Comp Reg 2	
01B0	Recr Addr Comp Reg 1	
01A8	Error Status	HDLC # 1
01A6	Receiver Status	
01A4	HDLC Cntrl	
01A2	Recr Addr Comp Reg 2	
01A0	Recr Addr Comp Reg 1	
0195-0194	Watch Dog Register	Watch Dog Logic
0193-0192	T0CON Register	Timer Block T0-T3
0191-0190	TMMODE Register	
018F-018E	DIVBY Register	
018D-018C	T3 Timer	
018B-018A	R3 Register	
0189-0188	T2 Timer	
0187-0186	R2 Register	
0185-0184	I2CR Register/ R1	
0183-0182	I3CR Register/ T1	
0181-0180	I4CR Register	
017F-017E	UART Counter	UART Timer
017D-017C	UART Register	
0179-0178	# Bytes 2	DMAT # 2 (Xmit)
0177-0176	Field Addr 2	
0175-0174	# Bytes 1	
0173-0172	Field Addr 1	
0171-0170	Xmit Cntrl & Status	
016B-016A	Frame Length	DMAR # 2 (Recv)
0169-0168	Data Addr 2	
0167-0166	Cntrl Addr 2	
0165-0164	Data Addr 1	
0163-0162	Cntrl Addr 1	
0161-0160	Recv Cntrl & Status	
0159-0158	# Bytes 2	DMAT # 1 (Xmit)
0157-0156	Field Addr 2	
0155-0154	# Bytes 1	
0153-0152	Field Addr 1	
0151-0150	Xmit Cntrl & Status	
014B-014A	Frame Length	DMAR # 1 (Recv)
0149-0148	Data Addr 2	
0147-0146	Cntrl Addr 2	
0145-0144	Data Addr 1	
0143-0142	Cntrl Addr 1	
0141-0140	Recv Cntrl & Status	
0128	ENUR Register	UART
0126	TBUF Register	
0124	RBUF Register	
0122	ENUI Register	
0120	ENU Register	
010E	Port R Pins	PORTS R & D
010C	DIR R Register	
010A	Port R Data Register	
0106	Serial Decoder/Enable Configuration Reg	
0104	Message Pending	
0102	Message Control	
0100	Port D Pins	
00F5-00F4	BFUN Register	PORT B
00F3-00F2	DIR B Register	
00E3-00E2	Port B	
00DE	Microcode ROM Dump	PORT CONTROL & INTERRUPT CONTROL REGISTERS
00DD-00DC	Halt Enable Register	
00D8	Port I Input Register	
00D6	SIO Register	
00D4	IRCD Register	
00D2	IRPD Register	
00D0	ENIR Register	
00CF-00CE	X Register	HPC16040 CORE REGISTERS
00CD-00CC	B Register	
00CB-00CA	K Register	
00C9-00C8	A Register	
00C7-00C6	PC Register	
00C5-00C4	SP Register	
00C3-00C2	(Reserved)	
00C0	PSW Register	
00BF-00BE : : 0001-0000	On Chip RAM	USER RAM

Design Considerations

Designs using the HPC family of 16-bit high speed CMOS microcontrollers need to follow some general guidelines on usage and board layout.

Floating inputs are a frequently overlooked problem. CMOS inputs have extremely high impedance and, if left open, can float to any voltage. You should thus tie unused inputs to V_{CC} or ground, either through a resistor or directly. Unlike the inputs, unused outputs should be left floating to allow the output to switch without drawing any DC current.

To reduce voltage transients, keep the supply line's parasitic inductances as low as possible by reducing trace lengths, using wide traces, ground planes, and by decoupling the supply with bypass capacitors. In order to prevent additional voltage spiking, this local bypass capacitor must exhibit low inductive reactance. You should therefore use high frequency ceramic capacitors and place them very near the IC to minimize wiring inductance.

- Keep V_{CC} bus routing short. When using double sided or multilayer circuit boards, use ground plane techniques.
- Keep ground lines short, and on PC boards make them as wide as possible, even if trace width varies. Use separate ground traces to supply high current devices such as relay and transmission line drivers.
- In systems mixing linear and logic functions and where supply noise is critical to the analog components' performance, provide separate supply buses or even separate supplies.
- When using local regulators, bypass their inputs with a tantalum capacitor of at least 1 μF and bypass their outputs with a 10 μF to 50 μF tantalum or aluminum electrolytic capacitor.
- If the system uses a centralized regulated power supply, use a 10 μF to 20 μF tantalum electrolytic capacitor or a 50 μF to 100 μF aluminum electrolytic capacitor to decouple the V_{CC} bus connected to the circuit board.
- Provide localized decoupling. For random logic, a rule of thumb dictates approximately 10 nF (spaced within 12 cm) per every two to five packages, and 100 nF for every 10 packages. You can group these capacitances, but it's more effective to distribute them among the ICs. If the design has a fair amount of synchronous logic with outputs that tend to switch simultaneously, additional decoupling might be advisable. Octal flip-flop and buffers in bus-oriented circuits might also require more decoupling. Note that wire-wrapped circuits can require more decoupling than ground plane or multilayer PC boards.

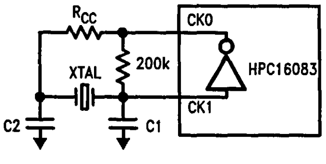
A recommended crystal oscillator circuit to be used with the HPC is shown below. See table for recommended component values. The recommended values given in the table below have yielded consistent results and are made to match a crystal with a 20 pF load capacitance, with some small allowance for layout capacitance.

A recommended layout for the oscillator network should be as close to the processor as physically possible, entirely within 1″ distance. This is to reduce lead inductance from long PC traces, as well as interference from other components, and reduce trace capacitance. The layout should contain a large ground plane either on the top or bottom surface of the board to provide signal shielding, and a convenient location to ground both the HPC, and the case of the crystal.

It is very critical to have an extremely clean power supply for the HPC crystal oscillator. Ideally one would like a V_{CC} and ground plane that provide low inductance power lines to the chip. The power planes in the PC board should be decoupled with three decoupling capacitors as close to the chip as possible. A 1.0 μF, a 0.1 μF, and a 0.001 μF dipped mica or ceramic cap mounted as close to the HPC as is physically possible on the board, using the shortest leads, or surface mount components. This should provide a stable power supply, and noiseless ground plane which will vastly improve the performance of the crystal oscillator network.

HPC Oscillator Table

f_C (MHz)	R_{CC} (Ω)	C1 (pF)	C2 (pF)
2	50	82	100
4	50	62	75
6	50	50	56
8	50	47	50
10	50	39	50
12	0	39	39
14	0	33	39
16	0	33	39
18	0	33	33
20	0	33	33
22	0	27	39
24	0	27	39
26	0	27	33
28	0	27	33
30	0	27	27



TL/DD/8802-35

Crystal Specifications:
"AT" cut, parallel resonant crystals tuned to the desired frequency with the following specifications are recommended:

Series Resistance < 65Ω

Loading Capacitance: C_L = 20 pF

HPC16400 CPU

The HPC16400 CPU has a 16-bit ALU and six 16-bit registers.

Arithmetic Logic Unit (ALU)

The ALU is 16 bits wide and can do 16-bit add, subtract and shift or logic AND, OR and exclusive OR in one timing cycle. The ALU can also output the carry bit to a 1-bit C register.

Accumulator (A) Register

The 16-bit A register is the source and destination register for most I/O, arithmetic, logic and data memory access operations.

Address (B and X) Registers

The 16-bit B and X registers can be used for indirect addressing. They can automatically count up or down to sequence through data memory.

Boundary (K) Register

The 16-bit K register is used to set limits in repetitive loops of code as register B sequences through data memory.

Stack Pointer (SP) Register

The 16-bit SP register is the stack pointer that addresses the stack. The SP register is incremented by two for each push or call and decremented by two for each pop or return. The stack can be placed anywhere in user memory and be as deep as the available memory permits.

Program (PC) Register

The 16-bit PC register addresses program memory.

Addressing Modes

ADDRESSING MODES—ACCUMULATOR AS DESTINATION

Register Indirect

This is the "normal" mode of addressing for the HPC16400 (instructions are single-byte). The operand is the memory addressed by the B register (or X register for some instructions).

Direct

The instruction contains an 8-bit or 16-bit address field that directly points to the memory for the operand.

Indirect

The instruction contains an 8-bit address field. The contents of the WORD addressed points to the memory for the operand.

Indexed

The instruction contains an 8-bit address field and an 8- or 16-bit displacement field. The contents of the WORD addressed is added to the displacement to get the address of the operand.

Immediate

The instruction contains an 8-bit or 16-bit immediate field that is used as the operand.

Register Indirect (Auto Increment and Decrement)

The operand is the memory addressed by the X register. This mode automatically increments or decrements the X register (by 1 for bytes and by 2 for words).

Register Indirect (Auto Increment and Decrement) with Conditional Skip

The operand is the memory addressed by the B register. This mode automatically increments or decrements the B register (by 1 for bytes and by 2 for words). The B register is then compared with the K register. A skip condition is generated if B goes past K.

ADDRESSING MODES—DIRECT MEMORY AS DESTINATION

Direct Memory to Direct Memory

The instruction contains two 8- or 16-bit address fields. One field directly points to the source operand and the other field directly points to the destination operand.

Immediate to Direct Memory

The instruction contains an 8- or 16-bit address field and an 8- or 16-bit immediate field. The immediate field is the operand and the direct field is the destination.

Double Register Indirect using the B and X Registers

Used only with Reset, Set and IF bit instructions; a specific bit within the 64 kbyte address range is addressed using the B and X registers. The address of a byte of memory is formed by adding the contents of the B register to the most significant 13 bits of the X register. The specific bit to be modified or tested within the byte of memory is selected using the least significant 3 bits of register X.

HPC Instruction Set Description

Mnemonic	Description	Action	
ARITHMETIC INSTRUCTIONS			
ADD	Add	MA+Meml → MA	carry → C
ADDS	Add short imm8	MA+imm8 → MA	carry → C
ADC	Add with carry	MA+Meml+C → MA	carry → C
DADC	Decimal add with carry	MA+Meml+C → MA (Decimal)	carry → C
SUBC	Subtract with carry	MA−Meml+C → MA	carry → C
DSUBC	Decimal subtract w/carry	MA−Meml+C → MA (Decimal)	carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C	
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C	
DIVD	Divide Double Word (unsigned)	(x8 MA)/Meml → MA, rem → X, 0 → K	carry → C
IFEQ	If equal	Compare MA & Meml, Do next if equal	
IFGT	If greater than	Compare MA & Meml, Do next if MA → Meml	
AND	Logical and	MA and Meml → MA	
OR	Logical or	MA or Meml → MA	
XOR	Logical exclusive-or	MA xor Meml → MA	
MEMORY MODIFY INSTRUCTIONS			
INC	Increment	Mem + 1 → Mem	
DECSZ	Decrement, skip if 0	Mem −1 → Mem, Skip next if Mem = 0	
BIT INSTRUCTIONS			
SBIT	Set bit	1 → Mem.bit (bit is 0 to 7 immediate)	
RBIT	Reset bit	0 → Mem.bit	
IFBIT	If bit	If Mem.bit is true, do next instr.	
MEMORY TRANSFER INSTRUCTIONS			
LD	Load	Meml → MA	
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X	
ST	Store to Memory	MA → Mem	
X	Exchange	A ←→ Mem; Mem ←→ Mem	
	Exchange, incr/decr X	A ←→ Mem(X), X ±1 (or 2) → X	
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP	
POP	Pop Stack to Memory	SP−2 → SP, W(SP) → W	
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K	
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ←→ A,B±1 (or 2) → B, Skip next if B greater/less than K	
REGISTER LOAD IMMEDIATE INSTRUCTIONS			
LD A	Load A immediate	imm → A	
LD B	Load B immediate	imm → B	
LD K	Load K immediate	imm → K	
LD X	Load X immediate	imm → X	
LD BK	Load B and K immediate	imm → B,imm → K	
ACCUMULATOR AND C INSTRUCTIONS			
CLR A	Clear A	0 → A	
INC A	Increment A	A + 1 → A	
DEC A	Decrement A	A − 1 → A	
COMP A	Complement A	1's complement of A → A	
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ←→ A3:0	
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C	
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C	
SHR A	Shift A right	0 → A15 → ... → A0 → C	
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0	
SC	Set C	1 → C	
RC	Reset C	0 → C	
IFC	IF C	Do next if C = 1	
IFNC	IF not C	Do next if C = 0	

HPC Instruction Set Description (Continued)

Mnemonic	Description	Action
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → W(SP),SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → W(SP),SP+2 → SP,PC+ # → PC (#is +1024 to −1023)
JSRL	Jump subroutine long	PC → W(SP),SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +256 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC ← PC + 1
RET	Return	SP−2 → SP,W(SP) → PC
RETS	Return then skip next	SP−2 → SP,W(SP) → PC, & skip
RETI	Return from interrupt	SP−2 → SP,W(SP) → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8 or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
Meml is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data

Memory Usage

Number Of Bytes For Each Instruction (number in parenthesis is 16-Bit field)

	Using Accumulator A						To Direct Memory			
	Reg Indir.		Direct	Indir	Index	Immed.	Direct		Immed.	
	(B)	(X)					*	**	*	**
LD	1	1	2(4)	3	4(5)	2(3)	3(5)	5(6)	3(4)	5(6)
X	1	1	2(4)	3	4(5)	—	—	—	—	—
ST	1	1	2(4)	3	4(5)	—	—	—	—	—
ADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
SBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DADC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
DSBC	1	2	3(4)	3	4(5)	4(5)	4(5)	5(6)	4(5)	5(6)
ADD	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
MULT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
DIV	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFEQ	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
IFGT	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
AND	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
OR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)
XOR	1	2	3(4)	3	4(5)	2(3)	4(5)	5(6)	4(5)	5(6)

*8-bit direct address
**16-bit direct address

Instructions that modify memory directly

	(B)	(X)	Direct	Indir	Index	B&X
SBIT	1	2	3(4)	3	4(5)	1
RBIT	1	2	3(4)	3	4(5)	1
IFBIT	1	2	3(4)	3	4(5)	1
DECSZ	3	2	2(4)	3	4(5)	
INC	3	2	2(4)	3	4(5)	

Immediate Load Instructions

	Immed.
LD B,*	2(3)
LD X,*	2(3)
LD K,*	2(3)
LD BK,*,*	3(5)

Memory Usage (Continued)

Register Indirect Instructions with Auto Increment and Decrement

Register B With Skip		
	(B+)	(B−)
LDS A,*	1	1
XS A,*	1	1

Register X		
	(X+)	(X−)
LD A,*	1	1
X A,*	1	1

Instructions Using A and C

CLR	A	1
INC	A	1
DEC	A	1
COMP	A	1
SWAP	A	1
RRC	A	1
RLC	A	1
SHR	A	1
SHL	A	1
SC	C	1
RC	C	1
IFC	C	1
IFNC	C	1

Transfer of Control Instructions

JSRP	1
JSR	2
JSRL	3
JP	1
JMP	2
JMPL	3
JID	1
JIDW	1
NOP	1
RET	1
RETS	1
RETI	1

Stack Reference Instructions

	Direct
PUSH	2
POP	2

Code Efficiency

One of the most important criteria of a single chip microcontroller is code efficiency. The more efficient the code, the more features that can be put on a chip. The memory size on a chip is fixed so if code is not efficient, features may have to be sacrificed or the programmer may have to buy a larger, more expensive version of the chip.

The HPC16400 has been designed to be extremely code-efficient. The HPC16400 looks very good in all the standard coding benchmarks; however, it is not realistic to rely only on benchmarks. Many large jobs have been programmed onto the HPC16400, and the code savings over other popular microcontrollers has been considerable.

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16400 are single-byte. There are two especially code-saving instructions:

JP is a 1-byte jump. True, it can only jump within a range of plus or minus 32, but many loops and decisions are often within a small range of program memory. Most other micros need 2-byte instructions for any short jumps.

JSRP is a 1-byte call subroutine. The user makes a table of his 16 most frequently called subroutines and these calls will only take one byte. Most other micros require two and even three bytes to call a subroutine. The user does not have to decide which subroutine addresses to put into his table; the assembler can give him this information.

EFFICIENT SUBROUTINE CALLS

The 2-byte JSR instructions can call any subroutine within plus or minus 1k of program memory.

MULTIFUNCTION INSTRUCTIONS FOR DATA MOVEMENT AND PROGRAM LOOPING

The HPC16400 has single-byte instructions that perform multiple tasks. For example, the XS instruction will do the following:

1. Exchange A and memory pointed to by the B register
2. Increment or decrement the B register
3. Compare the B register to the K register
4. Generate a conditional skip if B has passed K

The value of this multipurpose instruction becomes evident when looping through sequential areas of memory and exiting when the loop is finished.

BIT MANIPULATION INSTRUCTIONS

Any bit of memory, I/O or registers can be set, reset or tested by the single byte bit instructions. The bits can be addressed directly or indirectly. Since all registers and I/O are mapped into the memory, it is very easy to manipulate specific bits to do efficient control.

DECIMAL ADD AND SUBTRACT

This instruction is needed to interface with the decimal user world.

It can handle both 16-bit words and 8-bit bytes.

The 16-bit capability saves code since many variables can be stored as one piece of data and the programmer does not have to break his data into two bytes. Many applications store most data in 4-digit variables. The HPC16400 supplies 8-bit byte capability for 2-digit variables and literal variables.

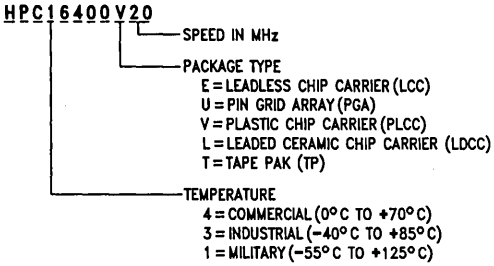
MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16400 has 16-bit multiply, 16-bit by 16-bit divide, and 32-bit by 16-bit divide instructions. This saves both code and time. Multiply and divide can use immediate data or data from memory. The ability to multiply and divide by immediate data saves code since this function is often needed for scaling, base conversion, computing indexes of arrays, etc.

Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. The number HPC16400 has been generally used throughout this datasheet to represent the whole family of parts. The following chart explains how to order various options available when ordering HPC family members.

Note: All options may not currently be available.


HPC16400V20
SPEED IN MHz
PACKAGE TYPE
E = LEADLESS CHIP CARRIER (LCC)
U = PIN GRID ARRAY (PGA)
V = PLASTIC CHIP CARRIER (PLCC)
L = LEADED CERAMIC CHIP CARRIER (LDCC)
T = TAPE PAK (TP)
TEMPERATURE
4 = COMMERCIAL (0°C TO +70°C)
3 = INDUSTRIAL (−40°C TO +85°C)
1 = MILITARY (−55°C TO +125°C)

TL/DD/8802-18

FIGURE 15. HPC Family Part Numbering Scheme

EXAMPLES

HPC46400V20—Commercial temp (0° to +70°C), PLCC

HPC16400L20—Military temp (−55°C to +125°C), LCC

Development Support

DEVELOPMENT SYSTEM

The Microcontroller On Line Emulator (MOLE™) is a low cost development system and emulator for all microcontroller products. These include COPs and the HPC family of products. The development system consists of a BRAIN Board, Personality Board and optional host software.

The purpose of the development system is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both software and hardware debugging of the system.

It is a self-contained computer with its own firmware which provides for all system operation, emulation control, communication, PROM programming and diagnostic operations.

It contains three serial ports to optionally connect to a terminal, a host system, a printer or modem, or to connect to other development systems in a multi-development system environment.

The development system can be used in either a stand alone mode or in conjunction with a selected host systems using PC-DOS communicating via a RS-232 port.

HOW TO ORDER

To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB2	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBMR	Relocatible Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM PC	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001
	MOLE-HPC-VMS	Assembler, Loader, Librarian for VAX/VMS	HPC Software User's Manual and 9 Track Tape	424410836-001
	MOLE-HPC-VMS-C	C Compiler for VAX/VMS	HPC Software User's Manual and 9 Track Tape (Includes Assembler)	424410883-001
	424410897-001	Users Manual		

Development Support (Continued)

DIAL-A-HELPER

Dial-A-Helper is a service provided by the Microcontroller Applications Group. Dial-A-Helper is an electronic bulletin board information system and additionally, provides the capability of remotely accessing the MOLE development system at a customer site.

INFORMATION SYSTEM

The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communications to and from the Microcontroller Applications Group and a FILE SECTION which consists of several file areas where valuable application software and utilities can be found. The minimum requirement for accessing Dial-A-Helper is a Hayes compatible modem.

If the user has a PC with a communications package then files from the FILE SECTION can be down loaded to disk for later use.

Order P/N: MOLE-DIAL-A-HLP

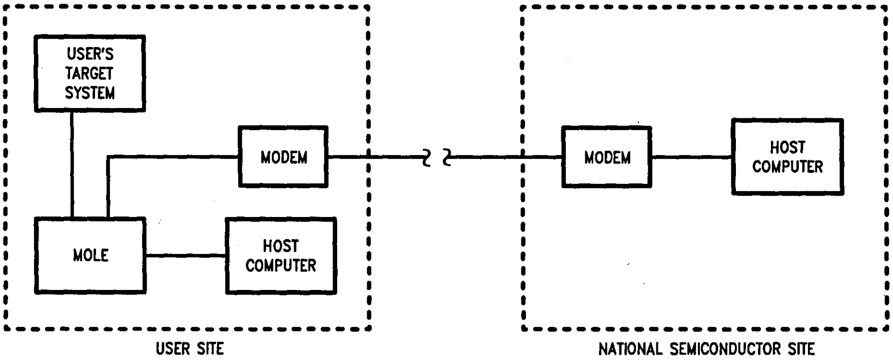
Information System Package Contains:
- Dial-A-Helper Users Manual
- Public Domain Communications Software

FACTORY APPLICATIONS SUPPORT

Dial-A-Helper also provides immediate factory applications support. If a user is having difficulty in operating a MOLE, he can leave messages on our electronic bulletin board, which we will respond to, or under extraordinary circumstances he can arrange for us to actually take control of his system via modem for debugging purposes.

Voice:	(408) 721-5582	
Modem:	(408) 739-1162	
	Baud:	300 or 1200 baud
	Set-Up:	Length: 8-Bit
		Parity: None
		Stop Bit: 1
	Operation:	24 Hrs, 7 Days

DIAL-A-HELPER



TL/DD/8802-20

National Semiconductor

ADVANCE INFORMATION

HPC16400E/HPC36400E/HPC46400E High-Performance Communications microController

General Description

The HPC16400E is an upgraded HPC16400. Features have been added to support V.120 and the UART has been changed to provide more flexibility and power. The HPC16400E is fully upward compatible with the HPC16400.

The HPC16400E has 4 functional blocks to support a wide range of communication application-2 HDLC channels, 4 channel DMA controller to facilitate data flow for the HDLC channels, programmable serial interface and UART.

The serial interface decoder allows the 2 HDLC channels to be used with devices using interchip serial link for point-to-point and multipoint data exchanges. The decoder generates enable signals for the HDLC channels allowing multiplexed D and B channel data to be accessed.

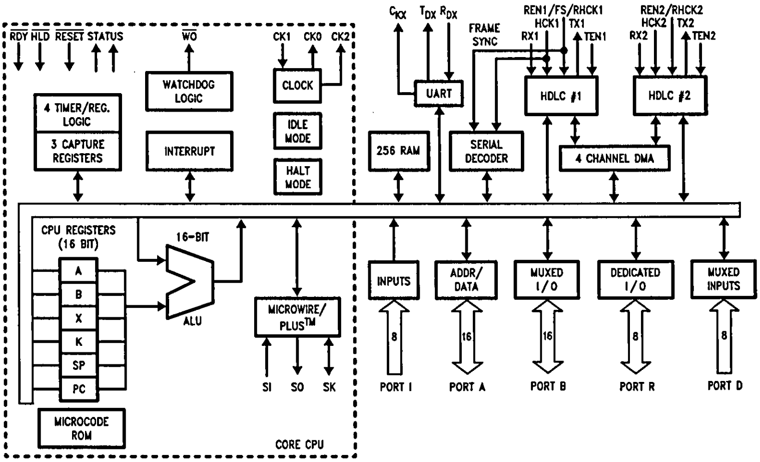
The HDLC channels manage the link by providing sequencing using the HDLC framing along with error control based upon a cyclic redundancy check (CRC). Multiple address recognition modes, and both bit and byte modes of operation are supported.

The HPC16400E is available in 68-pin PLCC, LCC, LDCC and 84-Pin TapePak® packages.

Features

- HPC™ family—core features:
  - —16-bit data bus, ALU, and registers
  - —64 kbytes of external memory addressing
  - —FAST!—20.0 MHz system clock
  - —Four 16-bit timer/counters with WATCHDOG logic
  - —MICROWIRE/PLUS™ serial I/O interface
  - —CMOS—low power with two power save modes
- Two full duplex HDLC channels
  - —Optimized for X.25, V.120, and LAPD applications
  - —Programmable frame address recognition
  - —Up to 4.65 Mbps serial data rate
  - —Built in diagnostics
  - —Synchronous bypass mode
  - —*Optional CRC generation*
  - —*Received CRC bytes can be read by the CPU*
- Four channel DMA controller
- *8 or 16-bit external data bus*
- *UART*
  - —*Full duplex*
  - —*7, 8, or 9 data bits*
  - —*Even, odd, mark, space or no parity*
  - —*7/8, 1 or 2 stop bit generation*
  - —*Accurate baud rate generation up to 625k baud without penalty of using expensive crystal*
  - —*Synchronous and asynchronous modes of operation*
- *Serial Decoder*
  - —*Supports 6 popular time division multiplexing protocols for inter-chip communications*
  - —*Optional rate adaptation of 64 kbit/s data rate to 56 kbit/s*
- 544 kbytes of extended addressing
- Easy interface to National's DASL, 'U' and 'S' transceivers—TP3400, TP3410 and TP3420
- Commercial (0°C to +70°C), industrial (−40°C to +85°C) and military (−55°C to +125°C) temperature ranges

Block Diagram



TL/DD/10422-1

Section 5

HPC Applications

Section 5 Contents

HPC MICROWIRE/PLUS™ Master-Slave Handshaking Protocol

National Semiconductor
Application Note 474
Richard Lazovick

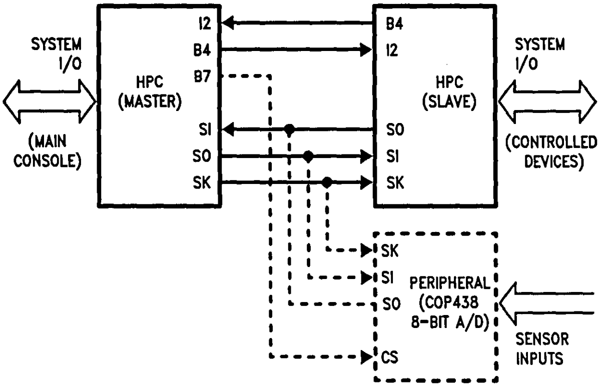
INTRODUCTION

This applications note describes how to use National Semiconductor's MICROWIRE/PLUS to communicate between two members of the HPC family of microcontrollers, and will discuss the implications of adding other MICROWIRE™ peripherals. MICROWIRE/PLUS (μWIRE) may be effectively used to communicate between chips, such as in Small Area Networks (SANs). Possible applications range from setting up a communications network within an automobile to home security systems. Among the standard MICROWIRE peripherals available are display drivers (LCD, VF, LED), memories (RAM, EEPROM), A/D converters, and frequency generators/timers. Each MICROWIRE peripheral requires its own handshaking protocol, however the HPC's MICROWIRE is flexible enough to work with any peripheral and allows you to define your own handshaking protocol when having two HPC family members communicate.

MICROWIRE

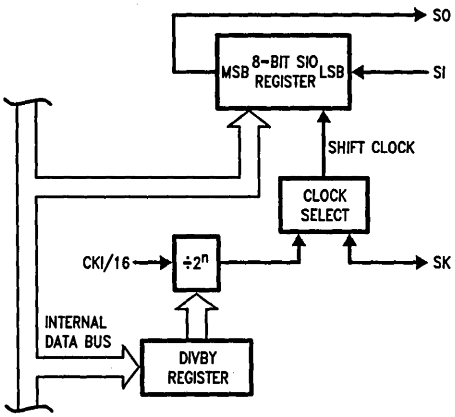
MICROWIRE/PLUS is an extension of National Semiconductor's MICROWIRE communications interface. It allows high speed two way serial communications between a master processor and one or more slave processors or peripherals. MICROWIRE/PLUS uses only three wires plus chip selects, therefore it saves on intricate bus routing and does not waste 8-bit ports. *Figure 1* shows the block diagram of a sample application using two HPC family members and an 8-bit A/D peripheral to monitor and control certain environmental conditions within a system.

MICROWIRE/PLUS has an 8-bit parallel-loaded, serial shift register (SIO) using SI as the serial input and SO as the serial output. The contents of the SIO register may be accessed through any of the memory access instructions. SK is the clock for the SIO register (see *Figure 2*). The SK clock signal can be provided by an internal or external source. The internal clock rate is programmable by the DIVBY register. Data to be transmitted from the SIO register is shifted out on the falling edge of the SK clock. Serial data on the SI pin is latched in on the rising edge of the SK clock (see *Figure 3* μWIRE Timing).



TL/DD/9140-1

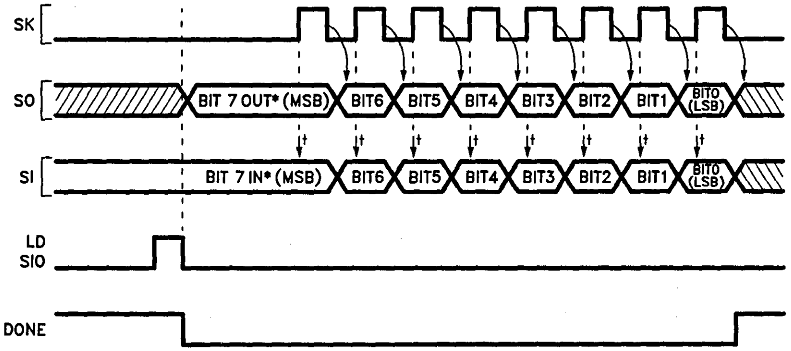
FIGURE 1. HPC μWIRE Block Diagram (Environmental Control System)



TL/DD/9140-2

Note: The most significant bit is shifted out first. The SO pin reflects the contents of the MSB in the SIO register.

FIGURE 2. MICROWIRE/PLUS Block Diagram



TL/DD/9140-3

Note: The first bit of every eight bits in the SIO register being shifted out will have a longer duration then the other bits. This results from the hardware implementation used for MICROWIRE.

* This bit becomes valid immediately when the transmitting device loads its SIO register.

† Arrows indicate points at which SI is sampled.

FIGURE 3. μWIRE Timing

A μWDONE flag in the IRPD (Interrupt Pending) register indicates when the data shift is completed.

The HPC can enter the MICROWIRE/PLUS mode as a master or a slave. The μWMODE control bit in the IRCD (Interrupt Condition) register determines whether the HPC is a master or slave. The shift clock is generated internally when the HPC is configured as a master. An externally generated shift clock on the SK pin is used when the HPC is configured as a slave. When the HPC is a master, the DIVBY register allows the SK clock frequency to be programmed in 14 selectable steps from 122 Hz to 1 MHz when CKI is 16 MHz (see Table I).

HOW TO USE MICROWIRE/PLUS

To use MICROWIRE, start by setting up the B port appropriately for the MICROWIRE functions. The SO and SK functions are multiplexed onto Port B pins B5 and B6 respectively. For the master, set bits 5 and 6 in the DIRB register (direction register for Port B) to set SO and SK as outputs. For the slave, set bit 5 and reset bit 6 in the DIRB register to set SO as an output and SK as an input . The BFUN register (Port B function register) is used to set SO and SK as alternate functions in the master and only SO as an alternate function in the slave. The MICROWIRE/PLUS mode can be enabled or disabled any time under program control. This is done through the BFUN register. Placing a "1" in the corresponding bit location causes the alternate function to be activated, a "0" causes the alternate function to be disabled. It is good practice to initialize the output pins by setting PORTB (Port B data register) to a known state.

The SI function is multiplexed onto Port I pin I5. This pin is always an input and the SI function is automatically selected when in the MICROWIRE mode. Setting the μWMODE control bit, bit 1, in the IRCD register will enable the part to be a master, resetting the bit will make it a slave. For the master, the DIVBY register has to be initialized to set the appropriate SK frequency (see Table I.). For example if the crystal frequency is 16 MHz and an SK frequency of 1 MHz is desired, load the least significant nibble of the DIVBY register with 2 (16 MHz/16 = 1 MHz).

For a summary of the register and pin configurations for the master and slave modes see Table II.

TABLE I. HPC μWIRE DIVBY Register

μWIRE SK Divisor				
MSB			LSB	CLOCK
0	0	0	0	not allowed
0	0	0	1	not recommended*
0	0	1	0	CKI/16
0	0	1	1	CKI/32
0	1	0	0	CKI/64
0	1	0	1	CKI/128
0	1	1	0	CKI/256
0	1	1	1	CKI/512
1	0	0	0	CKI/1024
1	0	0	1	CKI/2048
1	0	1	0	CKI/4096
1	0	1	1	CKI/8192
1	1	0	0	CKI/16384
1	1	0	1	CKI/32768
1	1	1	0	CKI/65536
1	1	1	1	CKI/131072

*This option uses timer T3 output, but does not generate a square wave. (See HPC users manual for more details.)

TABLE II. μWIRE Register and Pin Conditions for Master and Slave Operation

Operation	μWMODE bit	BFUN B5	BFUN B6	DIRB B5	DIRB B6	PIN B5	PIN B6	PIN I5
MICROWIRE Master	1	1	1	1	1	SO	INT. SK	SI
MICROWIRE Master	1	1	1	0	1	TRI-STATE®	INT. SK	SI
MICROWIRE Slave	0	1	0	1	0	SO	EXT. SK	SI
MICROWIRE Slave	0	1	0	0	0	TRI-STATE	EXT. SK	SI

DEFINING THE MASTER/SLAVE HANDSHAKING PROTOCOL

There are a few things to keep in mind when defining a handshaking protocol for the HPC:

1) Only the master can generate SK clocks.
2) As 8 bits are shifted into the SIO register, the 8 bits already in there are shifted out.
3) After 8 bits are shifted into (or out of) the SIO register the MICROWIRE done (μWIRE DONE) flag gets set.
4) ANY access to the SIO register in the master that performs a write operation causes the contents of SIO to be shifted out.
5) No data will be shifted into or out of the slave's SIO register if its μWIRE DONE flag is set.
6) Any write to the SIO register in the master or slave resets its μWIRE DONE flag.

Keeping the above six points in mind, let's look at one possible handshaking protocol between a master HPC and a slave HPC. Number two above tells us we can send and receive data at the same time, however since only the master initiates data transfer we want to be sure the slave is ready before we get started with the exchange. Since the master initiates the transfer process there is no need for the master's MICROWIRE routine to be interrupt driven (though it can be if it is desired to have the slave initiate data transfers also). On the other hand, since the slave will be off doing other tasks it is most effective to have its MICROWIRE routine be interrupt driven.

A FEW THINGS TO NOTE ABOUT THE PROGRAMS

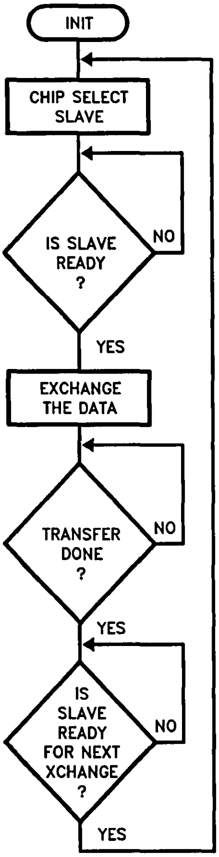
The following programs refer to the system configuration shown in *Figure 1.* This example code does a simple data transfer. The master reads in data on Port D, sends it via MICROWIRE to the slave, and reads it back. They both start by initializing the chip mode and number of wait states (PSW), disabling interrupts, setting the DIVBY register as necessary, initializing Port B, and enabling the appropriate MICROWIRE mode (IRCD). Then the slave continues with its main code (a wait loop) until interrupted. When the master decides it's ready to send MICROWIRE data, it signals the slave by setting the slave interrupt pin on Port B, then it waits for the slave to respond.

Meanwhile, the slave goes into action. It clears the μWDONE flag and loads the SIO register (X A, SIO), then notifies the master that it is ready to continue. Once the master realizes the slave is ready to continue, it removes the interrupt signal to the slave (RESET PORTB.SLAVI), reads in the data to be sent (LD A, PORTD), and starts transmitting it (X A, SIO). At the same time the master reads in the data received at the last data exchange with the slave. Then the master loops until it is done transferring data and loops again until the slave is finished with its interrupt routine. In a real program the master would be off executing code and not having to wait in these loops. Once the transmission is complete the slave reads in the new data (LD A, SIO), lets the master know it is done with its interrupt routine (RESET PORTB.MASTR), and re-enables interrupts as it returns to the main routine (RETI).

In the master's code there is only one access to the SIO register and that access is an exchange. Remember point #4, we can take advantage of the exchange instruction (X A, SIO), which is a read-modify-write instruction. Therefore, with one instruction, we can read the data from the previous transfer into the accumulator, and write the data to be transferred into the SIO register. If this method is not practical, then separate read and write instructions must be used.

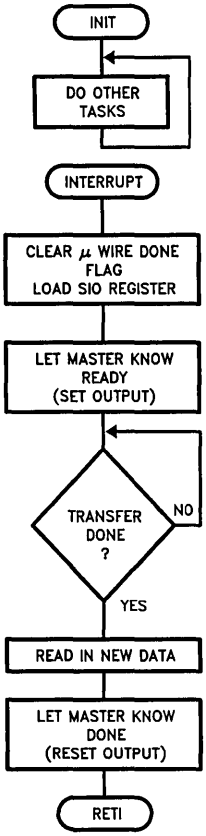
When accessing the SIO register be sure the μWIRE DONE flag is set so you know the data is not changing. At other times we have to be sure the flag is reset or no data will ever be transferred (shifted in or out). Notice that the "X A, SIO" was used to reset the μWIRE DONE flag as well as load the register with the data to be sent.

MASTER'S Flow Chart



TL/DD/9140–4

SLAVE'S Flow Chart



TL/DD/9140–5

MASTER's SAMPLE CODE

```
;
;VARIABLE DECLARE
;
PSW    = M(00C0)

BFUN   = W(0F4)           ;Port B ALTERNATE FUNCTION REGISTER
DIRB   = W(0F2)           ;Port B DIRECTION REGISTER
PORTB  = W(0E2)           ;Port B DATA REGISTER
PORTD  = M(0104)          ;Port D (INPUT PORT)

ENIR   = M(0D0)           ;INTERRUPT ENABLE REGISTER
IRPD   = M(0D2)           ;INTERRUPT PENDING REGISTER
IRCD   = M(0D4)           ;INTERRUPT CONDITION REGISTER
SIO    = M(0D6)           ;SERIAL I/O REGISTER
PORTI  = M(0D8)           ;INTERRUPT (AND uWIRE SERIAL IN) INPUT PORT
DIVBY  = W(018E)          ;TIMER DIVIDE BY REGISTER

SLAVI  = 4                ;SLAVE INTERRUPT BIT (IN Port B)
uWDONE = 0                ;uWIRE DONE BIT (IN IRPD)
uWMODE = 1                ;uWIRE MASTER/SLAVE BIT (IN IRCD)
SK     = 6                ;uWIRE SERIAL CLOCK (IN Port B)
SLAVR  = 2                ;SLAVE RESPONSE BIT (IN Port B)
```

MASTER's SAMPLE CODE (Continued)

```
.=0F800                  ;START PROGRAM

BEGIN:
        LD    PSW,008             ;SINGLE CHIP MODE, 1 WAIT STATE
        LD    ENIR,00             ;DISABLE ALL INTERRUPTS
        LD    DIVBY,02222         ;uWIRE CLOCK = /16

        LD    DIRB,OFFFF          ;Port B ALL OUTPUTS
        LD    BFUN,00060          ;ONLY SO & SK HAVE ALTERNATE FUNCTIONS
        LD    PORTB,00000         ;INIT PORTB TO ALL ZEROs

        SET   IRCD.uWMODE         ;SET THIS HPC AS MASTER

DOITAG:                       ;JUMP TO HERE TO DO IT AGAIN
        SET   PORTB.SLAVI         ;NOTIFY SLAVE (INTERRUPT THE SLAVE)

WAIT:
        IF    PORTI.SLAVR         ;SLAVE READY?
        JP    SLAVRS              ;GO SEND/RECEIVE uWIRE DATA
        JP    WAIT                ;NO IT IS NOT READY YET

SLAVRS:
        RESET PORTB.SLAVI         ;REMOVE SLAVE NOTIFIER
        LD    A,PORTD             ;LOAD A W/ DATA TO SEND
        X     A,SIO               ;SEND NEW DATA AND READ DATA FROM
                                  ;...LAST uWIRE EXCHANGE

DONE:
        IF    IRPD.uWDONE         ;WAIT TILL DONE EXCHANGING
        JP    CONT                ;uWIRE IS DONE
        JP    DONE                ;uWIRE NOT DONE (KEEP TESTING)
CONT:
        IF    PORTI.SLAVR         ;IS SLAVE READY TO CONTINUE?
        JP    CONT                ;NO
        JP    DOITAG              ;START ALL OVER (DO IT AGAIN)

.END BEGIN
```

SLAVE's SAMPLE CODE

```
;
;VARIABLE DECLARE
;
PSW    = M(00C0)

BFUN   = W(0F4)           ;Port B ALTERNATE FUNCTION REGISTER
DIRB   = W(0F2)           ;Port B DIRECTION REGISTER
PORTB  = W(0E2)           ;Port B DATA REGISTER
```

SLAVE's SAMPLE CODE (Continued)

```
ENIR   = M(0D0)                ;INTERRUPT ENABLE REGISTER
IRPD   = M(0D2)                ;INTERRUPT PENDING REGISTER
IRCD   = M(0D4)                ;INTERRUPT CONDITION REGISTER
SIO    = M(0D6)                ;SERIAL I/O REGISTER
SO     = 5                     ;uWIRE SERIAL OUTPUT PIN (ON Port B)
MASTR  = 4                     ;MASTER RESPONSE BIT (IN Port B)
uWDONE = 0                     ;uWIRE DONE BIT (IN IRPD)
uWMODE = 1                     ;uWIRE MASTER/SLAVE BIT (IN IRCD)
INT2   = 2                     ;INTERRUPT 2 BIT

.=0FFFA                        ;INT2 - INTERRUPT VECTOR
.WORD MASNOT                   ;...MASTER NOTIFICATION
.=0F800                   ;START PROGRAM

BEGIN:
       LD     PSW,008               ;SINGLE CHIP MODE, 1 WAIT STATE
       LD     ENIR,01               ;DISABLE ALL INTERRUPTs, BUT ENABLE GIE

       LD     DIRB,0FF10            ;Port B UPPER, & MASTR ARE OUTPUTS
                                    ;...(use LD DIRB,0FF30 to set SO as an
                                    ;...output if not using any peripherals)
       LD     BFUN,00020            ;ONLY SO HAS ALTERNATE FUNCTION
                                    ;...NOTE: SK is NOT an alternate
                                    ;...function in the slave!
       LD     PORT B,00000          ;INIT PORTB TO ALL ZEROS

       RESET  IRCD.uWMODE           ;SET THIS HPC AS A SLAVE
       SET    IRCD.INT2             ;SET INT2 INTERRUPT (+) POLARITY
       SET    ENIR.INT2             ;ENABLE EXTERNAL INTERRUPT TO
                                    ;...RECEIVE SLAVE RESPONSE

PAU:
       JP     PAU                   ;WAIT HERE FOR INTERRUPT FROM MASTER

MASNOT:                   ;uWIRE INTERRUPT ROUTINE

       X      A,SIO                 ;CLEAR uWDONE FLAG (AND LOAD DATA FROM
                                    ;...ACCUMULATOR TO SEND)
       SET    PORTB.SO              ;ENABLE SO (needed only if using a peripheral)
       SET    PORTB.MASTR           ;NOTIFY MASTER THAT READY TO CONTINUE

NOTDN:
       IF     IRPD.uWDONE           ;WAIT TILL DONE SHIFTING
       JP     DONE                  ;DONE, GO CONTINUE
       JP     NOTDN                 ;NOT DONE, CONTINUE LOOPING

DONE:
       LD     A,SIO                 ;READ IN NEW DATA

       RESET  PORT B.SO             ;TRI-STATE SO (needed only if
                                    ;               using a peripheral)

       RESET  PORTB.MASTR           ;REMOVE SIGNAL TO MASTER
       RETI

.END BEGIN
```

ADDING PERIPHERALS OR ANOTHER SLAVE

Adding another slave HPC or a peripheral to the above Microwire configuration can add more power to your design with minimal extra cost and design time. In *Figure 1,* an extra peripheral is shown in dotted outline form. The hardware and software modifications are straightforward, however there are a few considerations to keep in mind:

— Tri-state the SO pin on the slave HPC by resetting B5 in the DIRB register when the slave is not 'chip-selected' by the master.

— When adding more HPC slaves, the master's and slave's routines remain the same. Only different B port pins for chip select and I or B port pins for slave acknowledge need to be used.

— For peripherals the principals of operation are still the same and so are the initialization procedures, however some of the code will have to be modified to accommodate the specific handshaking required by the peripheral. (Note: some of the peripherals require 16 or more consecutive bits without interruption of the SK clock. To provide continuous SK clocks, set up the accumulator with next byte of data to send, loop until μWDONE is set, then exchange the contents of the accumulator and the SIO register (X A, SIO). The above steps will provide nearly continuous SK clocks—the slower the SK clock is set for, the more continuous they will appear.)

APPLICATIONS

Now that you are more familiar with MICROWIRE/PLUS, where can you get experience using it?

— It can be used in a security system where the on-site master lets the periphery slaves know which security codes they can now let in, while at the same time the slaves monitor fire alarms and smoke detectors.

— It can be used in automotive brakes to allow all the wheels to communicate with each other. The wheels can trade information on road conditions and a master can monitor all four wheels to coordinate them and check for malfunctions.

— It can be used in a robot arm to allow each joint to make the decision as to how it will help the entire arm reach its final position. This application is one example of how MICROWIRE/PLUS can be used for system task partitioning.

— It can be used in a MUX-WIRING system.

When using MICROWIRE to communicate between two chips on the same board, a high data rate can be used. When communicating over longer distances, slower speeds should be used.

SUMMARY

MICROWIRE/PLUS can be a very powerful tool that can easily add power to a microcontroller based system. It is easy to use and does not require much hardware to implement. To add a new feature to your current design, choose a peripheral and add a small amount of code. To start using MICROWIRE, define the handshake protocol best suited for your application keeping in mind the six points given above in the 'Defining the Master/Slave Handshaking Protocol' section. Then initialize the appropriate registers: BFUN, DIRB, PORTB, DIVBY, and IRCD. The MICROWIRE circuitry will then run independent of the CPU except to exchange data between the SIO register and the CPU, and to initiate the data exchange between the master and slaves. With a CPU clock of 16 MHz, MICROWIRE/PLUS may achieve a maximum data rate of 1 MHz. MICROWIRE can be used to add display controllers, A/D's, memories, timers, and even other microcontrollers to an HPC microcontroller based design. Remember MICROWIRE/PLUS is not a trivial piece of very fine wire, it is a high speed two way serial communications interface!

Interfacing Analog Audio Bandwidth Signals to the HPC

National Semiconductor
Application Note 484
Ashok Krishnamurthy

INTRODUCTION

This report describes a method of interfacing analog audio bandwidth signals to the National Semiconductor HPC microcontroller. The analog signal is converted to a digital value using the National Semiconductor TP3054 codec/filter combo. The digital value is then transferred to the HPC using the MICROWIRE/PLUS™ synchronous serial interface. The digital output sample computed by the HPC is also transferred to the TP3054 using the MICROWIRE/PLUS interface. The TP3054 then converts this digital value to an analog signal.

ADVANTAGES OF USING A CODEC

There are a number of advantages in using a codec for A/D and D/A conversion of analog signals.

1. The codec/filter combos such as the TP3054 integrate a number of functions on a single chip. Thus the TP3054 includes the analog anti-aliasing filters, the Sample-and-Hold circuitry and the A/D and D/A converters for analog signal interfacing.
2. The μ-law coding effectively codes a 14-bit conversion accuracy in 8 bits. This allows the interface to the HPC to be greatly simplified.

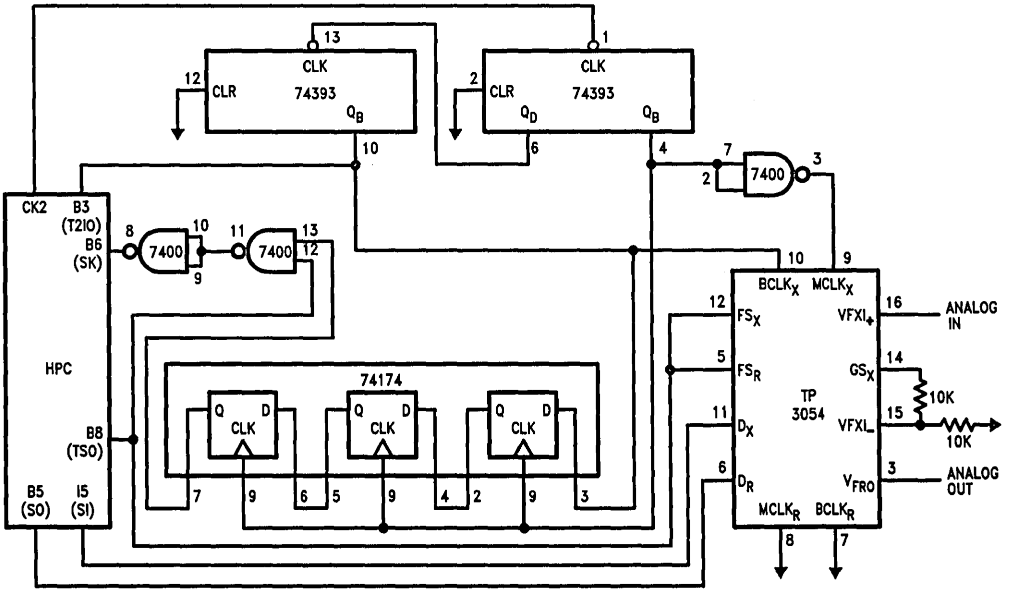
DISADVANTAGES IN USING A CODEC

While the use of a codec is appropriate for audio (in particular speech) processing applications, it has a number of disadvantages in other cases.

1. The sampling rate is fixed at 8 kHz. If lower or higher sampling rates are desired, the codec cannot be used. Note that the real-time signal processing that the HPC can perform at a 8 kHz sampling rate is limited.
2. The resolution is fixed, and is about 14 bits/sample.
3. Digital filtering algorithms require that the samples used in the processing be linear coded PCM. Thus the 8-bit μ-law PCM values output by the codec need to be converted to linear coded PCM. Correspondingly, the output of the digital filter, which is in linear coded PCM needs to be converted to 8-bit μ-law PCM before outputting to the codec. This requires additional processing per sample.

DESCRIPTION OF THE INTERFACE

The circuit schematic of the interface is shown in *Figure 1.* Note that the schematic does not show complete details of the HPC. Only the HPC pins that are relevant to this interface are shown. A wire-wrapped version of the circuit has been constructed on a NSC HPC 16040 Chip Carrier Board.



TL/DD/9248-1

FIGURE 1. Circuit Schematic

Note that this report does not go into the details about the use of the TP3054 codec chip or programming the HPC. It also does not discuss the μ-law to linear and linear to μ-law code conversion in detail. For more information on these issues, please consult the references listed at the end.

1. **Codec Signalling Considerations.** The TP3054 can operate in either synchronous or asynchronous modes. Further, in each of these modes, it uses short or long frame sync operation. The circuit shown in *Figure 1* runs the codec in synchronous mode with long-frame-sync operation.

The codec requires 4 clock sources for proper operation in the synchronous mode. These are MCLK-x, BCLK-x, FS-x and FS-r. MCLK-x is a master clock and is used to clock the switched-capacitor filters. BCLK-x is the bit shift clock, and FS-x and FS-r are the frame sync clocks. These clocks need to be synchronous.

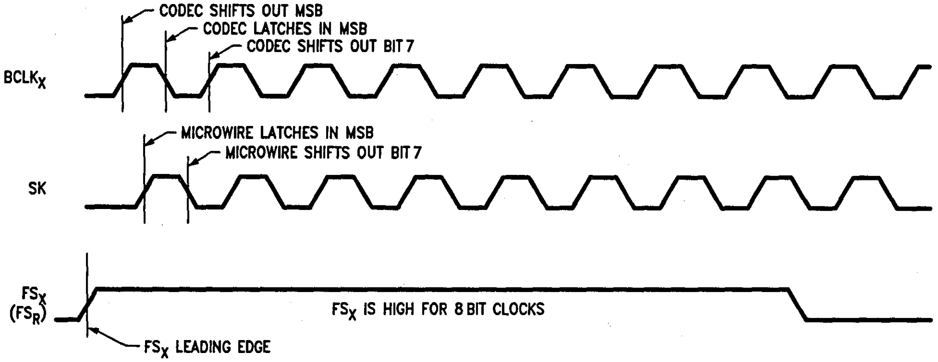
These clocks are obtained in the circuit as follows. MCLK-x is obtained by dividing the HPC CK2 clock output by 4. If the HPC is using a 16 MHz crystal, this results in MCKL-x being 2 MHz.

BCLK-x is obtained by dividing CK2 by 64. This gives an effective value for BCLK-x of 125 kHz. Note that MCLK-x is inverted before being fed to the codec. This is done to synchronize MCLK-x and BCLK-x on their leading edges.

FS-x and FS-r are the same clocks in the circuit. They are obtained by dividing BCLK-x by 16 using the timer T2 on the HPC. BCLK-x is used as the external clock input on pin T2IO of the HPC and FS-x (FS-r) is obtained from the timer synchronous output TS0. Note that the delay inherent in the HPC between the underflow of a timer and the toggling of the corresponding output allows FS-x and BCLK-x to be leading edge synchronized (more accurately, the delay is within the codec's acceptable limits.) Note that in order to accomplish these functions, the HPC pins need to be properly configured. This is not described here. Please refer to the appropriate HPC documentation and consult the sample program included with this report.

2. **MICROWIRE/PLUS Interface Considerations.** MICROWIRE/PLUS is a National Semiconductor defined 8-bit serial synchronous communication interface. It is designed to allow easy interfacing of NSC microcontrollers and peripheral chips. The HPC microcontroller has a MICROWIRE/PLUS interface; however the TP3054 codec does not. Thus some external "glue logic" is necessary to allow the HPC and the TP3054 to be interfaced.

The HPC MICROWIRE/PLUS interface is operated in Slave mode for this application. This means that the shift clock needed to latch-in/shift-out data from the Micro-wire SIO register is provided externally on the SK pin. Micro-wire latches in data on the leading edge of the SK clock and shifts out data on the trailing edge of SK. Also SK needs to be a burst clock for proper operation.



TL/DD/9246-2

FIGURE 2. Timing Waveforms

The codec shifts out data on the D-x pin on the first 8 leading edges of BCLK-x after a FS-x leading edge. Also, it latches in data on the D-r pin on the first 8 trailing edges of BCLK-x after a FS-r leading edge. Note that FS-x and FS-r are the same in this application. Refer to the timing diagram in *Figure 2*.

Thus, it is seen that there is a timing difference in the way the codec and the Micro-wire interfaces work. However, as seen in *Figure 2*, if the shift clock, SK, to the Microwire interface is delayed with respect to BCLK-x, the two interfaces should work compatibly. This delay is accomplished by clocking BCLK-x through a shift register using MCLK-x as the clock source. This can be seen in the circuit schematic in *Figure 1*. (The author thanks Mr. Richard Lazovick for this suggestion.)

μ-LAW TO LINEAR/LINEAR TO μ-LAW CONVERSION

It was explained earlier that the codec outputs digital values that are companded using the MU-255 PCM standard. However, for linear digital filtering applications, the input needs to be in linear PCM format. Similarly, it is necessary to provide the conversion from linear PCM to MU-255 PCM before output to the codec. The HPC accomplishes this in software.

1. μ-law to linear conversion. The codec output is actually the complement of the μ-law value. Thus, this first needs to be complemented to obtain the true μ-law value. The simplest way to obtain the corresponding linear value is through table look-up. The output of the table is the 16-bit 2's complement linear value. The sample program included with this report utilizes this technique. A macro that constructs this table is also provided.
2. Linear to μ-law conversion. An algorithm to convert a 13-bit positive linear number to 7-bit μ-law is described in *Figure 3*. The algorithm is based on the description in the book by Bellamy listed in the reference. The most significant 8th bit for the μ-law code is obtained from the sign of the input linear code.

1. Get 13-bit positive input value.
2. Add to it the bias value of 31-decimal.
3. The compressed μ-law word is then obtained as follows:

Biased Linear Value

Bits

12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	Q3	Q2	Q1	Q0	a
0	0	0	0	0	0	1	Q3	Q2	Q1	Q0	a	b
0	0	0	0	0	1	Q3	Q2	Q1	Q0	a	b	c
0	0	0	0	1	Q3	Q2	Q1	Q0	a	b	c	d
0	0	0	1	Q3	Q2	Q1	Q0	a	b	c	d	e
0	0	1	Q3	Q2	Q1	Q0	a	b	c	d	e	f
0	1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g
1	Q3	Q2	Q1	Q0	a	b	c	d	e	f	g	h

μ-Law Value

Bits

6	5	4	3	2	1	0
0	0	0	Q3	Q2	Q1	Q0
0	0	1	Q3	Q2	Q1	Q0
0	1	0	Q3	Q2	Q1	Q0
0	1	1	Q3	Q2	Q1	Q0
1	0	0	Q3	Q2	Q1	Q0
1	0	1	Q3	Q2	Q1	Q0
1	1	0	Q3	Q2	Q1	Q0
1	1	1	Q3	Q2	Q1	Q0

FIGURE 3. 13-Bit Linear to 8-Bit μ-Law Conversion

POSSIBLE APPLICATIONS

The codec/HPC interface described above can be used in a number of speech processing applications. One application, ADPCM coding of speech, is presently under development. Other applications include: a voiced/unvoiced/silence classifier, a voice pitch tracker, speech detection circuitry etc. Note that the main limitation here (at least for real-time applications) is the amount of effective computation that can be done by the HPC between samples.

REFERENCES

1. National Semiconductor Corporation, *Telecommunications Databook,* Santa Clara, California, 1984.
2. National Semiconductor Corporation, *HPC Programmers Reference Manual,* Santa Clara, California, 1986.
3. National Semiconductor Corporation, *HPC Hardware Reference Manual,* Santa Clara, California, 1986.
4. J. C. Bellamy, *Digital Telephony,* John Wiley & Sons, New York, 1982.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communication package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

PROGRAM TO TEST CODEC INTERFACE

```
NATIONAL SEMICONDUCTOR CORPORATION   Page:  1
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
TSTCDC

  1             ;
  2             ;
  3                             .TITLE TSTCDC
  4             ;
  5 01C0                        YOFK = M(01C0)        ; OUTPUT SAMPLE STORAGE.
  6 00C0                        PSW = M(00C0)
  7 00D0                        ENIR = M(00D0)
  8 00D2                        IRPD = M(00D2)
  9 00D4                        IRCD = M(00D4)
 10 00D6                        SIO = M(00D6)
 11 00D8                        PORTI = M(00D8)
 12 00E2                        PORTBL = M(00E2)
 13 00E3                        PORTBH = M(00E3)
 14 00E2                        PORTB = W(00E2)
 15 00F2                        DIRBL = M(00F2)
 16 00F3                        DIRBH = M(00F3)
 17 00F2                        DIRB = W(00F2)
 18 00F4                        BFUNL = M(00F4)
 19 00F5                        BFUNH = M(00F5)
 20 00F4                        BFUN = W(00F4)
 21 0188                        T2TIM = W(0188)
 22 0186                        T2REG = W(0186)
 23 018E                        DIVBYL = M(018E)
 24 018F                        DIVBYH = M(018F)
 25 018E                        DIVBY = W(018E)
 26 0190                        TMMDL = M(0190)
 27 0191                        TMMDH = M(0191)
 28 0190                        TMMD = W(0190)
 29             ;
 30             ;
 31             ;
 32                             .MACRO MUTBL,STADR
 33             ;
 34             ; MACRO TO CREATE LOOKUP TABLE FOR MU-255 LAW PCM TO LINEAR CONVERSION.
 35             ; STADR IS THE STARTING ADDRESS FOR THE TABLE, AND MUST BE AN EVEN ADDRESS.
 36             ; THE TABLE OCCUPIES 512 BYTES.
 37             ;
 38                             . = STADR
 39                             .SET SVAL,021
 40                             .SET INCRM,02
 41                             .DO 08
 42                                   .SET MVAL,SVAL-021
 43                                   .DO 010
 44                                     .WORD MVAL
 45                                     .SET MVAL,MVAL+INCRM
 46                                   .ENDDO
 47                                   .SET SVAL,SVAL*02
 48                                   .SET INCRM,INCRM*02
 49                             .ENDDO
 50             ;
 51                             .SET SVAL, 021
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  2
HPC CROSS ASSEMBLER, REV:C,30 JUL 86
TSTCDC

 52                             .SET INCRM, 02
 53                             .DO 08
 54                                   .SET MVAL,SVAL-021
 55                                   .DO 010
 56                                     .SET RVAL,-1*MVAL
 57                                     .WORD RVAL
 58                                     .SET MVAL,MVAL+INCRM
 59                                   .ENDDO
 60                                   .SET SVAL,SVAL*02
 61                                   .SET INCRM,INCRM*02
 62                             .ENDDO
 63              ;
 64                             .ENDM
 65              ;
 66              ;
 67              ;
 68                             .LOCAL
 69 F000                        MUTBL, 0F000
 70              ;
 71 F200                        .= 0F200
 72              CODEC:
 73 F200 B701F0C4               LD SP, 01F0          ; INITIALIZE STACK POINTER.
 74              ;
 75 F204 3059                   JSR INITCD           ; INITIALIZE THE CODEC
 76              FLOOP:
 77 F206 3005                   JSR INPUT            ; GET INPUT SAMPLE, OUTPUT
 78                                                  ; PREVIOUS SAMPLE.
 79 F208 E7                     SHL A
 80 F209 E7                     SHL A
 81 F20A 301F                   JSR OUTPUT           ; CONVERT OUTPUT VALUE TO
 82                                                  ; MU-255 LAW AND SAVE.
 83 F20C 66                     JP FLOOP             ; 60 DO NEXT SAMPLE.
 84              ;
 85              ;
 86              INPUT:
 87 F20D B601C088               LD A, YOFK           ; GET DATA TO BE OUTPUT.
 88              NOTDN:
 89 F211 96D210                 IF IRPD,0            ; IS MICROWIRE DONE?
 90 F214 41                     JP MWDONE            ; YES, SO GET DATA.
 91 F215 64                     JP NOTDN             ; NO, SO TRY AGAIN.
 92              MWDONE:
 93 F216 BED6                   X A, SIO             ; GET NEW SAMPLE, OUTPUT
 94                                                  ; COMPUTED DATA.
 95 F218 01                     COMP A               ; TAKE CARE OF CODEC INVERSION.
 96 F219 99FF                   AND A, OFF
 97 F21B E7                     SHL A
 98 F21C BAF000                 OR A,0F000           ; FORM MU-LAW TO LINEAR
 99                                                  ; TABLE ADDRESS.
100 F21F AECE                   X A, X
101 F221 D0                     LD A, M(X+)          ; GET LINEAR VALUE
102 F222 AECA                   X A, K
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC

103 F224 04                     LD A, M(X)            ; A BYTE AT A TIME.
104 F225 BCC8CB                 LD H(K), L(A)
105 F228 ABCA                   LD A, K
106 F22A 3C                     RET
107              ;
108              ;
109              OUTPUT:
110 F22B 96D41F                 RESET IRCD.7
111 F22E E7                     SHL A                 ; SIGN BIT TO C.
112 F22F 06                     IFN C                 ; IS IT POSITIVE?
113 F230 45                     JP OPOS
114 F231 96D40F                 SET IRCD.7
115 F234 01                     COMP A
116 F235 04                     INC A                 ; NEGATIVE, SO TAKE 2'S
117                                                   ; COMPLEMENT.
118              OPOS:
119 F236 B80108                 ADD A, 0108           ; ADD BIAS.
120 F239 9107                   LD K, 07              ; SET UP COUNTER.
121              ALIGN:                               ; LOOP AND LOCATE MS 1 BIT.
122 F23B E7                     SHL A
123 F23C 07                     IF C
124 F23D 44                     JP ODONE              ; FOUND MS 1 BIT.
125 F23E AACA                   DECSZ K
126 F240 65                     JP ALIGN
127 F241 E7                     SHL A                 ; HAS TO BE 1 IN C NOW.
128              ODONE:
129 F242 AECA                   X A, K
130 F244 E7                     SHL A
131 F245 E7                     SHL A
132 F246 E7                     SHL A
133 F247 E7                     SHL A                 ; COUNTER VALUE IN BITS 4-6.
134 F248 AECC                   X A, B
135 F24A 00                     CLR A
136 F24B 88CB                   LD A, H(K)
137 F24D 3B                     SWAP A
138 F24E 990F                   AND A , OF
139 F250 96CCFA                 OR A, B
140 F253 96D417                 IF IRCD.7
141 F256 96C80F                 SET A.7
142 F259 01                     COMP A
143 F25A B601C08B               ST A, YOFK
144 F25E 3C                     RET
145              ;
146              INITCD:
147 F25F B7FFB7F2               LD DIRB, OFFB7        ; SET B3 (T2IO) AND B6 (SK)
148                                                   ; ON PORT B AS INPUTS. SET ALL
149                                                   ; OTHER PINS ON B AS OUTPUT.
150 F263 B70000E2               LD PORTB, 0           ; OUTPUT 0 ON ALL PORT B PINS.
151 F267 96F40B                 SET BFUNL.3           ; ALT. FUN. ON B3-T2IO.
152 F26A 96F40D                 SET BFUNL.5           ; ALT. FUN. ON B5-SO.
153 F26D 96F508                 SET BFUNH.0           ; ALT. FUN. ON B8-TSO.
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC

154 F270 9700D0                LD ENIR, 0            ; DISABLE INTRPTS.
155 F273 9700D4                LD IRCD,0             ; SELECT SLAVE MODE FOR M-WIRE.
156 F276 83070188AB            LD T2TIM, 07          ; LOAD 7-DEC INTO T2 TIMER.
157 F27B 83070186AB            LD T2REG, 07          ; LOAD 7-DEC INTO T2 REG.
158 F280 8300018F8B            LD DIVBYH, 0          ; SELECT EXT, CLOCK FOR T2 TIMER.
159              ;
160 F285 8ED6                  X A, SIO
161 F287 8740400190AB          LD TMMD,04040         ; START TIMER T2.
162 F28D 3C                    RET
163              ;
164              ;
165 FFFE 00F2                  .END CODEC
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
SYMBOL TABLE

A       00C8 W        ALIGN   F23B          B       00CC W        BFUN    00F4 W*
BFUNH   00F5 M        BFUNL   00F4 M        CODEC   F200          DIRB    00F2 W
DIRBH   00F3 M*       DIRBL   00F2 M*       DIVBY   018E W*       DIVBYH  018F M
DIVBYL  018E M*       ENIR    00D0 M        FLOOP   F206          INCRM   0200
INITCD  F25F          INPUT   F20D          IRCD    00D4 M        IRPD    00D2 M
K       00CA W        MVAL    205F          MWDONE  F216          NOTDN   F211
ODONE   F242          OPOS    F236          OUTPUT  F22B          PC      00C6 W
PORTB   00E2 W        PORTBH  00E3 M*       PORTBL  00E2 M*       PORTI   00D8 M*
PSW     00C0 M*       RVAL    EDA1          SIO     00D6 M        SP      00C4 W
SVAL    2100          T2REG   D186 W        T2TIM   0188 W        TMMD    0190 W
TMMDH   0191 M*       TMMDL   0190 M*       X       00CE W        YOFK    01C0 M
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
MACRO TABLE

MUTBL

  NO WARNING LINES

  NO ERROR LINES

  656 ROM BYTES USED

SOURCE CHECKSUM = 81D3
OBJECT CHECKSUM = 0C3C

INPUT   FILE C:CODECTST.MAC
LISTING FILE C:CODECTST.PRN
OBJECT  FILE C:CODECTST.LM
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
TSTCDC
SYMBOL TABLE

A       00C8 W        ALIGN   F23B          B       00CC W        BFUN    00F4 W*
BFUNH   00F4 M        BFUNL   00F4 M        CODEC   F200          DIRB    00F2 W
DIRBH   00F3 M*       DIRBL   00F2 M*       DIVBY   018E W*       DIVBYH  018F M
DIVBYL  01B3 M*       ENIR    00D0 M        FLOOP   F206          INCRM   0200
INITCD  F25F          INPUT   F20D          IRCD    00D4 M        IRPD    00D2 M
K       00CA W        MVAL    205F          MWDONE  F216          NOTDN   F211
ODONE   F242          OPOS    F236          OUTPUT  F22B          PC      00C6 W
PORTB   00E2 W        PORTBH  00E3 M*       PORTBL  00E2 M*       PORTI   00D8 M*
PSW     00C0 M*       RVAL    EDA1          SIO     00D6 M        SP      00C4 W
SVAL    2100          T2REG   D186 W        T2TIM   0188 W        TMMD    0190 W
TMMDH   0191 M*       TMMDL   0190 M*       X       00CE W        YOFK    01C0 M
```

Digital Filtering Using the HPC

National Semiconductor
Application Note 485
Ashok Krishnamurthy

INTRODUCTION

This report discusses the implementation of Infinite Impulse Response (IIR) digital filters using the National Semiconductor HPC microcontroller. A general program, that can be used to implement cascaded second order sections, up to a maximum of 8 sections, is also included. The program may have to be modified for specific A/D and D/A interfaces.

This report is not intended to be a tutorial on Digital Filter Design methods or their implementation details. Such information can be found in references 1 and 2 below. The general discussion included here closely follows that in reference 3.

DIGITAL FILTERING

The general IIR filter with input x (n) and output y (n) can be described by a transfer function of the form

$$H(z) = \frac{Y(z)}{X(z)} = \frac{a(0) + a(1)\,z^{-1} + \ldots + a(m)\,z^{-m}}{1 + b(1)\,z^{-1} + \ldots + b(p)\,z^{-p}}$$

To minimize the effects of coefficient truncation, high order filters are usually implemented as a cascade of second order sections. (Another possible choice is parallel realization—see references below).

In cascade realizations, the numerator and denominator polynomials in the above are factored into second order terms, and the filter is realized as a cascade of such second order sections. This is shown in *Figure 1*. A typical second order section has a transfer function of the form

$$H(z) = \frac{A0 + A1 \times z^{-1} + A2 \times z^{-2}}{1 + B1 \times z^{-1} + B2 \times z^{-2}}$$

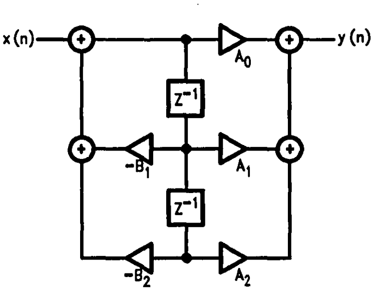
A second order section such as the above can be realized in a number of ways; the one of concern here is the so-called 1-D form (see Reference 3). The second order 1-D form is shown in *Figure 2*. Based on this figure, we can obtain the following equations:

$$m(k) = x(k) - B1 \times m(k-1) - B2 \times m(k-2)$$

$$y(k) = A0 \times m(k) + A1 \times m(k-1) + A2 \times m(k-2)$$

Define $T1 = -B1 \times m(k-1) - B2 \times m(k-2),$

$T2 = A1 \times m(k-1) + A2 \times m(k-2)$



TL/DD/9247-2

FIGURE 2. One Second Order Section

Since T1 and T2 depend on signal values at time k − 1 and k − 2, we can precompute and store these quantities in the time interval from k − 1 to k. Then, when x(k) becomes available at time k, y(k) and m(k) can be quickly computed using

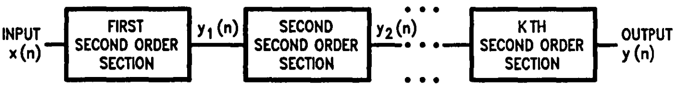
$$m(k) = x(k) + T1,$$

$$y(k) = A0 \times m(k) + T2$$

If there are a number of stages, then these computations should be repeated for each stage. Based on these discussions, the operation of a digital filter can be described using the flowchart in *Figure 3*.

USING THE FILTER PROGRAM

Appendix A contains the listing of the program FILTER that can be used to implement cascaded IIR filters as described above. The program as shown uses a codec interfaced to the HPC using MICROWIRE/PLUS™ to do the A/D and D/A conversion. The program can be used with other A/D and D/A converters by suitably modifying the following subroutines: INPUT, OUTPUT and INIT. Only the portions of INIT that deal with the codec interface need to be modified.



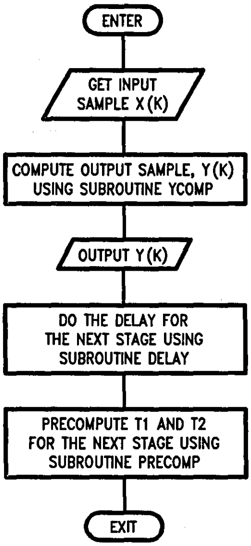
TL/DD/9247-1

FIGURE 1. Cascade Realization of a Digital Filter

The filter coefficients and the number of cascaded stages need to be supplied to the program. This is done as follows:

1. *Specification of filter order.* Define a word address called ROMNST and store the number of cascaded stages in that word. The program is presently set up for 4 cascaded stages.
2. *Specification of filter coefficients.* Each second order stage needs the specification of 5 coefficients, A0, A1, A2, B1 and B2. If the number of stages is m, let the coefficients be

   A0−1, A1−1, A2−1, B1−1, B2−1 for stage 1,
   A0−2, A1−2, A2−2, B1−2, B2−2 for stage 2,
   .
   .
   .
   A0−m, A1−m, A2−m, B1−m, B2−m for stage m.


ENTER
GET INPUT SAMPLE X(K)
COMPUTE OUTPUT SAMPLE, Y(K) USING SUBROUTINE YCOMP
OUTPUT Y(K)
DO THE DELAY FOR THE NEXT STAGE USING SUBROUTINE DELAY
PRECOMPUTE T1 AND T2 FOR THE NEXT STAGE USING SUBROUTINE PRECOMP
EXIT

TL/DD/9247-3

FIGURE 3. Flowchart for the Computations in a Second Order Module (Based on Reference 3)

Define 5 word addresses called ROMA0, ROMA1, ROMA2, ROMB1, ROMB2 and store these coefficients at these addresses as follows:

ROMA0: .WORD A0−1, A0−2, A0−3, ... A0−m
ROMA1: .WORD A1−1, A1−2, A1−3, ... A1−m
ROMA2: .WORD A2−1, A2−2, A2−3, ... A2−m
ROMB1: .WORD B1−1, B1−2, B1−3, ... B1−m
ROMB2: .WORD B2−1, B2−2, B2−3, ... B2−m.

Note that the coefficients are signed and need to be in 2's complement representation. Also, the stored coefficients need to be half their actual value. This is because of the way that the program does 2's complement multiplication using the subroutine SMULT.

The FILTER program copies all the coefficients to on-chip RAM for faster execution. Also temporary storage for m (k), m (k − 1), m (k − 2), T1 and T2 is obtained from on-chip RAM. This, along with the storage of various addresses used by the program consumes the entire 192 bytes of user base page RAM.

Note that the filter program does not check for overflow during the various additions. This is because the HPC does not have a signed addition/subtraction overflow flag, and it was felt that the simulation of this feature in software would add excessive overhead. It is therefore the user's responsibility to ensure that the filter coefficients are properly scaled so that the overflow will not occur.

16 x 16 2's COMPLEMENT MULTIPLICATION

One of the basic operations in digital filtering is that of signed multiplication. Since the HPC supports unsigned multiplication only, a method to perform 2's complement multiply using the unsigned multiply is needed.

Let A and B be 2's complement 16 bit integers. Consider the following cases.

1. $A \geq 0, B \geq 0$. In this case the unsigned multiply result is $A \times B$, which is also the 2's complement multiply result. Thus no further processing is needed.
2. $A \geq 0, B < 0$. In this case the unsigned multiply result is $(2^{16}) \times A - A \times |B|$. However the desired result is $(2^{32}) - A \times |B|$. Thus we need to add $(2^{32}) - (2^{16}) \times A$ to the unsigned multiply result to obtain the correct value.
3. $A < 0, B \geq 0$. This case is similar to the previous one. $(2^{32}) - (2^{16}) \times B$ should be added to the unsigned multiply result to get the correct answer.
4. $A < 0, B < 0$. The unsigned multiply result in this case is $(2^{32}) - (2^{16}) \times (|A| + |B|) + |A| \times |B|$. The desired result in this case is $|A| \times |B|$. To get the correct answer, add $(2^{16}) \times (|A| + |B|)$ to the unsigned multiply result.

Based on the above discussion, an algorithm for 2's complement multiplication, where the result is a 32 bit 2's complement integer is shown in *Figure 4*.

1. Let A and B be the two 2's complement integers to be multiplied.
2. Compute $C = A \times B$, the unsigned product of A and B. Let the upper half of C be C-hi and its lower half C-lo.
3. If A is negative, then add $(2^{16}) - B$ to C-hi. This can be easily done using the SET C, SUBC instructions of the HPC. Let the result be C-hi1.
4. If B is negative, then add $(2^{16}) - A$ to C-hi1. Again it is easily done using the SET C, SUBC instructions. Let the result be C-hi2.
5. The 2's complement product of A and B is C-hi2. C-lo.

FIGURE 4. Algorithm for 2's Complement Multiplication.

MULTIPLICATION BY FILTER COEFFICIENTS

The coefficients that arise in most IIR filter designs are numbers that are usually in the range from $-2 < \text{coeff} < 2$. The coefficients, in most instances can be scaled to be in this range. The action of digital filtering involves successive multiplications. If we want no loss in accuracy due to multiplication, the word length needed to store successive partial products increases rapidly—clearly an impractical choice. Thus the results of the multiplication at the various stages need to be truncated to 16 bits before proceeding to the next stage. The program FILTER does this as follows: The filter state variables are regarded as integers, while the filter coefficients are regarded as fixed point fractions with the binary point to the immediate right of the sign bit. After the multiplication, the result is shifted so that the integer part of the product is in one word, and the fractional part in another. The integer part is then returned as the result of the multiplication, i.e. the product is truncated to 16 bits. This is performed by the subroutine SMULT. Since the filter coefficients are regarded as fixed point fractions, only coefficients in the range $-1 < \text{coeff} < 1$ can be represented. However, as discussed earlier, the coefficients are usually in the $-2 < \text{coeff} < 2$ range. This is handled by storing half the coefficient value, and SMULT performs a multiplication by 2 (Shift left) to compensate for it. This is why the coefficient values need to be half their value—a fact mentioned earlier.

REFERENCES

1. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
2. L.R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
3. H.T. Nagle and V.P. Nelson, "Digital Filter Implementation on 16-bit Microcomputers", *IEEE Micro,* Feb. 1981, pp. 23–41.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

Listing of Code for the Program FILTER

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  1
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

            ;
            ; THIS IS A DEMO PROGRAM TO ILLUSTRATE THE IMPLEMENTATION OF A DIGITAL
            ; FILTER ON THE HPC. THE PROGRAM CAN BE USED TO IMPLEMENT CASCADED
            ; SECOND ORDER STAGES. THE MAXIMUM NUMBER OF CASCADED STAGES POSSIBLE
            ; IS 8 (I.E. THE MAXIMUM FILTER ORDER IS 16).
            ;
            ; THE PROGRAM IS DESIGNED FOR THE ANALOG INTERFACE BEING THROUGH
            ; A CODEC. THE CODEC OUTPUT AND INPUT ARE INTERFACED TO THE HPC USING
            ; MICROWIRE/PLUS. THIS RESTRICTS THE SAMPLING RATE TO 8 KHZ. ALSO, AT
            ; THIS SAMPLING RATE, THE HPC CAN ONLY IMPLEMENT A SECOND ORDER FILTER.
            ; IF A DIFFERENT ANALOG INTERFACE THAT ALLOWS A LOWER SAMPLING RATE IS
            ; USED, HIGHER ORDER FILTERS CAN BE IMPLEMENTED. THIS WILL INVOLVE CHANGES
            ; TO THE FOLLOWING SUBROUTINES: INPUT, OUTPUT AND THE PORTIONS OF INIT
            ; CONCERNED WITH CODEC INITIALIZATION.
            ;
            ; THE PROGRAM IS BASED ON THE DESCRIPTION GIVE IN:
            ;
            ;       H.T. NAGLE AND V.P. NELSON, "DIGITAL FILTER IMPLEMENTATION
            ;       ON 16-BIT MICROCOMPUTERS," IEEE MICRO, FEB. 1981, 23-41.
            ;
            ;
                           .TITLE FILTER
            ;
            ; DEFINE FILTER VARIABLES AND STORAGE.
            ;
0000                       YOUT = M(00)            ; OUTPUT SAMPLE STORAGE.
0002                       YOFK = W(02)            ; TEMPORARY STORAGE.
0004                       NSTG = W(04)            ; NUMBER OF FILTER STAGES.
0006                       NCNT = W(06)            ; TEMPORARY STORAGE.
0008                       PTEMP = W(08)           ; TEMPORARY STORAGE.
000A                       MTEMP = W(0A)           ; TEMPORARY STORAGE.
000C                       A0ADDR = W(0C)          ; ADDRESS OF START OF A0 AREA.
000E                       A1ADDR = W(0E)          ; ADDR. OF START OF A1 AREA.
0010                       A2ADDR = W(010)         ; ADDR. OF START OF A2 AREA.
0012                       B1ADDR = W(012)         ; ADDR. OF START OF B1 AREA.
0014                       B2ADDR = W(014)         ; ADDR. OF START OF B2 AREA.
0016                       M0ADDR = W(016)         ; ADDR. OF START OF M0 AREA.
0018                       M1ADDR = W(018)         ; ADDR. OF START OF M1 AREA.
001A                       M2ADDR = W(01A)         ; ADDR. OF START OF M2 AREA.
001C                       T1ADDR = W(01C)         ; ADDR. OF START OF T1 AREA.
001E                       T2ADDR = W(01E)         ; ADDR. OF START OF T2 AREA.
            ; MAXIMUM NUMBER OF STAGES IS 8.
0020                       A0 = W(020)             ; COEFF. A0.
0030                       A1 = W(030)             ; COEFF. A1.
0040                       A2 = W(040)             ; COEFF. A2.
0050                       B1 = W(050)             ; COEFF. B1.
0060                       B2 = W(060)             ; COEFF. B2.
0070                       M0 = W(070)             ; M(K).
0080                       M1 = W(080)             ; M(K-1).
0090                       M2 = W(090)             ; M(K-2).
00A0                       T1 = W(0A0)             ; T1.
```

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  2
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

      00B0                       T2 = W(0B0)             ; T2.
                 ;
                 ; DEFINITION OF HPC REGISTER NAMES.
                 ;
      00C0                       PSW = M(00C0)
      00D0                       ENIR = M(00D0)
      00D2                       IRPD = M(00D2)
      00D4                       IRCD = M(00D4)
      00D6                       SIO = M(00D6)
      00D8                       PORTI = M(00D8)
      00E2                       PORTBL = M(00E2)
      00E3                       PORTBH = M(00E3)
      00E2                       PORTB = W(00E2)
      00F2                       DIRBL = M(00F2)
      00F3                       DIRBH = M(00F3)
      00F2                       DIRB = W(00F2)
      00F4                       BFUNL = M(00F4)
      00F5                       BFUNH = M(00F5)
      00F4                       BFUN = W(00F4)
      0188                       T2TIM = W(0188)
      0186                       T2REG = W(0186)
      018E                       DIVBYL = M(018E)
      018F                       DIVBYH = M(018F)
      018E                       DIVBY = W(018E)
      0190                       TMMDL = M(0190)
      0191                       TMMDH = M(0191)
      0190                       TMMD = W(0190)
                 ;
                 ; INCLUDE THE MU-LAW TO LINEAR CODE CONVERSION TABLE.
                 ;
                                 .INCLD MUTBL.MAC
F000                             MUTBL, 0F000
                 ;
 F200                            . = 0F200
                 FILTER:
F200 B701F0C4                    LD SP, 01F0             ; INITIALIZE STACK POINTER.
F204 305A                        JSR INIT                ; INITIALIZE THE CODEC
                                                         ; AND FILTER VARIABLES.
                 ; NEXT COMES THE BASIC FILTER LOOP.
                 FLOOP:
F206 3101                        JSR INPUT               ; GET INPUT SAMPLE, OUTPUT
                                                         ; PREVIOUS FILTER OUTPUT.
F208 311B                        JSR YCOMP               ; COMPUTE NEW OUTPUT.
F20A 315B                        JSR OUTPUT              ; CONVERT OUTPUT VALUE TO
                                                         ; MU-255 LAW AND SAVE.
F20C 31AA                        JSR PRECOM              ; PRECOMPUTE FOR NEXT SAMPLE.
F20E 68                          JP FLOOP                ; GO DO NEXT SAMPLE.
                                 .LOCAL
                 ;
                 ;
                 ;
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  3
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                ; THIS SUBROUTINE COMPUTES 2*F*I, WHERE F IS A 2'S COMPLEMENT
                ; BINARY FRACTION AND I IS A 2'S COMPLEMENT INTEGER. THE INTEGER
                ; PART OF THE PRODUCT IS RETURNED IN A. ON INPUT, EITHER F OR I
                ; SHOULD BE IN A AND THE ADDRESS OF THE OTHER IN B.
                ;
                ;
                SMULT:
F20F B700000A                   LD MTEMP, 0             ; CLEAR TEMPORARY STORAGE.
F213 A9CC                       INC B                   ; B NOW POINTS TO UPPER BYTE
                                                        ; OF MULTIPLIER.
F215 17                         IF M(B).7               ; IS IT NEGATIVE?
F216 AB0A                       ST A, MTEMP             ; THEN SAVE MULTIPLICAND IN MTEMP.
F218 AACC                       DECSZ B                 ; B INTO WORD POINTER.
F21A 40                         NOP
F21B AE0A                       X A, MTEMP              ; SWAP A AND MTEMP.
F21D 960B17                     IF M(($MTEMP) + 1).7    ; IS MULTIPLICAND NEGATIVE?
F220 F8                         ADD A, W(B)             ; THEN ACCUMULATE MULTIPLIER
F221 AE0A                       X A, MTEMP
F223 FE                         MULT A, W(B)            ; UNSIGNED MULTIPLY.
F224 AECE                       X A, X                  ; UPPER HALF IN A.
F226 02                         SET C
F227 960AEB                     SUBC A, MTEMP
F22A E7                         SHL A
F22B 96CF17                     IF H(X).7
F22E 04                         INC A
F22F E7                         SHL A
F230 96CF16                     IF H(X).6
F233 04                         INC A
F234 3C                         RET
                ;
                ;
                ; THIS SUBROUTINE PERFORMS THE INITIALIZATION FOR THE FILTER.
                ; IT DOES THE FOLLOWING:
                ;               1. SET UP THE FILTER VARIABLES.
                ;               2. COPY THE FILTER COEFFS. FROM ROM TO ON CHIP RAM.
                ;               3. INITIALIZE AND START THE CODEC.
                ;
                ;
                ; DEFINE FILTER COEFFICIENTS.
                ;
F235 40                         .EVEN
F236 0400       ROMNST:         .WORD 4
F238 C430       ROMA0:          .WORD 12484, 3217, 4574, 7636
F23A 910C
F23C DE11
F23E D41D
F240 C430       ROMA1:          .WORD 12484, 3217, 4574, 7636
F242 910C
F244 DE11
F246 D41D
F248 C430       ROMA2:          .WORD 12484, 3217, 4574, 7636
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  4
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

F24A 910C
F24C DE11
F24E D41D
F250 B939        ROMB1:          .WORD 14777, 9826, 19308, 11207
F252 6226
F254 6C4B
F256 C72B
F258 A1D5        ROMB2:          .WORD -10847, -15783, -6940, -14068
F25A 59C2
F25C E4E4
F25E 0CC9
                 INIT:
F260 B6F236A8                    LD A, W(ROMNST)
F264 AB04                        ST A, NSTG               ; SET UP NO. OF STAGES.
F266 9020                        LD A, $A0
F268 AB0C                        ST A, A0ADDR             ; COPY ADDRESS OF A0 AREA.
F26A 9030                        LD A, $A1
F26C AB0E                        ST A, A1ADDR             ; COPY ADDRESS OF A1 AREA.
F26E 9040                        LD A, $A2
F270 AB10                        ST A, A2ADDR             ; COPY ADDRESS OF A2 AREA.
F272 9050                        LD A, $B1
F274 AB12                        ST A, B1ADDR             ; COPY ADDRESS OF B1 AREA.
F276 9060                        LD A, $B2
F278 AB14                        ST A, B2ADDR             ; COPY ADDRESS OF B2 AREA.
F27A 9070                        LD A, $M0
F27C AB16                        ST A, M0ADDR             ; COPY ADDRESS OF M0 AREA.
F27E 9080                        LD A, $M1
F280 AB18                        ST A, M1ADDR             ; COPY ADDRESS OF M1 AREA.
F282 9090                        LD A, $M2
F284 AB1A                        ST A, M2ADDR             ; COPY ADDRESS OF M2 AREA.
F286 90A0                        LD A, $T1
F288 AB1C                        ST A, T1ADDR             ; COPY ADDRESS OF T1 AREA.
F28A 9080                        LD A, $T2
F28C AB1E                        ST A, T2ADDR             ; COPY ADDRESS OF T2 AREA.
                 ;
                 ; COPY THE A0 COEFFS. TO ON-CHIP RAM.
                 ;
F28E B3F238                      LD X, ROMA0
F291 9220                        LD B, $A0
F293 AC04CA                      LD K, NSTG
                 CA0LP:
F296 F0                          LD A, W(X+)
F297 E1                          XS A, W(B+)
F298 40                          NOP
F299 AACA                        DECSZ K
F29B 65                          JP CA0LP
                 ;
                 ; COPY THE A1 COEFFS. TO ON-CHIP RAM.
F29C B3F240                      LD X, ROMA1
F29F 9230                        LD B, $A1
F2A1 AC04CA                      LD K, NSTG
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  5
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                     CA1LP:
F2A4 F0                                  LD A, W(X+)
F2A5 E1                                  XS A, W(B+)
F2A6 40                                  NOP
F2A7 AACA                                DECSZ K
F2A9 65                                  JP CA1LP
                     ;
                     ; COPY THE A2 COEFFS. TO ON-CHIP RAM.
F2AA B3F248                              LD X, ROMA2
F2AD 9240                                LD B, $A2
F2AF AC04CA                              LD K, NSTG
                     CA2LP:
F2B2 F0                                  LD A, W(X+)
F2B3 E1                                  XS A, W(B+)
F2B4 40                                  NOP
F2B5 AACA                                DECSZ K
F2B7 65                                  JP CA2LP
                     ;
                     ; COPY THE B1 COEFFS. TO ON-CHIP RAM.
F2BB B3F250                              LD X, ROMB1
F2BB 9250                                LD B, $B1
F2BD AC04CA                              LD K, NSTG
                     CB1LP:
F2C0 F0                                  LD A, W(X+)
F2C1 E1                                  XS A, W(B+)
F2C2 40                                  NOP
F2C3 AACA                                DECSZ K
F2C5 65                                  JP CB1LP
                     ;
                     ; COPY THE B2 COEFFS. TO ON-CHIP RAM.
F2C6 B3F258                              LD X, ROMB2
F2C9 9260                                LD B, $B2
F2CB AC04CA                              LD K, NSTG
                     CB2LP:
F2CE F0                                  LD A, W(X+)
F2CF E1                                  XS A, W(B+)
F2D0 40                                  NOP
F2D1 AACA                                DECSZ K
F2D3 65                                  JP CB2LP
                     ;
                     ; ZERO OUT THE REST OF USER BASE PAGE RAM.
                     ;
F2D4 8D70BE                              LD BK, $M0, 0BE
                     ZEROLP:
F2D7 00                                  CLR A
F2D8 E1                                  XS A, W(B+)
F2D9 62                                  JP ZEROLP
                     ;
                     ;
                     ; NOW INITIALIZE AND START THE CODEC.
                     ;
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  6
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                ;
F2DA B7FFB7F2                   LD DIRB,0FFB7           ; SET B3 (T2IO) AND B6 (SK)
                                                        ; ON PORT B AS INPUTS. SET ALL
                                                        ; OTHER PINS ON B AS OUTPUT.
F2DE B70000E2                   LD PORTB, 0             ; OUTPUT 0 ON ALL PORT B PINS.
F2E2 96F40B                     SET BFUNL.3             ; ALT. FUN. ON B3-T2IO.
F2E5 96F40D                     SET BFUNL.5             ; ALT. FUN. ON B5-SO.
F2E8 96F508                     SET BFUNH.0             ; ALT. FUN. ON B8-TSO.
F2EB 9700D0                     LD ENIR, 0              ; DISABLE INTRPTS.
F2EE 9700D4                     LD IRCD, 0              ; SELECT SLAVE MODE FOR M-WIRE.
F2F1 83070188AB                 LD T2TIM, 07            ; LOAD 7-DEC INTO T2 TIMER.
F2F6 83070186AB                 LD T2REG, 07            ; LOAD 7-DEC INTO T2 REG.
F2FB 8300018F8B                 LD DIVBYH, 0            ; SELECT EXT. CLOCK FOR T2 TIMER.
                ;
F300 8ED6                       X A, SIO
F302 8740400190AB               LD TMMD, 04040          ; START TIMER T2.
F308 3C                         RET
                ;
                ;
                ; THIS SUBROUTINE OUTPUTS THE PREVIOUS Y(K) TO THE CODEC AND READS
                ; THE NEW INPUT VALUE. THEN THE MU-255 VALUE IS CONVERTED TO LINEAR
                ; BY TABLE LOOK UP. THE TABLE IS ASSUMED TO START AT F000.
                ;
                ;
                INPUT:
F309 AB02                       LD A, YOFK              ; GET DATA TO BE OUTPUT.
                NOTDN:
F30B 96D210                     IF IRPD.0               ; IS MICROWIRE DONE?
F30E 41                         JP MWDONE               ; YES, SO GET DATA.
F30F 64                         JP NOTDN                ; NO, SO TRY AGAIN.
                MWDONE:
F310 8ED6                       X A, SIO                ; GET NEW SAMPLE, OUTPUT
                                                        ; COMPUTED DATA.
F312 01                         COMP A                  ; TAKE CARE OF CODEC INVERSION.
F313 99FF                       AND A, 0FF
F315 E7                         SHL A
F316 BAF000                     OR A, 0F000             ; FORM MU-LAW TO LINEAR
                                                        ; TABLE ADDRESS.
F319 AECE                       X A, X
F31B D0                         LD A, M(X+)             ; GET LINEAR VALUE
F31C AECA                       X A, K
F31E D4                         LD A, M(X)              ; A BYTE AT A TIME.
F31F 8CC8CB                     LD H(K), L(A)
F322 A8CA                       LD A, K
F324 3C                         RET
                ;
                ;
                YCOMP:
                ; THIS SUBROUTINE COMPUTES THE OUTPUT SAMPLE Y(K).
                ; THE INPUT SAMPLE X(K) IS INPUT IN REG. A.
                ; THE OUTPUT IS RETURNED IN REG. A.
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  7
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                ;
F325 AC0406                     LD NCNT, NSTG           ; COPY THE NUMBER OF STAGES TO
                                                        ; NCNT.
                YLOOP:
F328 AD1CF8                     ADD A, W(T1ADDR)        ; A ≤ X(K) + T1.
F32B AD16AB                     ST A, W(M0ADDR)         ; M(K) ≤ X(K) + T1.
F32E AC0CCC                     LD B, A0ADDR            ; B ≤ ADDR(A0).
F331 3522                       JSR SMULT               ; A ≤ A0*M(K).
F333 AD1EF8                     ADD A, W(T2ADDR)        ; A ≤ A0*M(K) + T2.
                                                        ;
F336 AA06                       DECSZ NCNT              ; DONE ALL STAGES?
F338 941B       R               JMP YMORE               ; NO GO DO SOME MORE.
                                                        ;
                                                        ; GET HERE MEANS ALL STAGES DONE.
F33A AB02                       ST A, YOFK              ; SAVE TEMPORARILY.
F33C A804                       LD A, NSTG
F33E 05                         DEC A
F33F E7                         SHL A
F340 01                         COMP A
F341 04                         INC A                   ; A ≤ -2*(NSTG-1).
F342 A0C81CF8                   ADD T1ADDR, A           ; RESTORE T1ADDR.
F346 A0C816F8                   ADD M0ADDR, A           ; RESTORE M0ADDR.
F34A A0C80CF8                   ADD A0ADDR, A           ; RESTORE A0ADDR.
F34E A0C81EF8                   ADD T2ADDR, A           ; RESTORE T2ADDR.
F352 A802                       LD A, YOFK              ; A ≤ Y(K).
F354 3C                         RET
                ;
                ; PREPARE FOR NEXT STAGE ITERATION.
                ;
                YMORE:
F355 82021CF8                   ADD T1ADDR, 02
F359 820216F8                   ADD M0ADDR, 02
F35D 82020CF8                   ADD A0ADDR, 02
F361 82021EF8                   ADD T2ADDR, 02
F365 953D                       JMP YLOOP
                ;
                ; THIS SUBROUTINE CONVERTS THE 16 BIT OUTPUT VALUE TO
                ; 8 BIT MU-LAW.
                ;
                OUTPUT:
F367 96D41F                     RESET IRCD.7
F36A E7                         SHL A                   ; SIGN BIT TO C.
F36B 06                         IFN C                   ; IS IT POSITIVE?
F36C 45                         JP OPOS
F36D 96D40F                     SET IRCD.7
F370 01                         COMP A
F371 04                         INC A                   ; NEGATIVE, SO TAKE 2'S
                                                        ; COMPLEMENT.
                ; OPOS:
F372 B80108                     ADD A, 0108             ; ADD BIAS.
F375 9107                       LD K, 07                ; SET UP COUNTER.
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  8
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                   ALIGN:
F377 E7                            SHL A                   ; LOOP AND LOCATE MS 1 BIT.
F378 07                            IF C
F379 44                            JP ODONE                ; FOUND MS 1 BIT.
F37A AACA                          DECSZ K
F37C 65                            JP ALIGN
F37D E7                            SHL A                   ; HAS TO BE 1 IN C NOW.
                   ODONE:
F37E AECA                          X R, K
F380 E7                            SHL A
F381 E7                            SHL A
F382 E7                            SHL A
F383 E7                            SHL A                   ; COUNTER VALUE IN BITS 4-6.
F384 AECC                          X A, B
F386 00                            CLR A
F387 88CB                          LD A, H(K)
F389 3B                            SWAP A
F38A 990F                          AND A, 0F
F38C 96CCFA                        OR A, B
F38F 96D417                        IF IRCD.7
F392 96C80F                        SET A.7
F395 01                            COMP A
F396 8B00                          ST A, YOUT
F398 3C                            RET
                   ;
                   ; THIS SUBROUTINE UPDATES M(K-1) AND M(K-2) FOR THE NEXT SAMPLE.
                   ;
F399 AC1ACC                        LD B, M2ADDR            ; B ≤ ADDR(M2),
F39C AC04CA                        LD K, NSTG              ; K ≤ NSTG.
F39F AC18CE                        LD X, M1ADDR            ; X ≤ ADDR(M1).
                   DLYLP1:
F3A2 F0                            LD A, W(X+)             ; A ≤ M(K-1).
F3A3 E1                            XS A, W(B+)             ; M(K-2) ≤ M(K-1).
F3A4 40                            NOP
F3A5 AACA                          DECSZ K
F3A7 65                            JP DLYLP1
                   ;
F3A8 AC18CC                        LD B, M1ADDR            ; B ≤ ADDR(M1),
F3AB AC04CA                        LD K, NSTG              ; K ≤ NSTG.
F3AE AC16CE                        LD X, M0ADDR            ; X ≤ ADDR(M0).
                   DLYLP2:
F3B1 F0                            LD A, W(X+)             ; A ≤ M(K).
F3B2 E1                            XS A, W(B+)             ; M(K-1) ≤ M(K).
F3B3 40                            NOP
F3B4 AACA                          DECSZ K
F3B6 65                            JP DLYLP2
F3B7 3C                            RET
                   ;
                   ;
                   PRECOMP:
                   ; THIS SUBROUTINE PRECOMPUTES T1 AND T2 BEFORE THE NEXT INPUT
```

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  9
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

                ; SAMPLE ARRIVES.
                ;
F3B8 AC0406                     LD NCNT, NSTG           ; COPY NO. OF STAGES.
                PRELP:
F3BB AD18A8                     LD A, W(M1ADDR)         ; A ≤ M(K-1).
F3BE AC12CC                     LD B, B1ADDR            ; B ≤ ADDR(-B1).
F3C1 35B2                       JSR SMULT               ; A ≤ -B1*M(K-1).
F3C3 AB08                       ST A, PTEMP
F3C5 AD1AA8                     LD A, W(M2ADDR)         ; A ≤ M(K-2).
F3C8 AC14CC                     LD B,B2ADDR             ; B ≤ ADDR(-B2).
F3CB 35BC                       JSR SMULT               ; A ≤ -B2*M(K-2).
F3CD 9608F8                     ADD A, PTEMP            ; A ≤ -B1*M(K-1) - B2*M(K-2).
F3D0 AD1CAB                     ST A,W(T1ADDR)
F3D3 AD18A8                     LD A, W(M1ADDR)         ; A ≤ M(K-1).
F3D6 AC0ECC                     LD B, A1ADDR            ; B ≤ ADDR(A1).
F3D9 35CA                       JSR SMULT               ; A ≤ A1*M(K-1).
F3DB AB08                       ST A, PTEMP
F3DD AD1AA8                     LD A, W(M2ADDR)         ; A ≤ M(K-2).
F3E0 AC10CC                     LD B, A2ADDR            ; B ≤ ADDR(A2).
F3E3 3504                       JSR SMULT               ; A ≤ A2*M(K-2).
F3E5 9608F8                     ADD A, PTEMP            ; A ≤ A1*M(K-1) + A2*M(K-2).
                                                        ;
F3E8 AA06                       DECSZ NCNT              ; DONE ALL STAGES?
F3EA 9427                       JMP PMORE               ; NO, GO DO SOME MORE.
                                                        ;
                                                        ; GET HERE MEANS DONE ALL STAGES.
F3EC A804                       LD A, NSTG
F3EE 05                         DEC A
F3EF E7                         SHL A
F3F0 01                         COMP A
F3F1 04                         INC A                   ; A ≤ -2*(NSTG - 1).
F3F2 A0C818F8                   ADD M1ADDR, A           ; RESTORE M1ADDR.
F3F6 A0C81AF8                   ADD M2ADDR, A           ; RESTORE M2ADDR.
F3FA A0C81CF8                   ADD T1ADDR, A           ; RESTORE T1ADDR.
F3FE A0C81EF8                   ADD T2ADDR, A           ; RESTORE T2ADDR.
F402 A0C812F8                   ADD B1ADDR, A           ; RESTORE B1ADDR.
F406 A0C814F8                   ADD B2ADDR, A           ; RESTORE B2ADDR.
F40A A0C80EF8                   ADD A1ADDR, A           ; RESTORE A1ADDR.
F40E A0C810F8                   ADD A2ADDR, A           ; RESTORE A2ADDR.
F412 3C                         RET
                ;
                ; PREPARE FOR NEXT STAGE ITERATION.
                ;
                PMORE:
F413 820218F8                   ADD M1ADDR, 02          ; UPDATE M1ADDR.
F417 82021AF8                   ADD M2ADDR, 02          ; UPDATE M2ADDR.
F41B 82021CF8                   ADD T1ADDR, 02          ; UPDATE T1ADDR.
F41F 82021EF8                   ADD T2ADDR, 02          ; UPDATE T2ADDR.
F423 820212F8                   ADD B1ADDR, 02          ; UPDATE B1ADDR.
F427 820214F8                   ADD B2ADDR, 02          ; UPDATE B2ADDR.
F42B 82020EF8                   ADD A1ADDR, 02          ; UPDATE A1ADDR.
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  10
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER

445 F42F 820210F8                    ADD A2ADDR, 02          ; UPDATE A2ADDR.
446 F433 9578                        JMP PRELP
447                 ;
448                 ;
449 FFFE 00F2                        .END FILTER
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  11
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER
SYMBOL TABLE

A       00C8 W        A0      0020 W        A0ADDR  000C W        A1      0030 W
A1ADDR  000E W        A2      0040 W        A2ADDR  0010 W        ALIGN   F377
B       00CC W        B1      0050 W        B1ADDR  0012 W        B2      0060 W
B2ADDR  0014 W        BFUN    00F4 W*       BFUNH   00F5 M        BFUNL   00F4 M
CA0LP   F296          CA1LP   F2A4          CA2LP   F2B2          CB1LP   F2C0
CB2LP   F2CE          DIRB    00F2 W        DIRBH   00F3 M*       DIRBL   00F2 M*
DIVBY   018E W*       DIVBYH  018F M        DIVBYL  018E M*       DLYLP1  F3A2
DLYLP2  F3B1          ENIR    00D0 M        FILTER  F200          FLOOP   F206
INCRM   0200          INIT    F260          INPUT   F309          IRCD    00D4 M
IRPD    00D2 M        K       00CA W        M0      0070 W        M0ADDR  0016 W
M1      0080 W        M1ADDR  0018 W        M2      0090 W        M2ADDR  001A W
MTEMP   000A W        MUAL    205F          MWDONE  F310          NCNT    0006W
NOTDN   F30B          NSTG    0004 W        ODONE   F37E          OPOS    F372
OUTPUT  F367          PC      00C6 W        PMORE   F413          PORTB   00E2 W
PORTBH  00E3 M*       PORTBL  00E2 M*       PORTI   0008 M*       PRECOM  F3B8
PRELP   F3BB          PSW     00C0 M*       PTEMP   0008 W        ROMA0   F238
ROMA1   F240          ROMA2   F248          ROMB1   F250          ROMB2   F258
ROMNST  F236          RVAL    E0A1          SIO     00D6 M        SMULT   F20F
SP      00C4 W        SVAL    2100          T1      00A0 W        T1ADDR  001C W
T2      00B0 W        T2ADDR  001E W        T2REG   0186 W        T2TIM   0188 W
TMMD    0190 W        TMMDH   0191 M*       TMMDL   0190 M*       X       00CE W
YCOMP   F325          YLOOP   F328          YMORE   F355          YOFK    0002 W
YOUT    0000 M        ZEROLP  F2D7
```

APPENDIX A (Continued)

Listing of Code for the Program FILTER (Continued)

```
NATIONAL SEMICONDUCTOR CORPORATION   PAGE:  12
HPC CROSS ASSEMBLER, REV:C, 30 JUL 86
FILTER
MACRO TABLE

MUTBL
   NO WARNING LINES

   NO ERROR LINES

 1079 ROM BYTES USED

SOURCE CHECKSUM = 4769
OBJECT CHECKSUM = 1378

INPUT   FILE C:FILTER.MAC
LISTING FILE C:FILTER.PRN
OBJECT  FILE C:FILTER.LM
```

AN-486

A Floating Point Package for the HPC

National Semiconductor
Application Note 486
Ashok Krishnamurthy

INTRODUCTION

This report describes the implementation of a Single Precision Floating Point Arithmetic package for the National Semiconductor HPC microcontroller. The package is based upon the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754−1985). However, the package is not a conforming implementation of the standard. The differences between the HPC implementation and the standard are described later in this report.

The following single precision (SP) operations have been implemented in the package.

(1) FADD. Addition of two SP floating point (FLP) numbers.

(2) FSUB. Subtraction of two SP FLP numbers.

(3) FMULT. Multiplication of two SP FLP numbers.

(4) FDIV. Division of two SP FLP numbers.

(5) ATOF. Convert an ASCII string representing a decimal FLP number to a binary SP FLP number.

(6) FTOA. Convert a binary SP FLP number to a decimal FLP number and output the decimal FLP number as an ASCII string.

The report is organized as follows. The next section discusses the representation of FLP numbers. Then, the differences between the HPC implementation and the IEEE/ANSI standard are described. This is followed by a description of the algorithms used in the computations. Appendix A is a User's Manual for the package, Appendix B describes the test data for the package and Appendix C is a listing of the code.

Note that this report assumes that the reader is familiar with the IEEE/ANSI Binary Floating-Point Standard. Please refer to this document for an explanation of the terms used here.

REPRESENTATION OF FLOATING POINT NUMBERS

The specification of a binary floating point number involves two parts: a mantissa and an exponent. The mantissa is a signed fixed point number and the exponent is a signed integer. The IEEE/ANSI standard specifies that a SP FLP number shall be represented in 32 bits as shown in *Figure 1.*

1	8	23
S	E	F

FIGURE 1

The significance of each of these fields is as follows:

1. S—this 1-bit field is the sign of the mantissa. S = 0 means that the number is positive, while S = 1 means that it is negative.
2. E—this is the 8-bit exponent field. The exponent is represented as a biased value with a bias of 127-decimal.
3. F—this is the 23-bit mantissa field. For normalized FLP numbers (see below), a MSB of 1 is assumed and not represented. Thus, for normalized numbers, the value of the mantissa is 1.F. This provides an effective precision of 24 bits for the mantissa.

Normalized FLP number: A binary FLP number is said to be normalized if the value of the MSB of the mantissa is 1. Normalization is important and useful because it provides maximum precision in the representation of the number. If we deal with normalized numbers only (as the HPC implementation does) then since the MSB of the mantissa is always 1, it need not be explicitly represented. This is as specified in the IEEE/ANSI standard.

Given the values of S , E and F, the value of the SP FLP number is obtained as follows.

If 0 < E < 255, then the FLP number is (−1) ^S*1.F*2^(E−127).

If E = 0, then the value of the FLP number is 0.

If E = 255, then the FLP number is not a valid number (NAN).

The above format for binary SP FLP numbers provides for the representation of numbers in the range −3.4*10^38 to −1.75*10^−38, 0, and 1.75*10^−38 to 3.4*10^38. The accuracy is between 7 and 8 decimal digits.

DIFFERENCES BETWEEN THE IMPLEMENTATION AND THE IEEE/ANSI STANDARD

The IEEE/ANSI standard specifies a comprehensive list of operations and representations for FLP numbers. Since an implementation that fully conforms to this standard would lead to an excessive amount of overhead, a number of the features in the standard were dropped. This section describes the differences between the implemented package and the standard.

1. Omission of −0. The IEEE/ANSI standard requires that both + and − zero be represented, and arithmetic carried out using both. The implementation does not represent −0. Only +0 is represented and arithmetic is carried out with +0 only.
2. Omission of Infinity Arithmetic. The IEEE/ANSI standard provides for the representation of plus and minus Infinity, and requires that valid arithmetic operations be carried out on Infinity. The HPC implementation does not support this.
3. Omission of Quiet NaN. The IEEE/ANSI standard provides for both quiet and signalling NaNs. The HPC implementation provides for signalling NaNs only. A signalling NaN can be produced as the result of overflow during an arithmetic operation. If the NaN is passed as input to further floating point routines, then these routines will produce another NaN as output. The routines will also set the Invalid Operation flag, and call the user floating point error trap routine at address FPTRAP.
4. Omission of denormalized numbers. Denormalized numbers are FLP numbers with a biased exponent, E of zero and a non zero mantissa F. Such denormalized numbers are useful in providing gradual underflow to zero. Denormalized numbers are not represented or used in the HPC implementation. Instead, if the result of a computation cannot be represented as a normalized number within the allowable exponent range, then an underflow is signaled, the result is set to zero, and the user floating point error trap routine at address FPTRAP is called.
5. Omission of the Inexact Result exception. The IEEE/ANSI standard requires that an Inexact Result exception be signaled when the rounded result of an operation is not exact, or it overflows without an overflow trap. This feature is not provided in the HPC implementation.

6. Biased Rounding to Nearest. The IEEE/ANSI standard requires that rounding to nearest be provided as the default rounding mode. Further, the rounding is required to be unbiased. The HPC implementation provides biased rounding to nearest only. An example will help clarify this.

   Suppose the result of an operation is .b1b2b3XXX and needs to be rounded to 3 binary digits. Then if XXX is 0YY, the round to nearest result is .b1b2b3. If XXX is 1YY, with at least one of the Y's being 1, then the result is .b1b2b3 + 0.001. Finally if XXX is 100, it is a tie situation. In such a case, the IEEE/ANSI standard requires that the rounded result be such that its LSB is 0. The HPC implementation, on the other hand, will round the result in such a case to .b1b2b3 + 0.001.

DESCRIPTION OF ALGORITHMS

1. **General Considerations.** The HPC implementation of the SP floating point package consists of a series of subroutines. The subroutines have been designed to be compatible with the CCHPC C Cross Compiler. They have, however, not been tested with the CCHPC Cross Compiler.

   The Arithmetic subroutines that compute F1 op F2 (where op is +, −, * or /) expect that F1 and F2 are input in the IEEE format. Each of F1 and F2 consists of two 16-bit words organized as follows.

   Fn−HI: S EXP 7 MS bits of F

   Fn−LO: 16 LS bits of F

   In the above, S is the sign of the mantissa, EXP is the biased exponent, and F is the mantissa.

   On input it is assumed that F1−HI is in register K, F1−LO is in the accumulator A, and F2−HI and F2−LO are on the stack just below the return address i.e., F2−HI is at W(SP-4) and F2−LO is at W(SP-6). The result, C, is also returned in IEEE format with C−HI in register K and C−LO in the accumulator A.

   The two Format Conversion routines, ATOF and FTOA expect that on entry, register B contains the address of the start of the ASCII byte string representing the decimal FLP number. ATOF reads the byte string starting from this address. Note that the string must be terminated with a null byte. The binary floating point number is returned in registers K and A. FTOA, on the other hand, writes the decimal FLP string starting from the address in register B on entry. A terminating null byte is also output. Also, FTOA expects that the binary FLP number to be converted is in registers K and A on entry.

   Most of the storage required by the subroutines is obtained from the stack. Two additional words of storage in the base page are also used. The first is W(0), and is referenced in the subroutines as W(TMP1). The second word of storage can be anywhere in the base page and is used to store the sticky flags used to signal floating point exceptions. This is referenced in the subroutines as W(FPERWD). Thus any user program that uses the floating point package needs to have the symbols TMP1 and FPERWD defined appropriately.

2. **Exception Handling.** The following types of exception can occur during the course of a computation.

   (i) Invalid Operand. This exception occurs if one of the input operands is a NaN.

   (ii) Exponent Overflow. This occurs if the result of a computation is such that its exponent has a biased value of 255 or more.

   (iii) Exponent Underflow. This occurs if the result of a computation is such that its exponent is 0 or less.

   (iv) Divide-by-zero. This exception occurs if the FDIV routine is called with F2 being zero.

   The package signals exceptions in two ways. First a word at address FPERWD is maintained that records the history of these exception conditions. Bits 0–3 of this word are used for this purpose.

   Bit 0—Set on Exponent Overflow.

   Bit 1—Set on Exponent Underflow.

   Bit 2—Set on Illegal Operand.

   Bit 3—Set on Divide-by-zero.

   These bits are never cleared by the floating point package, and can be examined by the user software to determine the exception conditions that occurred during the course of a computation. It is the responsibility of the user software to initialize this word before calling any of the floating point routines.

   The second method that the package uses to signal exceptions is to call a user floating point exception handler subroutine whenever an exception occurs. The corresponding exception bit in FPERWD is set before calling the handler. The starting address of the handler should be defined by the symbol FPTRAP.

3. **Unpacked Floating Point Format.** The IEEE/ANSI standard floating point format described earlier is very cumbersome to deal with during computation. This is primarily because of the splitting of the mantissa between the two words. The subroutines in the package unpack the input FLP numbers into an internal representation, do the computations using this representation, and finally pack the result into the IEEE format before return to the calling program. The unpacking is done by the subroutine FUNPAK and the packing by the subroutine FPAK. The unpacked format consists of 3 words and is organized as follows.

Fn-EXP.Fn-SIGN	8 bits biased exponent	sign (extended to 8 bits)
Fn-HI	MS 16 bits of mantissa (implicit 1 is present as MSB)	
Fn-LO	LS 8 bits of mantissa	Eight Zeros

   Since all computations are carried out in this format, note that the result is actually known to 32 bits. This 32-bit mantissa is rounded to 24 bits before being packed to the IEEE format.

4. **Algorithm Description.** All the arithmetic algorithms first check for the easy cases when either F1 or F2 is zero or a NaN. The result in these cases is immediately available. The description of the algorithms below is for those cases when neither F1 nor F2 is zero or a NaN. Also, in order to keep the algorithm description simple, the check for underflow/overflow at the various stages is not shown. The documentation in the program, the descriptions given below, and the theory as described in the references should allow these programs to be easily maintained.

   (i) FADD.

   The processing steps are as follows:

   1. Compare F1-EXP and F2-EXP. Let the difference be D. Shift right the mantissa (Fn-HI.Fn-LO, n = 1 or 2) of the FLP number with the smaller exponent D times. Let the numbers after this step be F1-EXP.F1-SIGN, F1-HI, F1-LO and F2-EXP.F2-SIGN,

F2-HI and F2-LO. This step equalizes the two exponents.

2. Take the XOR of F1-SIGN and F2-SIGN. If this is 0, then go to step 4, else go to step 3.

3. Do a true subtract of F2-LO from F1-LO. (A true subtract is when the SUBC instruction is preceded by a SET C instruction.) Then do a 1's complement subtract of F2-HI from F1-HI. If the last subtract resulted in C = 1, then go to step 3.2, else go to step 3.1.

   3.1. Get here means that F2 is larger than F1, and the computed result is negative. Take the 2's complement of the result to make it positive. Set the sign of the result to be the sign of F2. Go to step 3.3.

   3.2. Get here means F1 is larger than F2, and the result of the mantissa subtract is positive. Set the sign of the result to be the sign of F1. Go to step 3.3.

   3.3. The result after a subtract need not be normalized. Shift left the result mantissa until its MSB is 1. Decrement the exponent of the result by 1 for each such left shift. Go to step 5.

4. Add F2-LO to F1-LO. Next add with any carry from the previous add, F2-HI to F1-HI. If this last add results in C = 1, then go to step 4.1, else go to step 5.

   4.1. Rotate Right with carry C-HI. Next load C-LO in and rotate it right with carry. Increase the exponent of the result, C by 1. Go to step 5.

5. Round the result. Go to step 6.

6. Pack the result and return.

(ii) FSUB.

The processing steps are as follows:

1. Copy F2 to the stack and change its sign. Go to step 2.

2. Call FADD.

3. Remove the copy of -F2 from the stack and return.

(iii) FMULT.

The processing steps are as follows.

1. Add F1-EXP and F2-EXP to get C1-EXP. Subtract from C1-EXP 127-decimal which is the IEEE bias, to get C-EXP. Go to step 2.

2. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 3.

3. Compute F1-HI*F2-HI. Let the upper half of the product be C1-HI and the lower half C1-LO. Go to step 4.

4. Compute F1-HI*F2-LO. Let the upper half of this product be C2-HI. Add C2-HI to C1-LO to give C11-LO. If this last add results in C = 1, then increment C1-HI. Go to step 5.

5. Compute F1-LO*F2-HI. Let the upper half of this product be C3-HI. Add C3-HI to C11-LO to get C12-LO. If this last add results in C = 1, then increment C1-HI. Go to step 6.

6. Mantissa normalization. If the MSB of C1-HI is 1, then increment C-EXP, else shift left C1-HI.C12-LO. Go to step 7.

7. Round C1-HI.C12-LO to get C-HI.C-LO. Go to step 8.

8. Pack C-EXP.C-SIGN, C-HI and C-LO and return as the answer.

(iv) FDIV.

The processing steps are as follows:

1. Compare F1-HI and F2-HI. If F2-HI is greater than F1-HI then go to Step 3, else go to step 2.

2. Shift right F1-HI.F1-LO. Increase F1-EXP by 1.

3. Subtract F2-EXP from F1-EXP. Add to the result 127-decimal to get C1-EXP. Go to step 4.

4. Take the XOR of F1-SIGN and F2-SIGN to get C-SIGN. Go to step 5.

5. Compute F1-HI*F2-LO. Let the result be M1-HI.M1-LO. Go to step 6.

6. Divide M1-HI.M1-LO by F2-HI. Let the quotient be M2-HI. Go to step 7.

7. Do a true subtract of M2-HI from F1-LO. Let the result be M3-LO. If C = 1 as a result of this subtract, then go to step 8, else decrement F1-HI and go to step 8.

8. Divide F1-HI.M3-LO by F2-HI. Let the quotient be C1-HI and the remainder R1. Go to step 9.

9. Divide R1 .0000 by F2-HI. Let the quotient be C1-LO. Go to step 10.

10. If the MSB of C1-HI is 1 then go to step 11, else shift left C1-HI.C1-LO, decrease C1-EXP by 1 and go to step 11.

11. Round C1-HI.C1-LO to get C-HI.C-LO. go to step 12.

12. Pack C1-EXP.C-SIGN, C-HI and C-LO and return as the result.

(v) ATOF.

The processing steps in this case are as follows.

1. Set M-SIGN, the mantissa sign to 0.

   Set M10-EXP, the implicit decimal exponent to 0.

   Set HI-INT to 0.

   Set LO-INT to 0.

   Go to step 2.

2. Get a character from the input string. Let the character be C.

   If C is a '+', then go to the start of step 2.

   If C is a '−', then set M-SIGN to FF and go to start of step 2.

   If C is a '.', then go to step 5.

   If C is none of the above, then go to step 3.

3. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 3.1, else go to step 3.2.

   3.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Go to step 4.

   3.2. Increase M10-EXP by 1 and go to step 4.

4. Get a character from the input string. Let the character be C.

   If C is a '.', then go to step 5.

   If C is a 'E', then go to step 7.

   If C is the space character, then go to the start of step 4.

   If C is none of the above, then go to step 3.

5. Get a character from the input string. Let the character be C.

   If C is a 'E', then go to step 7.

   If C is the space character, then go to the start of step 5.

   If C is none of the above, then go to step 6.

6. Subtract 30 from C to get its integer value. Let this be I. Check and see if (HI-INT.LO-INT)*10 + 9 can fit in 32 bits. If it can, then go to step 6.1, else go to step 5.

   6.1. Multiply HI-INT.LO-INT by 10 and add I to the product. Store this sum back in HI-INT.LO-INT. Decrement M10-EXP by 1. Go to step 5.

7. Set SEXP, the exponent sign to be 0. Go to step 8.

8. Get a character from the input string. Let the character be C.

   If C is a '+', then go to start of step 8.

   If C is a '−', then set SEXP to be FF and go to the start of step 8.

   If C is none of the above, then go to step 9.

9. Set M20-EXP, the explicit decimal exponent to 0. Go to step 10.

10. Subtract 30 from C to get its integer value. Let this be I. Multiply M20-EXP by 10 and add I to the product. Store this sum back in M20-EXP. Go to step 11.

11. Get a character from the input string. Let this be C. If C is the null character, then go to step 12, else go to step 10.

12. Add M10-EXP and M20-EXP (with the proper sign as determined by SEXP) to get the 10's exponent M-EXP. Save in M-EXP the magnitude of the sum and in SEXP the sign of the sum. Go to step 13.

13. Check and see if HI-INT.LO-INT is 0. If it is, then set the resulting floating point number, C, to zero and return. If it is not then go to step 14.

14. Normalize HI-INT.LO-INT by left shifts such that the MSB is 1. Let the number of left shifts needed to do this be L. Set B1-EXP to 32-decimal − L. Go to step 15.

15. If SEXP is 0, then set P-HI.P-LO to the binary representation of 0.625, else set P-HI.P-LO to the binary representation of 0.8. Go to step 16.

16. Multiply HI-INT.LO-INT by P-HI.P-LO M-EXP times. After each multiplication, normalize the partial product if needed by left shifting. Accumulate the number of left shifts needed in B2-EXP. Let the final product be C-HI.C-LO. Go to step 17.

17. Subtract B2-EXP from B1-EXP. Let the result be B-EXP. Go to step 18.

18. If SEXP is 0, then multiply M-EXP by 4, else multiply M-EXP by −3. Let the result be B3-EXP. Go to step 19.

19. Add B-EXP and B3-EXP. Let the result be C1-EXP. Add 126 to C1-EXP to restore the IEEE bias, getting C-EXP. Go to step 20.

20. Round C-HI.C-LO. Go to step 21.

21. Pack C-EXP.M-SIGN, C-HI and C-LO and return.

(vi) FTOA.

The processing steps are as follows.

1. Unpack the input FLP number. Let the unpacked number be represented by C-EXP.C-SIGN, C-HI and C-LO. Go to step 2.

2. Subtract 126-decimal from C-EXP to remove the IEEE bias. Let the result be C1-EXP. Go to step 3.

3. Multiply C1-EXP by the binary representation of log(2). Let the product be U-HI.U-LO. Go to step 4.

4. Subtract 8 from U-HI.U-LO. Let the magitude of the integer part of the result be V and its sign VSIGN. Go to step 5.

5. If VSIGN is 0, then set P-HI.P-LO to the binary representation of 0.8, else set P-HI.P-LO to the binary representation of 0.625. Go to step 6.

6. Multiply C-HI.C-LO by P-HI.P-LO V times. Normalize the partial product after each multiplication, if needed, by left shifting. Accumulate any left shifts needed in B1-EXP. Let the final product be HI-INT.LO-INT. Go to step 7.

7. Subtract B1-EXP from C1-EXP. Let the result be B2-EXP. Go to step 8.

8. If VSIGN is 0, then multiply V by −3, else multiply it by 4. Let the result be B3-EXP. Go to step 9.

9. Add B2-EXP and B3-EXP. Let the result be B4-EXP. Go to step 10.

10. If B4-EXP is more than 32-decimal, then increase V and go to step 6, else go to step 11.

11. If B4-EXP is less than 28-decimal, then decrease V and go to step 6, else go to step 12.

12. Subtract B4-EXP from 32. Let the result be B5-EXP. Go to step 13.

13. Shift HI-INT.LO-INT right B5-EXP number of times. Go to step 14.

14. Add 16-decimal to the address of the start of the decimal string. Output a null byte there. Go to step 15.

15. Divide V by 10-decimal. Let the quotient be Q and the remainder R. Add 30 to R and output it to the decimal string. Next add 30 to Q and output it to the decimal string. Go to step 16.

16. If VSIGN is 0, then output '+' to the output string, else output '−' to the output string. Go to step 17.

17. Output 'E' to the output string. Output '.' to the output string. Go to step 18.

18. Divide C-HI.C-LO by 10-decimal 10 times. Let the remainder in each division be R. Add 30 to each R and output it to the output string. Go to step 19.

19. If C-SIGN is 0, then output the space character to the output string, else output '−' to the output string. Then return to the calling program.

REFERENCES

1. ANSI/IEEE Std 754-1985, IEEE Standard for Binary Floating-Point Arithmetic, IEEE, Aug. 12, 1985.

2. J.T. Coonen, "An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic," IEEE Computer, Jan. 1980, pp. 68–79.

3. K. Hwang, *Computer Arithmetic,* John-Wiley and Sons, 1979.

4. M. M. Mano, *Computer System Design,* Prentice-Hall, 1980.

APPENDIX A

A USER'S MANUAL FOR THE HPC FLOATING POINT PACKAGE

The Single Precision Floating Point Package for the HPC implements the following functions.

ARITHMETIC FUNCTIONS

1. FADD—Add two floating point numbers.
2. FSUB—Subtract two floating point numbers.
3. FMULT—Multiply two floating point numbers.
4. FDIV—Divide two floating point numbers.

FORMAT CONVERSION FUNCTIONS

5. ATOF—Convert an ASCII string representing a decimal floating point number to a single precision floating point number.
6. FTOA—Convert a single precision floating point number to an ASCII string that represents the decimal floating point value of the number.

The entire package is in the form of a collection of subroutines and is contained in the following files.

1. FERR.MAC
2. FNACHK.MAC
3. FZCHK.MAC
4. FUNPAK.MAC
5. FPAK.MAC
6. FPTRAP.MAC
7. ROUND.MAC
8. BFMUL.MAC
9. ISIOK.MAC
10. MUL10.MAC
11. ATOF.MAC
12. FTOA.MAC
13. FADD.MAC
14. FMULT.MAC
15. FDIV.MAC

The first 7 files are general utility routines that are used by all the Arithmetic and Format Conversion subroutines. The next 3 files, BFMUL.MAC, ISIOK.MAC and MUL10.MAC are used only by the Format Conversion subroutines, ATOF and FTOA. Depending on the functions being used in the user program, only the necessary files need be included.

INTERFACE WITH USER PROGRAMS

1. All the Arithmetic routines expect the input to be in the IEEE Single Precision format. This format requires 2 words for the storage of each floating point number. If the required arithmetic operation is FlopF2, where op is +, −, * or /, then the routines expect that F1 is available in registers K and A on entry, with the high half in K. Also, the two words of F2 are expected to be on the stack. If SP is the stack pointer on entry into one of the Arithmetic function subroutines, then the high word of F2 should be at W(SP-4) and the low word at W(SP-6). The result of the Arithmetic operation is returned in IEEE format in registers K and A, with the high word in K.
2. The Format Conversion subroutine ATOF expects that on entry, B contains the address of the ASCII string representing the decimal floating point number. This string must be of the form

   Siiiii.ffffffEsNND

   where

   S is an optional sign for the mantissa. Thus S can be '+', '−' or not present at all.

   iiiii is the optional integer part of the mantissa. If it is present, it can be of any length, must contain only the characters '0' through '9' and must not contain any embedded blanks.

   . is the optional decimal point. It need not be present if the number has no fractional part.

   ffffff is the optional fractional part of the mantissa. ffffff, if it is present must consist of a sequence of digits '0' through '9'. It can be of any length. Note that either iiiii, the integer part or .ffffff the fractional part must be present.

   E is the required exponent start symbol.

   s is the optional sign of the exponent. If it is present, it must be '+' or '−'.

   NN is the exponent and consists of at most two decimal digits. It is required to be present.

   D is the null byte <00> and must be present to terminate the string.

   The floating point number represented by the above string is returned by ATOF in IEEE format in registers K and A.
3. The format conversion routine FTOA expects the floating point number input to be in registers K and A in the IEEE format. Register B is expected to contain the starting address of a 17 byte portion of memory where the output string will be stored.
4. Three global symbols need to be defined in the user program before assembling the user program and any included floating point package files. These symbols are:

   (i) TMP1 which must be set to 0. The package uses W(TMP1) for temporary storage.

   (ii) FPERWD which must be set to an address in the base page. The package signals floating point exceptions using W(FPERWD). This is described below.

   (iii) FPTRAP which must be set to the address of the start of a user floating point exception handler. Again this is described below.

FLOATING POINT EXCEPTS

The package maintains a history of floating point exceptions in the 4 least significant bits of the word W(FPERWD). The value of the symbol FPERWD should be defined by the user program, and should be an address in the base page. This word should also be cleared by the user program before calling any floating point routine. The word is never cleared by the floating point package, and the user program can examine this word to determine the type of exceptions that may have occurred during the course of a computation.

The following 4 types of error can occur in the course of a floating point computation.

1. Invalid Operand. This happens if one of the input numbers for an Arithmetic routine or the input for FTOA is not a valid floating point number. An invalid floating point number (or NaN) can be created either by an overflow in a previous computation step, or if the ASCII decimal floating point number input to ATOF is too large to be represented in the IEEE format. The result, if one of the inputs is a NaN is always set to a NaN.
2. Overflow. This happens if the result of a computation is too large to be represented within the exponent range available. Overflow can occur in any of the arithmetic routines or ATOF. On overflow, the result is set to a representation called NaN. An NaN is considered an illegal operand in all successive steps.
3. Underflow. This occurs if the result of a computation is too small to be represented with the precision and exponent range available. On underflow, the result is set to zero.
4. Divide-by-zero. This error occurs if F2 is zero when computing F1/F2. The result is set to an NaN.

Each of the above errors results in a bit being set in W(FPERWD). This is done as follows:

Bit 0—Set on Overflow.

Bit 1—Set on Underflow.

Bit 2—Set on Illegal Operand.

Bit 3—Set on Divide-by-zero.

One further action is taken when a floating point exception occurs. After the result has been set to the appropriate value, and the corresponding bit in W(FPERWD) set, the package does a subroutine call to address FPTRAP. The user can provide any exception handler at this address. The file FPTRAP.MAC contains the simplest possible user exception handler. It does nothing, but merely returns back to the calling program.

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP

                                            .TITLE FLP
                        LISTER:
        0071                                .LIST 071
        F000                                . = 0F000
        0002                                FPERWD = W(2)
        0000                                TMP1 = W(0)

NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     2
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
THE FLP ROUTINES

                                            .FORM 'THE FLP ROUTINES'
```

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     3
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
FERR.MAC

  8                                      .FORM 'FERR.MAC'
  9                                      .INCLD FERR.MAC
  1                        ; EXCEPTION HANDLING.
  2                        ; DIVIDE BY ZERO.
  3                        DIVBYO:
  4 F000 820802FA                  OR FPERWD, 08    ; SET THE DIVIDE BY 0 BIT.
  5 F004 00                        CLR A
  6 F005 B17F80                    LD K, 07F80
  7 F008 3093                      JSR FPTRAP
  8 F00R 3FCC                      POP B
  9 F00C 3FCE                      POP X
 10 F00E 3C                        RET
 11                        ; ILLEGAL OPERAND - ONE OF F1 OR F2 IS A NAN.
 12                        FNAN:
 13 F00F 820402FA                  OR FPERWD, 04    ; SET THE ILLEGAL OPERAND BIT.
 14 F013 00                        CLR A
 15 F014 B17F80                    LD K, 07F80      ; RETURN NAN IN K AND A.
 16 F017 3084                      JSR FPTRAP       ; GO TO USER TRAP ROUTINE.
 17 F019 3FCC                      POP B
 18 F01B 3FCE                      POP X
 19 F01D 3C                        RET
 20                        ; EXPONENT UNDERFLOW.
 21                        UNDFL:
 22 F01E 820202FA                  OR FPERWD, 02    ; SET THE EXPONENT UNDERFLOW BIT.
 23 F022 00                        CLR A
 24 F023 ACC8CA                    LD K, A
 25 F026 3075                      JSR FPTRAP
 26 F028 3FC4                      POP SP
 27 F02A 3FCC                      POP B
 28 F02C 3FCE                      POP X
 29 F02E 3C                        RET
 30                        ; EXPONENT OVERFLOW.
 31                        OVRFL:
 32 F02F 820102FA                  OR FPERWD, 01    ; SET THE EXPONENT OVERFLOW BIT.
 33 F033 00                        CLR A
 34 F034 B17F80                    LD K, 07F80
 35 F037 3064                      JSR FPTRAP
 36 F039 3FC4                      POP SP
 37 F03B 3FCC                      POP B
 38 F03D 3FCE                      POP X
 39 F03F 3C                        RET
 40                        ;
 41                                .END
```

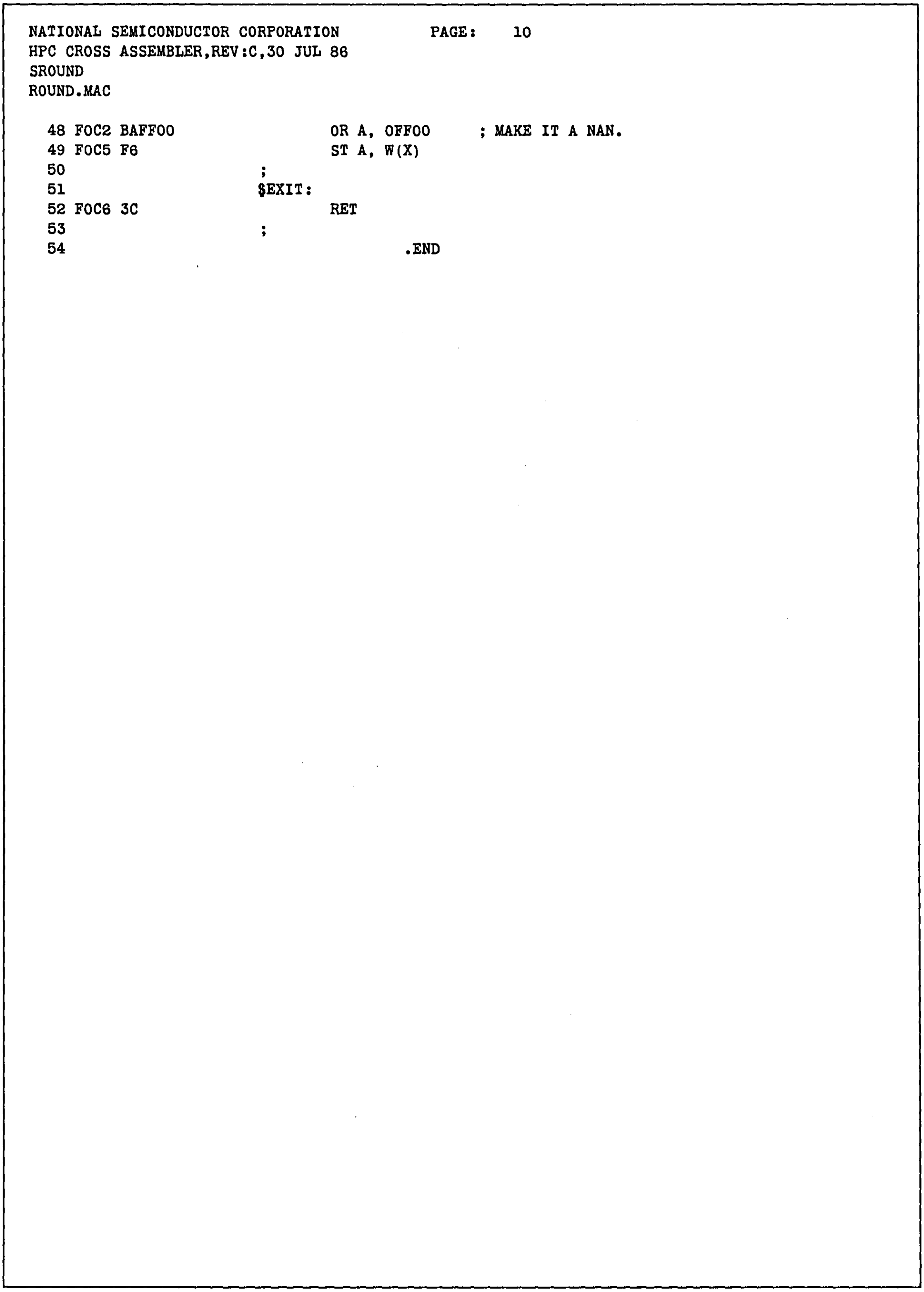
```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FLP
FNACHK.MAC

                                        .FORM 'FNACHK.MAC'
                                        .INCLD FNACHK.MAC
                                        .TITLE FNACHK
                                        .LOCAL
                        ;
                        ; SUBROUTINE TO CHECK IF A SP FLOATING POINT NUMBER STORED IN THE
                        ; IEEE FLOATING POINT FORMAT IN REGS. K AND A IS NAN.
                        ;
                        ; RETURNS 0 IN C IF THE NUMBER IS NOT A NAN.
                        ; RETURNS 1 IN C IF THE NUMBER IS A NAN.
                        ;
                        ; PRESERVES REGS. K, A, X AND B. DESTROYS C.
                        ;
                        FNACHK:
F040 AECA                       X A, K
F042 E7                         SHL A
F043 BDFEFF                     IFGT A,0FEFF
F046 45                         JP $ISNAN
F047 D7                         RRC A
F048 03                         RESET C
F049 AECA                       X A, K
F04B 3C                         RET
                        $ISNAN:
F04C D7                         RRC A
F04D 02                         SET C
F04E AECA                       X A, K
F050 3C                         RET
                        ;
                                        .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FNACHK
FZCHK.MAC

  12                                      .FORM 'FZCHK.MAC'
  13                                      .INCLD FZCHK.MAC
   1                                      .TITLE FZCHK
   2                                      .LOCAL
   3                       ;
   4                       ; SUBROUTINE THAT CHECKS IF A SP FLOATING POINT NUMBER STORED
   5                       ; IN THE IEEE FORMAT IN REGS K AND A IS ZERO.
   6                       ;
   7                       ; RETURNS 0 IN C IF THE NUMBER IS NOT ZERO.
   8                       ; RETURNS 1 IN C IF THE NUMBER IS ZERO.
   9                       ; SAVES REGS. K, A, X, AND B BUT DESTROYS C.
  10                       ;
  11                       FZCHK:
  12 F051 AECA                     X A, K
  13 F053 E7                       SHL A
  14 F054 9DFF                     IFGT A,OFF
  15 F056 45                       JP $ANOTO
  16 F057 D7                       RRC A
  17 F058 02                       SET C
  18 F059 AECA                     X A, K
  19 F05B 3C                       RET
  20                       $ANOTO:
  21 F05C D7                       RRC A
  22 F05D 03                       RESET C
  23 F05E AECA                     X A, K
  24 F060 3C                       RET
  25                       ;
  26                                      .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FZCHK
FUNPAK.MAC

                                         .FORM 'FUNPAK.MAC'
                                         .INCLD FUNPAK.MAC
                                         .TITLE FUNPAK
                                         .LOCAL
                         ;
                         ; SUBROUTINE TO UNPACK A SP FLOATING POINT NUMBER STORED IN THE
                         ; IEEE FORMAT IN REGS. K AND A. THE UNPACKED FORMAT OCCUPIES 3
                         ; WORDS AND IS ORGANIZED AS FOLLOWS:
                         ;                       |                |
                         ; increasing addrs |    |                |             <- X on exit
                         ;                       |EEEEEEEESSSSSSSS| FEXP-FSIGN
                         ;                       |MMMMMMMMMMMMMMMM| FHI
                         ;                       |MMMMMMMM00000000| FLO         <- X on entry
                         ;                       |                |
                         ;
                         ; EEEEEEEE - 8 BIT EXPONENT IN EXCESS-127 FORMAT
                         ; SSSSSSSS - SIGN BIT < 00 -> +, FF -> ->
                         ; M ... M - 24 BITS OF MANTISSA. NOTE THAT IMPLIED 1 IS PRESENT HERE.
                         ;
                         ; ON ENTRY TO THE SUBROUTINE X SHOULD POINT TO FLO. ON EXIT, X POINTS
                         ; TO THE WORD AFTER FSIGN.
                         ; REGS. K, A AND B ARE DESTROYED BY THIS SUBROUTINE.
                         ;
                         FUNPAK:
     F061 ABCC                   ST A,B          ; SAVE A IN B.
     F063 00                     CLR A
     F064 D1                     X A, M(X+)      ; ZERO LOW BYTE OF FLO.
     F065 88CC                   LD A, L(B)
     F067 D1                     X A, M(X+)      ; MOVE LOW BYTE OF F-RO INTO HIGH BYTE OF FLO.
     F068 88CD                   LD A, H(B)
     F06A D1                     X A, M(X+)      ; MOVE MID BYTE OF MANT INTO LOW BYTE OF FHI.
     F06B A8CA                   LD A, K
     F06D 96C80F                 SET A.7         ; SET IMPLIED 1 IN MANTISSA
     F070 D1                     X A, M(X+)      ; MOVE HIGH BYTE OF MANT INTO HIGH BYTE OF FHI.
     F071 A8CA                   LD A, K
     F073 E7                     SHL A           ; SIGN BIT TO CARRY.
     F074 B9FF00                 AND A, 0FF00    ; ZERO SIGN.
     F077 07                     IF C
     F078 9AFF                   OR A, 0FF       ; PUT SIGN BACK IF -.
     F07A F1                     X A, W(X+)      ; SAVE FEXP-FSIGN.
     F07B 3C                     RET
                         ;
                                         .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FUNPAK
FPAK.MAC

                                    .FORM 'FPAK.MAC'
                                    .INCLD FPAK.MAC
                                    .TITLE FPAK
                                    .LOCAL
                    ;
                    ; SUBROUTINE TO PACK A SP FLOATING POINT NUMBER STORED IN THE
                    ; 3 WORD FEXP-FSIGN/FHI/FLO FORMAT INTO THE IEEE FORMAT IN REGS.
                    ; K AND A.
                    ;
                    ; ON ENTRY TO THE SUBROUTINE, X POINTS TO FLO. ON EXIT, X POINTS
                    ; TO THE WORD AFTER FSIGN.
                    ;
                    ; REGS. K, A AND B ARE DESTROYED.
                    ;
                    FPAK:
F07C D1                     X A, M(X+)      ; GET RID OF ZERO LOW BYTE OF FLO.
F07D D1                     X A, M(X+)      ; GET HIGH BYTE OF FLO.
F07E ABCA                   ST A, K         ; STORE IT IN K.
F080 D1                     X A, M(X+)      ; GET LOW BYTE OF FHI.
F081 3B                     SWAP A
F082 3B                     SWAP A
F083 B9FF00                 AND A, 0FF00    ; SHIFT LEFT 8 TIMES.
F086 A0C8CAFA               OR K, A         ; LOW WORD OF RESULT IS NOW IN K.
                    ;
F08A D1                     X A, M(X+)      ; GET HIGH BYTE OF FHI.
F08B 96C81F                 RESET A.7       ; ZERO IMPLIED MSB 1 IN MANT.
F08E ABCC                   ST A,B          ; SAVE IN REG. B.
F090 D4                     LD A, M(X)      ; GET SIGN BYTE FROM ASIGN.
F091 C7                     SHR A           ; MOVE 1 SIGN BIT INTO CARRY.
F092 F0                     LD A, W(X+)     ; GET FEXP-FSIGN.
F093 B9FF00                 AND A, 0FF00    ; ZERO SIGN.
F096 D7                     RRC A           ; MOVE RIGHT 1 BIT. SIGN BIT FROM C
                                            ; ENTERS INTO THE MSB.
F097 96CCFA                 OR A, B         ; GET MANT BITS IN FROM B.
F09A AECA                   X A, K          ; SWAP A AND K
F09C 3C                     RET
                    ;
                                    .END


NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     8
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPAK
FPTRAP.MAC

                                    .FORM 'FPTRAP.MAC'
                                    .INCLD FPTRAP.MAC
                                    .TITLE FPTRAP
                    ; USER SUPPLIED FP TRAP ROUTINE.
                    FPTRAP:
F09D 3C                     RET
                    ;
                                    .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     9
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FPTRAP
ROUND.MAC

                                    .FORM 'ROUND.MAC'
                                    .INCLD ROUND.MAC
                                    .TITLE SROUND
                                    .LOCAL
                  ;
                  ; THIS SUBROUTINE IS USED TO ROUND THE 32 BIT MANTISSA OBTAINED
                  ; IN THE FLOATING POINT CALCULATIONS TO 24 BITS.
                  ;
                  ; THE UNPACKED FLOATING POINT NUMBER SHOULD BE STORED IN
                  ; CONSECUTIVE WORDS OF MEMORY. ON ENTRY, X SHOULD CONTAIN
                  ; THE ADDRESS OF C-HI. C-EXP.C-SIGN IS AT W(X+2) AND
                  ; C-LO IS AT W(X-2).
                  ;
                  ; ON EXIT X HAS THE ADDRESS OF C-EXP.C-SIGN.
                  SROUND:
F09E F2                     LD A, W(X-)       ; REMEMBER X POINTS TO C-HI.
F09F F4                     LDA, W(X)         ; LOAD C-LO.
F0A0 96C817                 IF A.7            ; IF BIT 25 OF MANTISSA IS 1,
F0A3 43                     JP $RNDUP         ; THEN NEED TO INCREASE MANTISSA.
F0A4 F0                     LD A, W(X+)
F0A5 F0                     LD A, W(X+)       ; X NOW POINTS TO C-EXP.C-SIGN.
F0A6 5F                     JP $EXIT          ; DONE, SO GET OUT.
                  ; INCREASE MANTISSA.
                  $RNDUP:
F0A7 B80100                 ADD A, 0100
F0AA F1                     X A, W(X+)        ; INCREASE LOW BYTE BY 1.
F0AB 07                     IF C              ; IF THERE IS A CARRY,
F0AC 42                     JP $HIUP          ; THEN NEED TO INCREASE C-HI.
F0AD F0                     LD A, W(X+)       ; X NOW POINTS TO C-EXP.C-SIGN.
F0AE 57                     JP $EXIT          ; DONE, SO GET OUT.
                  ; MANTISSA INCREASE PROPAGATING TO HIGH WORD.
                  $HIUP:
F0AF F4                     LD A, W(X)
F0B0 B8                     .BYTE 0B8,00,01 ; DO ADD A, 01 BUT WITH WORD CARRY!!
F0B1 00
F0B2 01
F0B3 07                     IF C              ; IF THERE IS A CARRY,
F0B4 42                     JP $EXIN2         ; THEN NEED TO INCREASE EXPONENT.
F0B5 F1                     X A,W(X+)
F0B6 4F                     JP $EXIT          ; GET OUT.
                  ; ROUND UP LEADS TO EXPONENT INCREASE.
                  $EXIN2:
F0B7 D7                     RRC A             ; CARRY->MSB, LSB->CARRY.
F0B8 F3                     X A,W(X-)
F0B9 F4                     LD A,W(X)         ; LOW WORD IS NOW IN A.
F0BA D7                     RRC A
F0BB F1                     X A, W(X+)
F0BC F0                     LD A, W(X+)       ; X NOW POINTS TO C-EXP.CSIGN.
F0BD F4                     LD A, W(X)
F0BE B80100                 ADD A, 0100
F0C1 07                     IF C
```



```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    10
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
ROUND.MAC

  48 F0C2 BAFF00                OR A, 0FF00     ; MAKE IT A NAN.
  49 F0C5 F6                    ST A, W(X)
  50                    ;
  51                    $EXIT:
  52 F0C6 3C                    RET
  53                    ;
  54                                   .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    11
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SROUND
BFMUL.MAC

 22                               .FORM 'BFMUL.MAC'
 23                               .INCLD BFMUL.MAC
  1                               .TITLE BFMUL
  2                     ;
  3                     ; THIS SUBROUTINE IS USED TO MULTIPLY TWO 32 BIT FIXED POINT FRACTIONS.
  4                     ; THE ASSUMED BINARY POINT IS TO THE IMMEDIATE LEFT OF THE MSB.
  5                     ;
  6                     ; THE FIRST FRACTION IS STORED IN REGS K AND A, WITH THE MORE
  7                     ; SIGNIFICANT WORD BEING IN K.
  8                     ;
  9                     ; THE SECOND FRACTION IS STORED ON THE STACK. THE MORE SIGNIFICANT
 10                     ; WORD IS AT W(SP-4) AND THE LOWER SIGNIFICANT WORD
 11                     ; IS IN THE WORD BELOW IT.
 12                     ;
 13                     ; THE 32 BIT PRODUCT IS LEFT IN REGS. K AND A, WITH THE MORE
 14                     ; SIGNIFICANT WORD BEING IN K.
 15                     ;
 16                     ; IMPORTANT NOTE : THE FRACTIONS ARE ASSUMED TO BE UNSIGNED.
 17                     ;
 18                     ; REGS. B AND X ARE UNCHANGED.
 19                     ;
 20                     BFMUL:
 21 F0C7 AFCE                   PUSH X          ; SAVE X.
 22 F0C9 AFC8                   PUSH A          ; SAVE F1-LO
 23 F0CB AFCA                   PUSH K          ; SAVE F1-HI.
 24 F0CD ABCA                   LD A, K         ; MOVE F1-HI TO A.
 25 F0CF A6FFF6C4FE             MULT A, W(SP-0A); MULTIPLY F1-HI BY F2-HI.
 26 F0D4 3FCA                   POP K           ; GET FI-HI.
 27 F0D6 AFCE                   PUSH X          ; SAVE PR-HI.
 28 F0D8 AFC8                   PUSH A          ; SAVE PR-LO.
 29 F0DA A8CA                   LD A, K         ; MOVE F1-HI TO A.
 30 F0DC A6FFF2C4FE             MULT A, W(SP-0E); MULTIPLY F1-HI BY F2-LO.
 31 F0E1 3FC8                   POP A           ; GET PR-LO SAVED. NOTE THAT THE
 32                                             ; LO WORD OF THIS PRODUCT IS DISCARDED.
 33 F0E3 3FCA                   POP K           ; GET PR-HI SAVED.
 34 F0E5 96CEF8                 ADD A, X        ; ADD TO PR-LO THE HI WORD OF THIS PRODUCT.
 35 F0E8 07                     IF C            ; ON CARRY,
 36 F0E9 A9CA                   INC K           ; PROPAGATE THRU TO PR-HI.
 37 F0EB 3FCE                   POP X           ; GET F1-LO.
 38 F0ED AFCA                   PUSH K          ; SAVE PR-HI.
 39 F0EF AFC8                   PUSH A          ; SAVE PR-LO.
 40 F0F1 A8CE                   LD A, X         ; MOVE F1-LO TO A.
 41 F0F3 A6FFF6C4FE             MULT A, W(SP-0A); MULTIPLY BY F2-HI.
 42 F0F8 3FC8                   POP A           ; GET PR-LO SAVED.
 43 F0FA 3FCA                   POP K           ; GET PR-HI SAVED.
 44 F0FC 96CEF8                 ADD A, X        ; ADD TO PR-LO THE HI-WORD OF THIS PRODUCT.
 45 F0FF 07                     IF C
 46 F100 A9CA                   INC K           ; PROPAGATE ANY CARRY TO PR-HI.
 47 F102 3FCE                   POP X           ; RESTORE X.
 48 F104 3C                     RET
 49                     ;
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    12
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
BFMUL
BFMUL.MAC

  50                                       .END



NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    13
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
BFMUL
ISIOK.MAC

  24                                       .FORM 'ISIOK.MAC'
  25                                       .INCLD ISIOK.MAC
   1                                       .TITLE ISIOK
   2                                       .LOCAL
   3                       ;
   4                       ; THIS SUBROUTINE IS USED TO DETERMINE IF ANOTHER DECIMAL DIGIT CAN
   5                       ; BE ACCUMULATED IN THE 32 BIT INTEGER STORED IN REGS. K AND A.
   6                       ; THE MORE SIGNIFICANT WORD IS IN K.
   7                       ; SETS THE CARRY TO 1 IF IT CAN BE ACCUMULATED; RESETS THE CARRY
   8                       ; OTHERWISE. PRESERVES ALL REGS.
   9                       ;
  10                       ISIOK:
  11 F105 02                       SET C
  12 F106 861999CAFC               IFEQ K, 01999
  13 F10B 47                       JP $CHKOT
  14 F10C 861999CAFD               IFGT K, 01999
  15 F111 03                       RESET C
  16 F112 3C                       RET
  17 F113 BD9998           $CHKOT: IFGT A, 09998
  18 F116 03                       RESET C
  19 F117 3C                       RET
  20                       ;
  21                                       .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    14
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ISIOK
MUL10.MAC

  26                                    .FORM 'MUL10.MAC'
  27                                    .INCLD MUL10.MAC
   1                                    .TITLE MUL10
   2                                    .LOCAL
   3                        ;
   4                        ; THIS SUBROUTINE MULTIPLIES THE 32 BIT INTEGER STORED IN REGS K AND A
   5                        ; BY 10-DECIMAL AND ADDS TO IT THE INTEGER STORED IN X.
   6                        ; THE RESULT IS RETURNED IN K AND A.
   7                        ; REGS. B AND X ARE NOT CHANGED.
   8                        ;
   9                        MUL10:
  10 F118 AFCE                      PUSH X           ; SAVE INTEGER.
  11 F11A AFC8                      PUSH A           ; SAVE LONG INT-LO.
  12 F11C A8CA                      LD A, K
  13 F11E 9E0A                      MULT A, 0A       ; MULT LONG INT-HI BY 10.
  14 F120 AFC8                      PUSH A           ; SAVE LOW WORD OF PRODUCT.
  15 F122 A6FFFCC4A8                LD A, W(SP-4)    ; GET LONG INT-LO.
  16 F127 9E0A                      MULT A, 0A
  17 F129 3FCA                      POP K            ; GET LO WORD OF LAST PRODUCT.
  18 F12B A0CECAF8                  ADD K, X         ; ADD TO IT HI WORD OF THIS PRODUCT.
  19 F12F 3FCE                      POP X            ; GET RID OF GARBAGE.
  20 F131 3FCE                      POP X            ; GET INTEGER TO BE ADDED.
  21 F133 96CEF8                    ADD A, X
  22 F136 07                        IF C
  23 F137 A9CA                      INC K
  24 F139 3C                        RET
  25                        ;
  26                                    .END
  28                        ;
```

```
                                .FORM 'ATOF.MAC'
                                .INCLD ATOF.MAC
                                .TITLE ATOF
                                .LOCAL
                ;
                ; THIS SUBROUTINE CONVERTS A DECIMAL FLOATING POINT STRING TO
                ; AN IEEE FORMAT SINGLE PRECISION FLOATING POINT NUMBER. THE
                ; INPUT DECIMAL STRING IS ASSUMED TO BE OF THE FORM
                ;               SMMMMMMM.FFFFFEDNN
                ; WHERE S IS THE SIGN OF THE DECIMAL MANTISSA,
                ;       M...M IS THE INTEGER PART OF THE MANTISSA,
                ;       F...F IS THE FRACTIONAL PART OF THE MANTISSA,
                ;       D IS THE SIGN OF THE DECIMAL EXPONENT,
                ; AND   NNN IS THE DECIMAL EXPONENT.
                ;
                ;       ON ENTRY, B SHOULD POINT TO THE ADDRESS OF THE ASCII
                ; STRING HOLDING THE DECIMAL FLOATING POINT NUMBER. THIS STRING
                ; MUST BE TERMINATED BY A NULL BYTE.
                ;
                ;       THE BINARY FLOATING POINT NUMBER IS RETURNED IN
                ; REGS. K AND A.
                ;
                ; REGS. B AND X ARE LEFT UNCHANGED.
                ;
                ;
                ;
                ;
                ;
                ATOF:
F13A AFCE               PUSH X
F13C AFCC               PUSH B
F13E 00                 CLR A           ; ZERO A.
F13F AFC8               PUSH A          ; STORAGE FOR MANTISSA SIGN.
F141 AFC8               PUSH A          ; STORAGE FOR IMPLICIT 10'S EXPONENT.
F143 AFC8               PUSH A          ; STORAGE FOR HI-INT.
F145 AFC8               PUSH A          ; STORAGE FOR LO-INT.
                ;
                ; DECIMAL STRING MUST START WITH A '+', '-', '.' OR A DIGIT.
                ; RESULTS ARE UNPREDICTABLE IF IT DOES NOT.
                ; THE '+' MEANS THAT THE MANTISSA IS POSITIVE. IT CAN BE OMITTED.
                ; THE '-' MEANS THAT THE MANTISSA IS NEGATIVE.
                ; THE '.' MEANS THAT THE MANTISSA HAS NO INTEGER PART.
                ;
                $LOOP1:
F147 C0                 LDS A, M(B+)
F148 40                 NOP             ; GET THE CHARACTER.
F149 9C2B               IFEQ A, '+'     ; IF IT IS A '+',
F14B 64                 JP $LOOP1       ; DO NOTHING, BUT GET 1 MORE.
F14C 9C2D               IFEQ A, '-'     ; IF IT IS A '-',
F14E 45                 JP $MSIGN       ; GO AND CHANGE THE MANTISSA SIGN.
F14F 9C2E               IFEQ A, '.'     ; IF IT IS A '.',
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    16
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

F151 9438                  JMP $FRCOL      ; GO AND COLLECT THE FRACTION PART.
                                           ; GET HERE MEANS IT IS A DIGIT.
F153 48                    JP $INCOL       ; SO GO AND COLLECT THE INTEGER PART.
                   $MSIGN:
F154 90FF                  LD A, OFF
F156 A6FFF8C4AB            ST A, W(SP-08)  ; CHANGE MANTISSA SIGN TO NEG.
F15B 74                    JP $LOOP1       ; GO BACK FOR SOME MORE.
                   ;
                   $INCOL:
                   ; GET HERE MEANS COLLECTING INTEGER PART OF MANTISSA.
                   ;
F15C 02                    SET C
F15D 8230C8EB              SUBC A, '0'     ; CONVERT DIGIT FROM ASCII TO INTEGER.
F161 ACC8CE                LD X, A         ; MOVE INTEGER TO X.
F164 3FCA                  POP K           ; GET HI-INT COLLECTED SO FAR.
F166 3FC8                  POP A           ; GET LO-INT COLLECTED SO FAR.
F168 3463                  JSR ISIOK       ; CHECK IF THE DIGIT CAN BE ACCUMULATED.
F16A 07                    IF C            ; LOOK AT C.
F16B 4B                    JP $ACCM        ; YES, IT CAN BE SO GO DO IT.
                                           ; GET HERE MEANS CAN ACCUMULATE ANY MORE.
                                           ; SO INCREASE THE IMPLICIT 10'S EXPONENT.
F16C 3FCE                  POP X           ; GET IMPLICIT 10'S EXPONENT COLLECTED
F16E A9CE                  INC X           ; SO FAR AND INCREMENT IT.
F170 AFCE                  PUSH X          ; SAVE IT BACK.
F172 AFC8                  PUSH A          ; SAVE LO-INT.
F174 AFCA                  PUSH K          ; SAVE HI-INT.
F176 46                    JP $ISNXT
                   ;
                   $ACCM:
                   ; GET HERE MEANS THE PRESENT DIGIT CAN BE ACCUMULATED.
F177 345F                  JSR MUL10       ; MULTIPLY BY 10 AND ADD DIGIT.
F179 AFC8                  PUSH A          ; SAVE LO-INT.
F17B AFCA                  PUSH K          ; SAVE HI-INT.
                   $ISNXT:
                   ; PROCESS THE NEXT CHARACTER.
F17D C0                    LDS A, M(B+)
F17E 40                    NOP
F17F 9C2E                  IFEQ A, '.'     ; IF IT IS A '.'
F181 49                    JP $FRCOL       ; GO COLLECT FRACTION PART.
F182 9C45                  IFEQ A, 'E'     ; IF IT IS 'E',
F184 9434                  JMP $EXCOL      ; GO COLLECT EXPONENT PART.
F186 9C20                  IFEQ A, ' '     ; IF IT IS A SPACE,
F188 6B                    JP $ISNXT       ; GO GET SOME MORE.
                                           ; GET HERE MEANS IT IS A DIGIT.
F189 952D                  JMP $INCOL
                   ;
                   $FRCOL:
                   ; GET HERE MEANS COLLECT THE FRACTIONAL PART OF THE MANTISSA.
                   ;
F18B C0                    LDS A, M(B+)
F18C 40                    NOP
```

```
F18D 9C45                    IFEQ A, 'E'     ; IF IT IS A 'E',
F18F 9429                    JMP $EXCOL      ; GO COLLECT EXPONENT.
F191 9C20                    IFEQ A, ' '     ; IF IT IS SPACE,
F193 68                      JP $FRCOL       ; GO GET SOME MORE.
                                             ; GET HERE MEANS IT IS A DIGIT.
F194 D2                      SET C
F195 8230C8EB                SUBC A, '0'     ; GET INTEGER FROM DIGIT.
F199 ACC8CE                  LD X, A         ; SAVE IT IN A.
F19C 3FCA                    POP K           ; GET HI-INT.
F19E EFC8                    POP A           ; GET LO-INT.
F1A0 349B                    JSR ISIOK       ; CHECK IF IT CAN BE ACCUMULATED.
F1A2 07                      IF C
F1A3 45                      JP $ACCF        ; YES, SO GO DO IT.
                                             ; GET HERE MEANS CAN'T COLLECT MORE DIGITS.
F1A4 AFC8                    PUSH A
F1A6 AFCA                    PUSH K
F1A8 7D                      JP $FRCOL       ; SO JUST IGNORE IT.
                     ;
                     $ACCF:
                     ; ACCUMULATE THE FRACTIONAL DIGIT.
F1A9 3491                    JSR MUL10       ; MULTIPLY BY 10 AND ADD DIGIT.
F1AB 3FCE                    POP X           ; GET IMPLICIT 10'S EXPONENT COLLECTED SO FAR,
F1AD 86FFFFCEF8              ADD X, 0FFFF    ; AND DECREMENT IT BY 1.
F1B2 AFCE                    PUSH X          ; SAVE IT BACK.
F1B4 AFC8                    PUSH A          ; SAVE LO-INT.
F1B6 AFCA                    PUSH K          ; SAVE HI-INT.
F1B8 952D                    JMP $FRCOL      ; GO GET SOME MORE.
                     ;
                     $EXCOL:
                     ; GET HERE MEANS THE EXPLICIT 10'S EXPONENT NEEDS TO BE
                     ; COLLECTED FROM THE STRING.
F1BA 03                      RESET C         ; MAKE EXPONENT SIGN POST.
                     $EXCHR:
F1BB C0                      LDS A, M(B+)
F1BC 40                      MOP
F1BD 9C2B                    IFEQ A, '+'     ; IF IT IS A '+',
F1BF 64                      JP $EXCHR       ; GET SOME MORE.
F1C0 9C2D                    IFEQ A, '-',    ; IF IT IS A '-',
F1C2 44                      JP $ESIGN       ; GO FIX EXPONENT SIGN.
F1C3 9C20                    IFEQ A, ' '     ; IF IT IS SPACE,
F1C5 6A                      JP $EXCHR       ; GO GET SOME MORE.
                                             ; GET HERE MEANS IT IS A DIGIT.
F1C6 42                      JP $EXACC       ; SO GO COLLECT THE EXPONENT.
                     $ESIGN:
F1C7 02                      SET C
F1C8 6D                      JP $EXCHR
                     ;
                     $EXACC:
                     ; ACCUMULATE THE EXPLICIT 10'S EXPONENT.
F1C9 9100                    LD K, 0
F1CB 07                      IF C
```

```
152 F1CC 91FF                   LD K, OFF        ; GET SIGN BITS SET.
153 F1CE AFCA                   PUSH K           ; SAVE EXPLICIT EXPONENTS SIGN.
154 F1D0 9300                   LD X, 0
155 F1D2 AFCE                   PUSH X           ; ZERO EXPLICIT EXPONENT COLLECTED SO FAR.
156                     $EXCLP:
157 F1D4 02                     SET C
158 F1D5 8230C8EB               SUBC A, '0'      ; GET INTEGER FROM ASCII DIGIT.
159 F1D9 ACC8CE                 LD X, A
160 F1DC 3FC8                   POP A            ; GET EXPLICIT EXPONENT COLLECTED SO FAR.
161 F1DE A0C8CEF8               ADD X, A         ; X CONTAINS DIGIT + EXP.
162 F1E2 A0C8CEF8               ADD X, A         ; X CONTAINS DIGIT + 2*EXP.
163 F1E6 E7                     SHL A
164 F1E7 E7                     SHL A
165 F1E8 E7                     SHL A            ; A CONTAINS 8*EXP.
166 F1E9 96CEF8                 ADD A, X         ; A CONTAINS DIGIT + 10*EXP.
167 F1EC AFC8                   PUSH A           ; SAVE BACK ON STACK.
168 F1EE C0                     LDS A, M(B+)
169 F1EF 40                     NOP              ; GET NEXT CHAR.
170 F1F0 9C00                   IFEQ A, 0        ; IS IT A NULL ?
171 F1F2 41                     JP $A10EX        ; YES SO ADD EXPLICIT AND IMPLICIT
172                                              ; 10'S EXPONENT.
173                                              ; GET HERE MEANS IT IS A DIGIT,
174 F1F3 7F                     JP $EXCLP        ; SO GO BACK AND ACCUMULATE IT.
175                     ;
176                     $A10EX:
177                     ; DONE COLLECTING DIGITS. ADD THE EXPLICIT AND IMPLICIT
178                     ; 10'S EXPONENT COLLECTED SO FAR.
179 F1F4 3FC8                   POP A            ; GET EXPLICIT 10'S EXPONENT.
180 F1F6 3FCA                   POP K            ; GET ITS SIGN.
181 F1F8 8200CAFC               IFEQ K, 0        ; IS IT POSITIVE ?
182 F1FC 42                     JP $ADDEX        ; YES SO ADD 'EM.
183 F1FD 01                     COMP A
184 F1FE 04                     INC A            ; CHANGE TO 2'S COM.
185                     $ADDEX:
186 F1FF AGFFFAC4F8             ADD A, W(SP-06)  ; ADD IMPLICIT EXPONENT.
187 F204 BD7FFF                 IFGT A, 07FFF    ; IS IT NEGATIVE ?
188 F207 43                     JP $NEG10        ; YES, CHANGE IT.
189 F208 9300                   LD X, 0          ; LOAD POST. SIGN IN X.
190 F20A 44                     JP $ESAVE
191                     $NEG10
192 F20B 93FF                   LD, X, OFF       ; LOAD NEG. SIGN IN X.
193 F20D 01                     COMP A
194 F20E 04                     INC A            ; MAKE IT POSITIVE.
195                     $ESAVE:
196 F20F ACC8CC                 LD B, A          ; SAVE 10'S EXPONENT IN A.
197 F212 3FC8                   POP A            ; GET HI-INT.
198 F214 3FCA                   POP K            ; GET LO-INT.
199 F216 AFCE                   PUSH X           ; SAVE SIGN OF 10'S EXPONENT.
200 F218 AFCC                   PUSH B           ; AND ITS VALUE.
201                     ;
202                     ; NOW CONVERT HI-INT.LO-INT TO A NORMALIZED FLOATING POINT
```

```
                     ; NUMBER. THE BINARY EXPONENT IS COLLECTED IN B.
                     ;
F21A 9000                    IFGT A, 0       ; IF HI-INT IS NOT 0,
F21C 58                      JP $NORM2       ; NEED TO SHIFT K AND A.
F21D 8200CAFD                IFGT K, 0       ; IF HI-INT IS 0, BUT NOT LO-INT,
F221 4E                      JP $NORM1       ; NEED TO SHIFT ONLY K.
                                             ; GET HERE MEANS MANTISSA IS 0.
F222 00                      CLR A
F223 ACC8CA                  LD K, A
F226 02                      SET C
F227 8208C4EB                SUB SP, 08      ; ADJUST SP. DONE!!!
F22B 3FCC                    POP B
F22D 3FCE                    POP X
F22F 3C                      RET
                     ;
                     $NORM1:
                     ; HI-INT IS 0, SO WORK WITH LO-INT ONLY.
F230 AECA                    X A, K
F232 9210                    LD B, 010       ; LOAD 16 INTO EXPONENT COUNTER.
F234 42                      JP $NRLUP
                     ;
                     $NORM2:
                     ; HI-INT IS NOT 0, SO NEED TO HANDLE BOTH.
F235 9220                    LD B, 020       ; LOAD 32 INTO LOOP COUNTER.
                     $NRLUP:
F237 E7                      SHL A
F238 07                      IF C            ; DID A 1 COME OUT ?
F239 4D                      JP $NRDUN       ; YES IT IS NORMALIZED NOW.
F23A AECA                    X A, K
F23C E7                      SHL A
F23D 07                      IF C
F23E 96CA08                  SET K.0
F241 AECA                    X A, K
F243 AACC                    DECSZ B
F245 40                      NOP             ; SHOULD NEVER BE SKIPPED!!
F246 6F                      JP $NRLUP
                     $NRDUN:
F247 D7                      RRC A           ; RESTORE SHIFTED 1.
F248 AB00                    ST A, TMP1      ; STORE IN W(0).
F24A 3FCE                    POP X           ; GET 10'S EXPONENT.
F24C 3FC8                    POP A           ; GET 10'S EXPONENT SIGN.
F24E AE00                    X A, TMP1       ; A IS HI-INT ONCE MORE.
F250 AFCC                    PUSH B          ; SAVE BINARY EXPONENT.
F252 AFCE                    PUSH X          ; SAVE 10'S EXPONENT.
F254 AECA                    X A, K          ; HI-INT TO K, LO-INT TO A.
F256 960010                  IF TMP1.0       ; IS 10'S EXPONENT NEGATIVE ?
F259 58                      JP $DIV10       ; YES, GO TO DIVIDE BY 10.
                     ; GET HERE MEANS 10'S EXPONENT IS POSITIVE, SO MULTIPLY BY 10.
                     ; ACTUALLY, WHAT IS USED IS
                     ;       10^N = (0.625*(2^4)^N
                     ; SO MULTIPLY BY 0.625 NOW AND TAKE CARE OF 2^(4*N) LATER.
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    20
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

F25A A4F26ACCAB                      LD B, W($MTLO)
F25F AFCC                            PUSH B           ; SAVE LO WORD OF 0.625 ON STACK.
F261 A4F26CCCAB                      LD B, W($MTHI)
F266 AFCC                            PUSH B           ; SAVE HI WORD OF 0.625 ON STACK.
F268 57                              JP $JAMIT        ; GO TO ROUTINE THAT JAMS 0.625^N
                                                      ; BY REPEATED MULTIPLICATION INTO HI-INT.LO-INT.
                             ;
                             ; DEFINE SOME CONSTANTS.
F269 40                                      . EVEN   ; FORCE EVEN ADDRESS.
F26A 0000                    $MTLO:  .WORD 0
F26C 00A0                    $MTHI:  .WORD 0A000
F26E CDCC                    $DTLO:  .WORD 0CCCD
F270 CCCC                    $DTHI:  .WORD 0CCCC
                             ;
                             $DIV10:
                             ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO DIVIDE BY 10.
                             ; ACTUALLY WHAT IS DONE IS
                             ;       10^(-N) = ((2^3)/(0.8))^(-N) = ((0.8)^N)*(2^(-3*N)))
                             ; SO MULTIPLY BY 0.8 NOW AND TAKE CARE OF 2^(-3*N) LATER.
F272 A4F26ECCAB                      LD B, W($DTLO)
F277 AFCC                            PUSH B           ; SAVE LO WORD OF .8
F279 A4F270CCAB                      LD B, W($DTHI)
F27E AFCC                            PUSH B           ; SAVE HI WORD OF .8
                             ;
                             $JAMIT:
                             ; JAM IN THE MULTIPLICATION PART NEEDED TO HANDLE THE 10'S EXP.
F280 9200                            LD B, 0          ; B IS USED TO TRACK ANY BINARY POWERS
                                                      ; THAT COME UP DURING NORMALIZATION.
F282 8200CEFC                        IFEQ X, 0        ; IS 10'S EXPONENT 0 ?
F286 57                              JP $JAMDN        ; YES, DONE ALREADY.
                             $JAMLP:
F287 35C0                            JSR BFMUL        ; MULTIPLY USING 32 BIT UNSIGNED.
F289 AECA                            X A, K           ; SWAP HI AND LO WORDS.
F28B E7                              SHL A
F28C 07                              IF C             ; IS THERE A CARRY ?
F28D 4A                              JP $ISNED        ; YES, SO IT IS ALREADY NORMALIZED.
F28E A9CC                            INC B            ; NEED TO SHIFT LEFT TO NORMALIZE, SO
                                                      ; INCREASE B BY 1.
F290 AECA                            X A, K
F292 E7                              SHL A
F293 07                              IS C
F294 96CA08                          SET K.0
F297 43                              JP $OVR1
                             $ISNED:
F298 D7                              RRC A
F299 AECA                            X A, K
                             $OVR1:
F29B AACE                            DECSZ X          ; DONE YET ?
F29D 76                              JP $JAMLP        ; NO SO DO IT ONCE MORE.
                             ; GET HERE MEANS MULTIPLICATIONS HAVE BEEN DONE. NOW TAKE
                             ; CARE OF THE EXPONENTS.
```

```
                        $JAMDN:
F29E 3FCE                       POP X
F2A0 3FCE                       POP X             ; GET 0.625 OR 0.8 OFF THE STACK.
F2A2 3FCE                       POP X             ; GET THE 10'S EXPONENT.
F2A4 AFC8                       PUSH A            ; SAVE LO WORD OF FLP NUMBER.
F2A6 AFCA                       PUSH K            ; SAVE HI WORD OF FLP NUMBER.
F2A8 A6FFFAC4A8                 LD A, W(SP-6)     ; GET THE BINARY EXPONENT THAT WAS SAVED.
F2AD 02                         SET C
F2AE 96CCEB                     SUBC A, B         ; SUBTRACT FROM IT BINARY EXPONENT COLLECTED
                                                  ; DURING THE JAMMING.
F2B1 ACC8CC                     LD B, A           ; SAVE IT IN B.
F2B4 A8CE                       LD A, X           ; MOVE THE 10'S EXPONENT TO A.
F2B6 960010                     IF TMP1.0         ; IS THE 10'S EXPONENT NEGATIVE ?
F2B9 49                         JP $NAGAS         ; YES, SO GOT TO SUBTRACT.
                                                  ; GET HERE MEANS 10'S EXPONENT IS
                                                  ; POSITIVE, SO MUL IT BY 4.
F2BA E7                         SHL A             ; MULTIPLY BY 2.
F2BB E7                         SHL A             ; MULTIPLY BY 2 AGAIN.
F2BC 96CCF8                     ADD A, B          ; GET THE BINARY EXPONENT IN ALSO.
F2BF B8007E                     ADD A, 07E        ; AND THE IEEE BIAS.
F2C2 4C                         JP $EXCPT         ; GO CHECK FOR OVER/UNDERFLOW.
                        ;
                        $NAGAS:
                        ; GET HERE MEANS 10'S EXPONENT IS NEGATIVE, SO GOT TO MULTIPLY
                        ; IT BY -3.
F2C3 E7                         SHL A             ; MULTIPLY BY 2.
F2C4 96CEF8                     ADD A, X          ; ADD TO GIVE MULTIPLY BY 3.
F2C7 01                         COMP A
F2C8 04                         INC A             ; MAKE IT NEGATIVE.
F2C9 96CCF8                     ADD A, B          ; GET IN THE BINARY EXPONENT.
F2CC B8007E                     ADD A, 07E        ; AND THE IEEE BIAS.
                        $EXCPT:
                        ; CHECK FOR OVERFLOW/UNDERFLOW.
F2CF ACC4CE                     LD X, SP          ; FIRST DO SOME JUGGLING
F2D2 02                         SET C             ; TO BE COMPATIBLE WITH EXCEPTION
F2D3 820ACEEB                   SUBC X, 0A        ; HANDLING IN OTHER ROUTINES.
F2D7 AFCE                       PUSH X
F2D9 BD7FFF                     IFGT A, 07FFF     ; IS BIASED EXPONENT NEGATIVE ?
F2DC B4FD3F                     JMPL UNDFL
F2DF 9C00                       IFEQ A, 0         ; IS IT 0 ?
F2E1 B4FD3A                     JMPL UNDFL        ; YES IT IS STILL UNDERFLOW.
F2E4 9DFE                       IFGT A, 0FE       ; IS IT GT THAN 254 ?
F2E6 B4FD46                     JMPL OVRFL
                        ; GET HERE MEANS VALID SP FLP NUMBER.
F2E9 3FCE                       POP X             ; X POINTS TO MANTISSA SIGN.
F2EB E7                         SHL A
F2EC E7                         SHL A
F2ED E7                         SHL A
F2EE E7                         SHL A
F2EF E7                         SHL A
F2FD E7                         SHL A
```

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:    22
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
ATOF
ATOF.MAC

F2F1 E7                      SHL A
F2F2 E7                      SHL A           ; MOVE EXPONENT TO HIGH BYTE.
F2F3 8FFA                    OR A, W(X)      ; GET THE MANTISSA SIGN IN.
F2F5 AB00                    ST A, TMP1      ; SAVE IT IN TMP1.
F2F7 3FCA                    POP K           ; FI-HI TO K.
F2F9 3FC8                    POP A           ; F1-LO TO A.
F2FB F1                      X A, W(X+)      ; SAVE F1-LO.
F2FC A8CA                    LD A, K
F2FE F1                      X A, W(X+)      ; SAVE F1-HI.
F2FF A800                    LD A, TMP1
F301 F3                      X A, W(X-)      ; SAVE F1-EXP.F1-SIGN, X POINTS TO F1-HI.
F302 3664                    JSRL SROUND     ; ROUND THE RESULT.
F304 F2                      LD A, W(X-)     ; X POINTS TO F1-HI.
F305 F2                      LD A, W(X-)     ; X POINTS TO F1-LO.
F306 AFCE                    PUSH X
F308 368C                    JSR FPAK        ; PACK IT INTO IEEE FORMAT.
F30A 3FC4                    POP SP
F30C 3FCC                    POP B
F30E 3FCE                    POP X
F310 3C                      RET
                     ;
                                     .END
```

```
                                    .FORM 'FTOA.MAC'
                                    .INCLD FTOA.MAC
                                    .TITLE FTOA
                                    .LOCAL
                    ;
                    ; THIS SUBROUTINE CONVERTS A SINGLE PRECISION, BINARY FLOATING
                    ; POINT NUMBER IN THE IEEE FORMAT TO A DECIMAL FLOATING POINT
                    ; STRING. THE DECIMAL FLOATING POINT STRING IS OBTAINED TO A
                    ; PRECISION OF 9 DECIMAL DIGITS.
                    ;
                    ; THE ALGORITHM USED IS BASED ON:
                    ; J.T. COONEN, 'AN IMPLEMENTATION GUIDE TO A PROPOSED STANDARD
                    ; FOR FLOATING POINT ARITHMETIC,' IEEE COMPUTER, JAN. 1980, PP 68-79.
                    ;
                    ; ON INPUT, THE BINARY SP FLP NUMBER IS IN REGS. K AND A.
                    ; B CONTAINS THE ADDRESS OF THE LOCATION WHERE THE DECIMAL FLOATING
                    ; POINT STRING IS TO START. NOTE THAT AT LEAST 17 BYTES ARE NEEDED
                    ; FOR THE STORAGE OF THE STRING. THE LAST BYTE IS ALWAYS NULL.
                    ;
                    ; ALL REGISTERS ARE PRESERVED BY THIS SUBROUTINE.
                    ;
                    FTOA:
F311 AFCE                   PUSH X          ; SAVE X ON THE STACK.
F313 AFCC                   PUSH B          ; SAVE B ON THE STACK.
                    ; CHECK AND SEE IF F1 IS A NAN.
F315 3605                   JSR FNACHK
F317 07                     IF C
F318 B401B4                 JMPL $NAN       ; YET IT IS, SO GET OUT.
                    ; CHECK AND SEE IF F1 IS ZERO.
F31B 36CA                   JSR FZCHK
F31D 07                     IF C
F31E B401C8                 JMPL $ZERO      ;YES IT IS, SO GET OUT.
                    ; GET HERE MEANS F1 IS A NON-ZERO, NON-NAN FLP NUMBER.
F321 ACC4CE                 LD X, SP
F324 8206C4F8               ADD SP, 06      ; ADJUST SP.
F328 36C7                   JSR FUNPAK      ; UNPACK THE NUMBER.
                                            ; X POINTS ONE WORD PAST F1-EXP.F1-SIGN
                                            ; ON RETURN.
                    ;
                    ; COMPUTE THE EXPONENT OF 10 FOR DECIMAL FLP NO.
                    ; THIS IS DONE AS FOLLOWS:
                    ;       SUPPOSE F1 = FM * (2^M)
                    ;       LET U = M*LOG(2)       NOTE: LOG IS TO BASE 10.
                    ;       THEN V = INT(U+1-9)
                    ;       IS USED AS THE 10'S EXPONENT.
                    ;                              NOTE: INT REFERS TO INTEGER PART.
                    ;
F32A D2                     LD A, M(X-)     ; X POINTS TO F1-EXP.
F32B D2                     LD A, M(X-)     ; LOAD F1-EXP. X POINTS TO F1-SIGN.
F32C B7000000               LD TMP1, 0      ; FIRST GUESS POSITIVE SIGN FOR EXP.
F330 B8FF82                 ADD A, 0FF82    ; REMOVE IEEE BIAS FROM F1-EXP.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    24
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

F333 AFC8                   PUSH A          ; SAVE IT ON THE STACK.
F335 AFCE                   PUSH X          ; SAVE F1-SIGN ADDRESS ALSO.
F337 07                     IF C            ; WAS THERE A CARRY ON THE LAST ADD ?
F338 46                     JP $MLOG2       ; YES, SO 2'S EXP IS POSITIVE.
F339 B700FF00               LD TMP1, OFF    ; 2'S EXPONENT IS NEGATIVE.
F33D 01                     COMP A
F33E 04                     INC A           ; MAKE IT POSITIVE.
                    $MLOG2:
                    ; MULTIPLY M BY LOG(2).
F33F BE4D10                 MULT A, 04D10   ; LOG(2) IS 0.0100110100010000 TO 16 BITS.
                                            ; X CONTAINS INTEGER PART, AND A FRACT. PART.
F342 AECE                   X A, X          ; SWAP THE TWO.
F344 960010                 IF TMP1.0       ; WAS THE 2'S EXPONENT NEGATIVE ?
F347 41                     JP $CSIGN       ; YES, SO MAKE U NEGATIVE.
F348 4B                     JP $REMV9       ; NO, SO GO DO V = U + 1 - 9.
                    $CSIGN:
F349 01                     COMP A          ; COMP INTEGER PART.
F34A AECE                   X A, X
F34C 01                     COMP A          ; FRACTION PART.
F34D B80001                 ADD A, 01
F350 AECE                   X A, X
F352 07                     IF C
F353 04                     INC A
                    $REMV9:
F354 04                     INC A           ; INCREASE FRACTION PART.
F355 B8FFF7                 ADD A, OFFF7    ; SUBTRACT 9.
F358 BD7FFF                 IFGT A, 07FFF   ; IS IT NEGATIVE ?
F35B 45                     JP $CHNGS       ; YES, SO CHANGE ITS SIGN.
F35C B7000000               LD TMP1, 0      ; REMEMBER POSITIVE SIGN.
F36D 4F                     JP $DIV10
                    $CHNGS:
F361 B700FF00               LD TMP1, OFF    ; REMEMBER NEGATIVE SIGN.
F365 01                     COMP A          ; MAKE V POSITIVE.
F366 AECE                   X A, X
F368 01                     COMP A
F369 B80001                 ADD A, 01
F36C AECE                   X A, X
F36E 07                     IF C
F36F 04                     INC A
                    $DIV10:
                    ;
                    ; V = INT (U+1-9) HAS BEEN COMPUTED AND IS IN A.
                    ; NOW COMPUTE W = F1/(10^V). W SHOULD BE AN INTEGER, AND IT IS
                    ; COMPUTED TO A 32 BIT PRECISION.
                    ; THIS COMPUTATION IS DONE AS FOLLOWS:
                    ;       IF V > 0, THEN F1/(10^V) = F1*(0.8^V)*(2^(-3V)).
                    ;       IF V < 0, THEN F1/(10^V) = F1*(0.625^U)*(2^(4V)).
                    ; SO FIRST MULTIPLY THE MANTISSA OF F1 V TIMES BY 0.8 (OR 0.625)
                    ; AND THEN ADJUST THE EXPONENT OF F1. NOTE THAT THE PARTIAL PRODUCTS
                    ; IN MULTIPLYING BY 0.8 (OR 0.625) ARE KEPT NORMALIZED. THIS IS
                    ; ESSENTIAL TO PRESERVE 32 BIT ACCURACY IN THE FINAL RESULT.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    25
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

                          ; SINCE THE MANTISSA OF F1 IS NORMALIZED, AND 0.8 (OR 0.625 IS ALSO
                          ; NORMALIZED, EACH PRODUCT NEEDS AT MOST 1 LEFT SHIFT FOR
                          ; RENORMALIZATION. THE SHIFTS ACCUMULATED DURING RENORMALIZATION ARE
                          ; TRACKED AND ACCOUNTED FOR IN THE CALCULATION.
F370 3FCE                           POP X             ; X NOW POINTS TO F1-SIGN.
F372 AFC8                           PUSH A            ; SAVE U ON THE STACK.
F374 ACC8CC                         LD B, A           ; MOVE V TO B ALSO.
F377 F2                             LD A, W(X-)       ; X POINTS TO F1-HI.
F378 F2                             LD A, W(X-)       ; LOAD F1-HI. X POINTS TO F1-LO.
F379 ACC8CA                         LD K, A
F37C F4                             LD A, W(X)        ; LOAD F1-LO.
F370 960010                         IF TMP1.0         ; IS V NEGATIVE?
F380 57                             JP $MUL10         ; YES, SO MULTIPLY V TIMES BY .625.
                                                      ; GET HERE MEANS MULTIPLY V TIMES BY .8.
F381 A4F390CEAB                     LD X, W($DTLO)
F386 AFCE                           PUSH X            ; LO WORD OF 0.8 TO STACK.
F388 A4F392CEAB                     LD X, W($DTHI)
F38D AFCE                           PUSH X            ; HI WORD OF 0.8 TO STACK.
F38F 56                             JP $JAMIT         ; GO DO MULTIPLICATION.
                          ;
                                    .EVEN             ; FORCE EVEN ADDRESS.
F390 CDCC                 $DTLO:    .WORD 0CCCD
F392 CCCC                 $DTHI:    .WORD 0CCCC
F394 0000                 $MILO:    .WORD 0
F396 00A0                 $MTHI:    .WORD 0A000
                          ;
                          $MUL10:
F398 A4F394CEAB                     LD X, W($MTLO)
F39D AFCE                           PUSH X            ; LO WORD OF 0.625 TO STACK.
F39F A4F396CEAB                     LD X, W($MTHI)
F3A4 AFCE                           PUSH X            ; HI WORD OF 0.625 TO STACK.
                          ;
                          $JAMIT:
F3A6 9300                           LD X, 0           ; INIT X TO TRACK ANY POWERS OF
                                                      ; 2 GENERATED DURING NORMALIZATION
                                                      ; OF PARTIAL PRODUCTS.
F3A8 8200CCFC                       IFEQ B, 0         ; IS B ALREADY 0 ?
F3AC 57                             JP $JAMON         ; YES, SO SKIP MULTIPLY LOOP.
                          $JAMLP:
F3AD 36E6                           JSR BFMUL         ; MULTIPLY.
F3AF AECA                           X A, K            ; SWAP HI AND LO WORDS OF PART. PROD.
F3B1 E7                             SHL A
F3B2 07                             IF C              ; IS THERE A CARRY ?
F3B3 4A                             JP $ISNED         ; YES, SO SKIP OVER RENORMALIZATION.
                                                      ; GET HERE MEANS NEED TO RENORM.
F3B4 A9CE                           INC X             ; UPDATE RENORM COUNT.
F3B6 AECA                           X A, K            ; SWAP HI AND LO PART. PROD.
F3B8 E7                             SHL A
F3B9 07                             IF C
F3BA 96CA08                         SET K.0           ; SET BIT SHIFTED OUT FROM LO WORD.
F3BD 43                             JP $OVR1
```

```
                     $ISNED:
F3BE D7                      RRC A          ; PUT BACK SHIFTED BIT.
F3BF AECA                    X A, K
                     $OVR1:
F3C1 AACC                    DECSZ B        ; IS B 0 YET ?
F3C3 76                      JP $JAMLP      ; NO, SO DO IT AGAIN.
                     $JAMON:
                     ; GET HERE MEANS MULTIPLICATION HAS BEEN DONE, SO TAKE CARE
                     ; OF EXPONENT.
F3C4 3FCC                    POP B
F3C6 3FCC                    POP B          ; GET RID OF 0.8 (OR 0.625) FROM STACK.
F3C8 AFC8                    PUSH A         ; SAVE LO WORD OF PROD.
F3CA AFCA                    PUSH K         ; SAVE HI WORD OF PRODUCT.
F3CC A6FFF8C4A8              LD A, W(SP-08) ; GET F1'S BINARY EXPONENT.
F3D1 02                      SET C
F3D2 96CEEB                  SUBC A, X      ; SUBTRACT FROM IT ANYTHING COLLECTED
                                            ; DURING RENORM.
F3D5 ACC8CE                  LD X, A        ; AND SAVE IT IN X.
F3D8 A6FFFAC4A8              LD A, W(SP-06) ; GET V FROM THE STACK.
F3DD 960010                  IF TMP1.0      ; IS V NEGATIVE ?
F3E0 49                      JP $ML4        ; YES, SO MULTIPLY V BY 4.
                                            ; GET HERE MEANS MULTIPLY V BY -3.
F3E1 E7                      SHL A          ; NOW A CONTAINS 2*V.
F3E2 A6FFFAC4F8              ADD A, W(SP-06) ; NOW A CONTAINS 3*V.
F3E7 01                      COMP A
F3E8 04                      INC A          ; NOW A CONTAINS -3*V.
F3E9 42                      JP $ADEM       ; GO FIGURE FINAL EXPONENT.
                     $ML4:
F3EA E7                      SHL A
F3EB E7                      SHL A          ; NOW A CONTAINS 4*V.
                     $ADEM:                 ;
F3EC 96CEF8                  ADD A, X       ; A SHOULD NOW BE A POSITIVE INTEGER
                                            ; IN THE RANGE 0 TO 32.
                     ; NOW CHECK AND SEE IF A HAS ENOUGH PRECISION.
F3EF 9020                    IFGT A, 020    ; NEED MORE THAN 32 BITS ?
F3F1 5A                      JP $INCRV      ; YES, SO GO INCREASE V.
F3F2 9D1B                    IFGT A, 01B    ; NEED AT LEAST 28 BITS ?
F3F4 9435                    JMP $GOON      ; YES, SO ALL IS OK. GO ON.
                                            ; GET HERE MEANS NEED MORE
                                            ; PRECISION, SO DECREASE V.
F3F6 3FC8                    POP A          ; GET HI-PROD OFF STACK.
F3F8 3FC8                    POP A          ; GET LO WORD OFF STACK.
F3FA 3FC8                    POP A          ; GET MAGN. OF V.
F3FC 96D010                  IF TMP1.0      ; IS V NEG. ?
F3FF 56                      JP $VUP        ; YES, SO GO INCR. MAGN. OF V.
                                            ; GET HERE MEANS V IS POSITIVE,
                                            ; AND NEED TO DECREMENT IT.
F400 B8FFFF                  ADD A, 0FFFF   ; SUBTRACT 1 FROM A.
F403 07                      IF C           ; GOT A CARRY ?
F404 5A                      JP $GOBAK      ; THEN OK.
F405 01                      COMP A
```

```
F406 04                      INC A
F407 B700FF00                LD TMP1, OFF    ; U CHANGES SIGN.
F40B 53                      JP $GOBAK
                     $INCRV:
F40C 3FC8                    POP A           ; GET HI PROD. OFF STACK.
F40E 3FC8                    POP A           ; GET LO PROD. OFF STACK.
F410 3FC8                    POP A           ; GET MAGN. OF V.
F412 960010                  IF TMP1.0       ; IS V NEGATIVE ?
F415 42                      JP $VDOWN       ; YES.
                     $VUP:
F416 04                      INC A
F417 47                      JP $GOBAK
                     $VDOWN:
F418 AAC8                    DECSZ A
F41A 44                      JP $GOBAK
F41B B7000000                LD TMP1, 0      ; V CHANGES SIGN.
                     $GOBAK:
F41F ACC4CE                  LD X, SP
F422 02                      SET C
F423 8204CEEB                SUBC X, 04
F427 AFCE                    PUSH X
F429 95B9                    JMP $DIV10
                     $GOON:
F42B 01                      COMP A
F42C 04                      INC A           ; NEGATE A.
F42D B80020                  ADD A, 020      ; SUBTRACT IT FROM 32.
F430 ACC8CE                  LD X, A         ; AND MOVE IT TO X.
F433 3FCA                    POP X           ; GET HI WORD OF PRODUCT.
F435 3FC8                    POP A           ; GET LO WORD OF PROD.
F437 8200CEFC                IFEQ X, 0       ; IS X 0 ?
F43B 49                      JP $INDUN       ; YES, SO ALREADY A 32 BIT INTEGER.
                     $INTFY:
                     ; NOW ADJUST THE PRODUCT TO FORM A 32 BIT INTEGER.
F43C AECA                    X A, K          ; SWAP HI AND LO WORDS.
F43E C7                      SHR A
F43F AECA                    X A, K
F441 D7                      RRC A           ; SHIFT IT RIGHT ONCE.
F442 AACE                    DECSZ X         ; X 0 YET ?
F444 68                      JP $INTFY       ; NO SO GO DO SOME MORE.
                     $INDUN:
                     ; GET HERE MEANS K.A CONTAIN THE 32 BIT INTEGER THAT IS THE
                     ; MANTISSA OF THE DECIMAL FLP NUMBER.
F445 AFC8                    PUSH A          ; SAVE LO-INT.
F447 AFCA                    PUSH K          ; SAVE HI INT.
F449 A6FFF0C4A8              LD A, W(SP-010) ; GET STARTING ADDRESS OF DECIMAL STRING.
F44E B80010                  ADD A, 010      ; ADD 16 TO IT.
F451 ACC8CC                  LD B, A         ; AND MOVE IT B.
F454 00                      CLR A
F455 C3                      XS A, M(B-)     ; OUTPUT TERMINATING NULL BYTE.
F456 40                      NOP
F457 A6FFFAC4A8              LD A, W(SP-06)  ; GET V.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    28
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

F45C 9F0A                    DIV A, 0A       ; DIVIDE IT BY 10. QUOT. IN A,
                                             ; REM. IN X.
F45E AECE                    X A, X          ; REM TO A.
F460 B80030                  ADD A, 030      ; MAKE IT INTO ASCII BYTE.
F463 C3                      XS A, M(B-)     ; OUTPUT IT.
F464 40                      NOP
F465 A8CE                    LD A, X
F467 B80030                  ADD A, 030
F46A C3                      XS A, M(B-)
F46B 40                      NOP             ; FINISHED OUTPUTING EXPONENT.
F46C 902B                    LD A, 028       ; SAY EXP SIGN IS '+'.
F46E 960010                  IF TMP1.0
F471 902D                    LD A, 02D       ; NOPE, IT IS '-'.
F473 C3                      XS A, M(B-)     ; OUTPUT IT.
F474 40                      NOP
F475 9045                    LD A, 045
F477 C3                      XS A, M(B-)     ; OUTPUT 'E'.
F478 40                      NOP
F479 902E                    LD A, 02E
F47B C3                      XS A, M(B-)     ; OUTPUT '.'.
F47C 40                      NOP
                     ; NOW NEED TO OUTPUT 10 DECIMAL DIGITS.
F47D B7000A00                LD TMP1, 0A     ; LOAD 10 INTO TMP1 AS LOOP COUNTER.
                     $DOLUP:
F481 3FC8                    POP A           ; A CONTAINS HI INT.
F483 9F0A                    DIV A, 0A       ; DIVIDE IT BY 10. QUOT. IN A,
                                             ; REM. IN X.
F485 ACC8CA                  LD K, A
F488 3FC8                    POP A           ; A CONTAINS LO INT.
F48A AFCC                    PUSH B          ; SAVE DEC. STR. ADDR.
F48C ACCACC                  LD B, K         ; B CONTAINS HI-QUOT.
F48F 82                      .BYTE 082,0A,0C8,0EF
F490 0A
F491 C8
F492 EF
                     ; THE ABOVE 4 BYTES REPRESENT THE INSTRUCTION DIVD A, 0A.
                     ; BECAUSE THE ASSEMBLER DOES NOT KNOW ABOUT IT YET, WE HAVE TO
                     ; KLUDGE IT THIS WAY.
                     ; AFTER THE DIVD, A CONTAINS THE LO-QUOT. AND X THE REM.
F493 ACCCCA                  LD K, B         ; MOVE HI-QUOT TO K.
F496 3FCC                    POP B           ; B CONTAINS DEC. STR. ADDR.
F498 AFC8                    PUSH A          ; SAVE LO INT.
F49A AFCA                    PUSH K          ; SAVE HI INT.
F49C A8CE                    LD A, X         ; MOVE REM TO A.
F49E B80030                  ADD A, 030      ; ASCII-FY IT.
F4A1 C3                      XS A, M(B-)     ; AND OUTPUT IT.
F4A2 40                      NOP
F4A3 AA00                    DECSZ TMP1      ; IS TMP1 0 YET ?
F4A5 9524                    JMP $DOLUP      ; NO, GO GET SOME MORE.
                     ; GET HERE MEANS DONE WITH OUTPUTING MANTISSA.
F4A7 3FC8                    POP A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    29
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FTOA.MAC

F4A9 3FC8                    POP A
F4AB 3FC8                    POP A
F4AD 3FC8                    POP A           ; GET SOME GARBAGE OFF THE STACK.
F4AF 3FCA                    POP K           ; GET F1-EXP.F1-SIGN TO K.
F4B1 9020                    LD A, 02D       ; LOAD SP INTO A.
F4B3 96CA10                  IF K.0
F4B6 902D                    LD A, 02D       ; IF MAINT. IS NEG. LOAD '-'.
F4B8 C6                      ST A, M(B)      ; OUTPUT SIGN.
F4B9 AFCA                    PUSH K          ; F1-EXP.F1-SIGN BACK ON STACK.
F4BB ACC4CE                  LD X, SP
F4BE 02                      SET C
F4BF 8206CEEB                SUBC X, 06      ; X POINTS TO F1-LO.
F4C3 AFCE                    PUSH X
F4C5 B5FBB4     +            JSR FPAK        ; PACK IT, SO RESTORING K AND A.
F4C8 3FC4                    POP SP
F4CA 3FCC                    POP B           ; RESTORE B.
F4CC 3FCE                    POP X           ; RESTORE X.
F4CE 3C                      RET
                     ;
                     $NAN:
                     ; GET HERE MEANS F1 IS A NAN.
F4CF AFC8                    PUSH A
F4D1 ACCCCE                  LD X, B
F4D4 8210CEF8                ADD X, 010
F4D8 00                      CLR A
F4D9 03                      X A, M(X-)
F4DA 9210                    LD B, 010
                     $NANLP:
F4DC 90FF                    LD A, 0FF
F4DE D3                      X A, M(X-)
F4DF AACC                    DECSZ B
F4E1 65                      JP $NANLP
F4E2 3FC8                    POP A
F4E4 3FCC                    POP B
F4E6 3FCE                    POP X
F4E8 3C                      RET
                     ;
                     $ZERO:
                     ; GET HERE MEANS F1 IS ZERO.
F4E9 AFC8                    PUSH A
F4EB ACCCCE                  LD X, B         ; X CONTAINS DECIMAL STRING ADDR.
F4EE 8210CEF8                ADD X, 010
F4F2 00                      CLR A
F4F3 D3                      X A, M(X-)      ; OUTPUT TERMINATING NULL BYTE.
F4F4 9030                    LD A, 030       ; LOAD 0 INTO A.
F4F6 D3                      X A, M(X-)
F4F7 9030                    LD A, 030
F4F9 D3                      X A, M(X-)      ; OUTPUT 00 FOR EXPONENT.
F4FA 902B                    LD A, 02B       ; LOAD '+' SIGN.
F4FC D3                      X A, M(X-)
F4FD 9045                    LD A, 045       ; LOAD 'E'
```

```
F4FF D3                          X A, M(X-)
F500 902E                        LD A, 02E        ; LOAD '.'.
F502 D3                          X A, M(X-)
F503 920A                        LD B, 0A
                         $ZERLP:
F505 9030                        LD A, 030
F507 D3                          X A, M(X-)
F508 AACC                        DECSZ B
F50A 65                          JP $ZERLP
F508 9020                        LD A, 020        ; LOAD SP.
F50D D5                          X A, M(X)
F50E 3FC8                        POP A
F510 3FCC                        POP B
F512 3FCE                        POP X
F514 3C                          RET
                         ;
                                 .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    31
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FTOA
FADD.MAC

                                    .FORM, 'FADD.MAX'
                                    .INCLD FADD.MAC
                                    .TITLE FADD
                                    .LOCAL
                    ;
                    ; SUBROUTINE TO ADD/SUBTRACT TWO SP FLOATING POINT NUMBERS.
                    ;          C = F1 + F2 OR C = F1 - F2
                    ;
                    ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                    ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                    ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                    ;
                    ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                    ; STACK POINTER ON ENTRY, THEN
                    ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                    ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                    ;
                    ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                    ;
                    FSUB:
F515 AB00                   ST A, TMP1
F517 A6FFFAC4A8             LD A, W(SP-06)  ; LOAD F2-R0.
F51C AFC8                   PUSH A          ; AND SAVE ON STACK.
F51E A6FFFAC4A8             LD A, W(SP-06)  ; LOAD F2-R1.
F523 BB8000                 XOR A, 08000    ; CHANGE THE SIGN.
F526 AFC8                   PUSH A          ; AND SAVE ON THE STACK.
F528 A800                   LD A, TMP1      ; RESTORE A.
F52A 3009                   JSR FADD        ; CALL THE ADD ROUTINE.
F52C AB00                   ST A, TMP1      ; SAVE A.
F52E 3FC8                   POP A           ; GET RID OF JUNK
F530 3FC8                   POP A           ; FROM THE STACK.
F532 A800                   LD A, TMP1      ; RESTORE A.
F534 3C                     RET
                    ;
                    FADD:
                    ; SAVE ADDRESS OF F2-R0 IN TMP1.
F535 AFCE                   PUSH X          ; SAVE X ON ENTRY.
F537 AFCC                   PUSH B          ; AND B ON ENTRY.
F539 ACC4CE                 LD X, SP
F53C 86FFF6CEF8             ADD X, 0FFF6    ; SUBTRACT 10.
F541 ACCE00                 LD TMP1, X      ; AND SAVE IN TMP1.
                    ; CHECK AND SEE IF F1 IS A NAN.
F544 B5FAF9      +          JSR FNACHK
F547 07                     IF C
F548 B4FAC4                 JMPL FNAN       ; F1 IS A NAN.
                    ; CHECK AND SEE IF F2 IS A NAN.
F54B ACCACC                 LD B, K
F54E ACC8CE                 LD X, A
F551 A20200A8               LD A, W(TMP1+2)
F555 ACC8CA                 LD K, A
F558 AECE                   X A, X
```

5

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:    32
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

F55A B5FAE3      +          JSR FNACHK
F55D 07                     IF C
F55E B4FAAE                 JMPL FNAN       ; F2 IS NAN.
                    ; CHECK AND SEE IF F2 IS ZERO.
F561 B5FAED      +          JSR FZCHK
F564 06                     IFN C
F565 48                     JP $F1CHK       ; F2 IS NOT ZERO. CHECK F1.
F566 ACCCCA                 LD K, B         ; F2 IS ZERO, SO ANSWER IS F1.
F569 3FCC                   POP B
F56B 3FCE                   POP X
F56D 3C                     RET
                    ; CHECK AND SEE IF F1 IS ZERO.
                    $F1CHK:
F56E ACCCCA                 LD K, B         ; RESTORE F1-R1 FROM B.
F571 B5FADD      +          JSR FZCHK
F574 06                     IFN C
F575 4F                     JP $NTZERO      ; JUMP SINCE F1 IS ALSO NOT ZERO.
F567 A20200AB               LD A, W(TMP1+2) ; GET HERE MEANS F1 IS ZERO,
F57A ACC8CA                 LD K, A         ; SO ANSWER IS F2.
F57D AD00A8                 LD A, W(TMP1)
F580 3FCC                   POP B
F582 3FCE                   POP X
F584 3C                     RET
                    ; GET HERE MEANS NORMAL ADDITION.
                    ; UNPACK F1 AND F2.
                    $NTZERO:
F585 ACC4CE                 LD X, SP        ; X POINTS TO F1-LO.
F588 82100C4F8              ADD SP, 010     ; MOVE SP PAST LOCAL STORAGE.
F58C AFCE                   PUSH X          ; SAVE SP ON STACK FOR QUICK RETURN.
F58E B5FAD0      +          JSR FUNPAK      ; UNPACK F1.
F591 AC00CC                 LD B, TMP1      ; B NOW POINTS TO F2-RO.
F594 ACCE00                 LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
F597 E0                     LDS A, W(B+)    ; LOAD F2-RO INTO A.
F598 40                     NOP
F599 AECA                   X A, K
F59B E4                     LD A, W(B)
F59C AECA                   X A, K          ; LOAD F2-R1 INTO K.
F59E B5FAC0      +          JSR FUNPAK      ; UNPACK F2.
                    ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
F5A1 F2                     LD A, W(X-)
F5A2 AC00CC                 LD B, TMP1
F5A5 E2                     LDS A, W(B-)
F5A6 40                     NOP
                    ; COMPARE F1-EXP AND F2-EXP.
F5A7 F2                     LD A, W(X-)     ; LOAD F2-EXP.F2-SIGN INTO A.
F5A8 B9FF00                 AND A, 0FF00    ; MASK OUT SIGN.
F5AB A6FFFCC4AB             ST A, W(SP-4)   ; SAVE IN C-SIGN.C-EXP.
F5B0 E2                     LDS A, W(B-)
F5B1 40                     NOP             ; LOAD F1-EXP.F1-SIGN INTO A.
F5B2 B9FF00                 AND A, 0FF00    ; CHANGE TO F1-EXP.00000000.
F5B5 02                     SET C
```

```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:    33
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

F5B6 A6FFFCC4EB                 SUBC A, W(SP-4) ; SUBTRACT F2-EXP.00000000.
F5BB 06                         IFN C
F5BC 942D                       JMP $F2GTR      ; F2-EXP IS BIGGER THAN F1-EXP.
                        ; GET HERE MEANS F1-EXP IS BIGGER THAN F2-EXP.
F5BE 80C9CAAB                   LD K, H(A)      ; SAVE DIFF. IN K TO BE USED AS LOOP COUNTER.
F5C2 A6FFFCC4FB                 ADD A, W(SP-4)
F5C7 A6FFFCC4AB                 ST A, W(SP-4)   ; RESTORE F1-EXP AND STORE IN C-SIGN.
F5CC 8217CAFD                   IFGT K, 017
F5D0 51                         JP $ZROF2       ; K GT 23-DEC MEANS F2 GETS ZEROED IN SHIFTS.
                        ; LOOP TO SHIFT F2 INTO ALIGNMENT.
F5D1 8200CAFC                   IFEQ K, 0
F5D5 943B                       JMP $ADDMN      ; K = 0 MEANS DONE SHIFTING.
                        $LOOP2:
F5D7 F4                         LD A, W(X)
F5D8 C7                         SHR A
F5D9 F3                         X A, W(X-)
F5DA F4                         LD A, W(X)
F5DB D7                         RRC A
F5DC F1                         X A, W(X+)
F5DD AACA                       DECSZ K
F5DF 68                         JP $LOOP2
F5E0 9430                       JMP $ADDMN
                        $ZROF2:
                        ; SET F2 MANTISSA TO 0.
F5E2 F0                         LD A, W(X+)     ; X POINTS TO F2-EXP.F2-SIGN.
F5E3 00                         CLR A
F5E4 F3                         X A, W(X-)      ; AND STORE IT BACK.
F5E5 00                         CLR A
F5E6 F3                         X A, W(X-)
F5E7 00                         CLR A
F5E8 F1                         X A, W(X+)
F5E9 9427                       JMP $ADDMN
                        ; F2 EXPONENT IS GREATER THAN F1 EXPONENT.
                        $F26TR:
F5EB 01                         COMP A
F5EC 04                         INC A           ; CHANGE DIFF IN EXP TO POSITIVE.
F5ED 80C9CAAB                   LD K, H(A)      ; LOAD K WITH LOOP COUNTER.
F5F1 8217CAFD                   IFGT K, 017
F5F5 51                         JP $ZROF1       ; F1 MANT. REDUCED TO 0 IN SHIFTS.
                        ; LOOP TO SHIFT F1 MANT INTO ALIGNMENT.
F5F6 8200CAFC                   IFEQ K, 0
F5FA 57                         JP $ADDMN       ; K=0 MEANS DONE SHIFTING.
                        $LOOP1:
F5FB E4                         LD A, W(B)
F5FC C7                         SHR A
F5FD E3                         XS A, W(B-)
F5FE 40                         NOP
F5FF E4                         LD A, W(B)
F600 D7                         RRC A
F601 E1                         XS A, W(B+)
F602 40                         NOP
```

```
F603 AACA                     DECSZ K         ;
F605 6A                       JP $LOOP1
F606 4B                       JP $ADDMN
                      $ZROF1:                 ; SET F1 MANT TO 0.
F607 E0                       LOS A, W(B+)    ; B POINTS TO F1-EXP.F1-SIGN.
F608 40                       NOP
F609 00                       CLR A
F60A E3                       XS A, W(B-)     ; STORE IT BACK.
F60B 40                       NOP
F60C 00                       CLR A
F60D E3                       XS A, W(B-)
F60E 40                       NOP
F60F 00                       CLR A
F610 E1                       XS A, W(B+)
F611 40                       NOP
                      ; DETERMINE IF MANTISSAS ARE TO BE ADDED OR SUBTRACTED.
                      $ADDMN:                 ; B POINTS TO F1-HI, X TO F2-HI.
F612 E0                       LDS A, W(B+)
F613 40                       NOP
F614 F0                       LD A, W(X+)
F615 D4                       LD A, M(X)      ; LOAD F2-SIGN.
F616 D8                       XOR A, M(B)     ; XOR WITH F1-SIGN.
F617 9C00                     IFEQ A, 0
F619 9451                     JMP $TRADD      ; SAME SIGN SO GO TO ADD MANTISSA.
                      ; GET HERE MEANS TRUE SUBTRACT OF MANTISSA.
F61B F2                       LD A, W(X-)
F61C F2                       LD A, W(X-)     ; X POINTS TO F2-LO.
F61D E2                       LDS A, W(B-)
F61E 40                       NOP
F61F E2                       LDS A, W(B-)
F620 40                       NOP             ; B NOW POINTS TO F1-LO.
F621 E0                       LDS A, W(B+)
F622 40                       NOP             ; A NOW CONTAINS F1-LO.
F623 02                       SET C
F624 8FEB                     SUBC A, W(X)    ; SUBTRACT F2-LO.
F626 F1                       X A, W(X+)
F627 E0                       LDS A, W(B+)    ; A CONTAINS F1-HI.
F628 40                       NOP
F629 8FEB                     SUBC A, W(X)    ; SUBTRACT F2-HI.
F62B F1                       X A, W(X+)
F62C 07                       IF C
F62D 55                       JP $F1SIN       ; F1 GE F2, SO SIGN IS F1-SIGN.
                      ; GET HERE MEANS F1 LT F2, SO SIGN IS F2-SIGN.
F62E A6FFFCC4A8               LD A, W(SP-4)
F633 8FDA                     OR A, M(X)
F635 F3                       X A, W(X-)      ; C-EXP.C-SIGN HAS BEEN DETERMINED.
F636 F4                       LD A, W(X)
F637 D1                       COMP A
F638 F3                       X A, W(X-)
F639 F4                       LD A, W(X)
F63A 01                       COMP A
```

```
F63B B80001                  ADD A, 01
F63E F1                      X A, W(X+)
F63F 07                      IF C
F640 8FA9                    INC W(X)
F642 47                      JP $ANORM
                     ; GET HERE MEANS F1 GE F2.
                     $F1SIN:
F643 A6FFFCC4A8              LD A, W(SP-4)
F648 DA                      OR A, M(B)
F649 F3                      X A, W(X-)
                     ; NORMALIZE THE MANTISSA.
                     $ANORM:
F64A ACCECC                  LD B, X         ; B POINTS TO C-HI.
F64D E0                      LDS A, W(B+)
F64E 40                      NOP
F64F C0                      LDS A, M(B+)
F650 40                      NOP             ; B NOW POINTS TO C-EXP BYTE.
F651 9118                    LD K, 018       ; SET UP LOOP LIMIT OF 24-DEC IN K.
                     $NLOOP:
F653 F4                      LD A, W(X)
F654 E7                      SHL A
F655 07                      IF C            ; CARRY MEANS NORMALIZED.
F656 9448                    JMP $ROUND      ; SO JUMP TO ROUNDING CODE.
F658 F3                      X A, W(X-)
F659 F4                      LD A, W(X)
F65A E7                      SHL A
F65B F1                      X A, W(X+)
F65C 07                      IF C
F65D 8F08                    SET W(X).0
F65F ADCC8A                  DECSZ M(B)      ; ADJUST EXPONENT.
F662 43                      JP $OV1
F663 B4F9B8                  JMPL UNDFL      ; C-EXP ZERO MEANS UNDERFLOW.
                     $OV1:
F666 AACA                    DECSZ K         ; DECREMENT LOOP COUNTER.
F668 75                      JP $NLOOP       ; GO BACK TO LOOP.
F669 B4F9B2                  JMPL UNDFL      ; UNDERFLOW
                     ;GET HERE MEANS TRUE ADDITION OF MANTISSA.
                     $TRADD:
F66C E2                      LDS A, W(B-)
F66D 40                      NOP
F66E E2                      LDS A, W(B-)
F66F 40                      NOP             ; B NOW POINTS TO F1-HI.
F670 F2                      LD A, W(X-)
F671 F2                      LD A, W(X-)
F672 F4                      LD A, W(X)      ; LOAD F2-LO INTO A.
F673 F8                      ADD A, W(B)     ; ADD F1-LO.
F674 F1                      X A, W(X+)      ; STORE IN F2-LO.
F675 E0                      LDS A, W(B+)
F676 40                      NOP             ; B NOW POINTS TO F1-HI.
F677 F4                      LD A, W(X)      ; LOAD F2-HI INTO A.
F678 E8                      ADC A, W(B)     ; ADD F1-HI WITH CARRY FROM LO ADD.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    36
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FADD.MAC

F679 07                      IF C
F67A 4A                      JP $ADJEX        ; IF CARRY, NEED TO INCREASE EXP.
F67B F1                      X A, W(X+)       ; STORE RESULT IN F2-HI.
F67C A6FFFCC4A8              LD A, W(SP-4)    ; GET C-EXP.00000000.
F681 8FDA                    OR A, M(X)       ; INTRODUCE SIGN.
F683 F3                      K A, W(X-)       ; STORE IN F2-EXP.F2-SIGN.
F684 5B                      JP $ROUND
                     ; GET HERE MEANS NEED TO INCREASE EXP BY 1.
                     $ADJEX:
F685 D7                      RRC A
F686 F3                      X A, W(X-)
F687 F4                      LD A, W(X)
F688 D7                      RRC A
F689 F1                      X A, W(X+)
F68A F0                      LD A, W(X+)      ; X NOW POINTS TO F2-EXP.F2-SIGN.
F68B A6FFFCC4A8              LD A, W(SP-4)    ; GET C-EXP.00000000.
F690 B80100                  ADD A, 0100      ; INCREASE EXP BY 1.
F693 07                      IF C
F694 B4F998                  JMPL OVRFL
F697 BDFEFF                  IFGT A, 0FEFF    ; IS BIASED EXPONENT 255-DEC ?
F69A B4F992                  JMPL OVRFL
F69D 8FDA                    OR A, M(X)
F69F F3                      X A, W(X-)
                     ; NEED TO ROUND THE RESULT. X POINTS TO C-HI.
                     $ROUND:
F6A0 B5F9FB                  JSRL SROUND
                     ; FINAL CHECK OF EXPONENT.
F6A3 D0                      LD A, M(X+)      ; X NOW POINTS TO C-EXP.
F6A4 D2                      LD A, M(X-)
F6A5 9C00                    IFEQ A, 0
F6A7 B4F974                  JMPL UNDFL
F6AA 90FE                    IFGT A, 0FE
F6AC B4F980                  JMPL OVRFL
F6AF F2                      LD A, W(X-)
F6B0 F2                      LD A, W(X-)      ; X NOW POINTS TO C-LO.
F6B1 B5F9C8      +           JSR FPAK         ; PACK C.
F6B4 3FC4                    POP SP           ; SET UP SP FOR RETURN.
F6B6 3FCC                    POP B
F6B8 3FCE                    POP X
F6BA 3C                      RET
                     ;
                             .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    37
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FADD
FMULT.MAC

  35                                     .FORM 'FMULT.MAC'
  36                                     .INCLD FMULT.MAC
   1                                     .TITLE FMULT
   2                                     .LOCAL
   3                     ;
   4                     ; SUBROUTINE TO MULTIPLY TWO SP FLOATING POINT NUMBERS.
   5                     ;       C = F1*F2
   6                     ;
   7                     ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
   8                     ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
   9                     ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
  10                     ;
  11                     ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
  12                     ; STACK POINTER ON ENTRY, THEN
  13                     ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
  14                     ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
  15                     ;
  16                     ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
  17                     ; REGS. X AND B ARE PRESERVED.
  18                     ;
  19                     FMULT:
  20 F6BB AFCE                   PUSH X          ; SAVE X ON ENTRY.
  21 F6BD AFCC                   PUSH B          ; SAVE B ON ENTRY.
  22                     ; SAVE ADDRESS OF F2-R0 IN TMP1.
  23 F6BF ACC4CE                 LD X, SP
  24 F6C2 86FFF6CEF8             ADD X, 0FFF6    ; SUBTRACT 10.
  25 F6C7 ACCE00                 LD TMP1, X      ; SAVE IN TMP1.
  26                     ; CHECK AND SEE IF F1 IS A NAN.
  27 F6CA B5F973      +          JSR FNACHK
  28 F6CD 07                     IF C
  29 F6CE B4F93E                 JMPL FNAN       ; F1 IS A NAN.
  30                     ; CHECK AND SEE IF F2 IS A NAN.
  31 F6D1 ACCACC                 LD B, K
  32 F6D4 ACC8CE                 LD X, A
  33 F6D7 A20200A8               LD A, W(TMP1+2)
  34 F6DB ACC8CA                 LD K, A
  35 F6DE AECE                   X A, X
  36 F6E0 B5F95D      +          JSR FNACHK
  37 F6E3 07                     IF C
  38 F6E4 B4F928                 JMPL FNAN       ; F2 IS NAN.
  39                     ; CHECK AND SEE IF F2 IS ZERO.
  40 F6E7 B5F967      +          JSR FZCHK
  41 F6EA 07                     IF C
  42 F6EB 94DC                   JMP $CZERO      ; F2 IS ZERO.
  43                     ; CHECK AN SEE IF F1 IS ZERO.
  44 F6ED ACCCCA                 LD K, B         ; RESTORE F1-R1 FROM B.
  45 F6FD B5F95E      +          JSR FZCHK
  46 F6F3 07                     IF C
  47 F6F4 94D3                   JMP $CZERO      ; F1 IS ZERO.
  48                     ; GET HERE MEANS NORMAL MULTIPLICATION.
  49                     ; UNPACK F1 AND F2.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    38
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

  50 F6F6 ACC4CE                 LD X, SP        ; X POINTS TO F1-LO.
  51 F6F9 8210C4F8               ADD SP, 010     ; MOVE SP PAST LOCAL STORAGE.
  52 F6FD AFCE                   PUSH X          ; SAVE SP ON STACK FOR QUICK RETURN.
  53 F6FF B5F95F       +         JSR FUNPAK      ; UNPACK F1.
  54 F702 AC00CC                 LD B, TMP1      ; B NOW POINTS TO F2-R0.
  55 F705 ACCE00                 LD TMP1, X      ; TMP1 NOW POINTS TO F2-LO.
  56 F708 E0                     LOS A, W(B+)    ; LOAD F2-R0 INTO A.
  57 F709 40                     NOP
  58 F70A AECA                   X A, K
  59 F70C E4                     LD A, W(B)
  60 F70D AECA                   X A, K          ; LOAD F2-R1 INTO K.
  61 F70F B5F94F       +         JSR FUNPAK      ; UNPAK F2.
  62                     ; SET X TO POINT TO F2-SIGN AND B TO POINT TO F1-SIGN.
  63 F712 F2                     LD A, W(X-)
  64 F713 AC00CC                 LD B, TMP1
  65 F716 E2                     LDS A, W(B-)
  66 F717 40                     NOP
  67                     ; COMPUTE C-EXP AND C-SIGN AND STORE IN F2-EXP AND F2-SIGN.
  68 F718 F4                     LD A, W(X)      ; A IS (EEEEEEEE-F2).(SSSSSSSS-F2)
  69 F719 C7                     SHR A           ; SHR SINCE SUM OF EXPS CAN BE 9 BITS.
  70 F71A ACC8CA                 LD K, A         ; K IS (DEEEEEEEE-F2).(SSSSSSS-F2)
  71 F71D E4                     LD A, W(B)      ; A IS (EEEEEEEE-F1).(SSSSSSSS-F1)
  72 F71E B9FF00                 AND A, 0FF00    ; MASK OUT SIGN BITS.
  73 F721 C7                     SHR A           ; A IS (DEEEEEEEE-F1).(0000000)
  74 F722 96CAF8                 ADD A, K        ; A IS (EEEEEEEEE-C).(SSSSSSS-F2)
  75 F725 F6                     ST A, W(X)      ; STORE IN F2-SIGN.
  76 F726 E2                     LOS A, W(B-)    ; A IS (EEEEEEEE-F1).(SSSSSSSS-F1)
  77 F727 40                     NOP
  78 F728 99FF                   AND A, 0FF      ; MASK OUT EXP BITS.
  79 F72A C7                     SHR A           ; A IS (000000000SSSSSSS-F1)
  80 F72B 8FFB                   XOR A, W(X)     ; A IS (EEEEEEEEESSSSSSS-C)
  81 F72D B8C080                 ADD A, 0C080    ; REMOVE EXCESS BIAS OF 127-DEC FROM EXP.
  82 F730 07                     IF C
  83 F731 46                     JP $EXCH2       ; IF CARRY, THEN NO UNDERFLOW NOW
  84                     ; CHECK TO SEE IF EXP IS ZERO. IF NOT, UNDERFLOW FOR SURE.
  85 F732 E7                     SHL A
  86 F733 07                     IF C
  87 F734 B4F8E7                 JMPL UNDFL      ; UNDERFLOW, SO JUMP.
  88 F737 C7                     SHR A           ; RESTORE BIT SHIFTED OUT (0).
  89                     ; CHECK FOR EXPONENT OVERFLOW.
  90                     $EXCH2:
  91 F738 E7                     SHL A
  92 F739 07                     IF C            ; IF C IS 1,
  93 F73A B4F8F2                 JMPL OVRFL      ; THEN OVERFLOW FOR SURE.
  94 F73D 96C817                 IF A.7
  95 F740 96C808                 SET A.0         ; RESTORE LAST BIT OF SIGN.
  96 F743 F3                     X A, W(X-)      ; STORE C-EXP. C-SIGN IN F2-EXP.F2-SIGN.
  97                     ;
  98                     ; MULTIPLY THE MANTISSA.
  99                     ; FIRST COMPUTE F1-HI*F2-HI.
 100 F744 F2                     LD A, W(X-)
```

```
F745 ACCE00                  LD TMP1, X        ; TMP1 NOW POINTS TO F2-LO.
F748 FE                      MULT A, W(B)
F749 A6FFFAC4AB              ST A, W(SP-6)     ; STORE LOW WORD OF PRODUCT ON STACK.
F74E AECE                    X A, X
F750 A6FFFCC4AB              ST A, W(SP-4)     ; STORE HIGH WORD OF PRODUCT ON STACK.
                     ; NOW COMPUTE F1-HI*F2-LO.
F755 AC00CE                  LD X, TMP1
F758 F0                      LD A, W(X+)
F759 ACCE00                  LD TMP1, X        ; TMP1 NOW POINTS TO F2-HI.
F75C FE                      MULT A, W(B)
F75D AECE                    X A, X
F75F A6FFFAC4F8              ADD A, W(SP-6)    ; ADD LOW WORD OF LAST PROD. TO HIGH WORD.
F764 A6FFFAC4AB              ST A, W(SP-6)
F769 07                      IF C
F76A A6FFFCC4A9              INC W(SP-4)       ; IF CARRY, INCREASE HIGH WORD BY 1.
                     ; FINALLY COMPUTE F1-LO*F2-HI.
F76F E2                      LDS A, W(B-)      ; ADJUST B TO POINT TO F1-LO.
F770 40                      NOP
F771 AC00CE                  LD X, TMP1
F774 F4                      LD A, W(X)
F775 FE                      MULT A, W(B)
F776 AECE                    X A, X
F778 A6FFFAC4F8              ADD A, W(SP-6)    ; ADD LOW WORD ACCUMULATED SO FAR.
F77D A6FFFAC4AB              ST A, W(SP-6)
F782 A6FFFCC4AB              LD A, W(SP-4)     ; A CONTAINS HIGH WORD OF PRODUCT.
F787 07                      IF C              ; IF CARRY ON LAST LOW WORD ADD,
F788 04                      INC A             ; THEN INCREASE HIGH WORD.
                     ;
                     ; MANTISSA MULTIPLICATION DONE. NOW CHECK FOR NORMALIZATION.
F789 AC00CE                  LD X, TMP1
F78C BD7FFF                  IFGT A, 07FFF     ; IS MSB OF PRODUCT 1 ?
F78F 4D                      JP $EXINC         ; YES, INCREASE MANTISSA.
                                               ; NEED TO SHIFT MANTISSA LEFT BY 1 BIT.
F790 E7                      SHL A
F791 F3                      X A, W(X-)
F792 A6FFFAC4AB              LD A, W(SP-6)
F797 E7                      SHL A
F798 F1                      X A, W(X+)
F799 07                      IF C              ; DID SHIFT OF LOW WORD PUSH OUT A 1 ?
F79A 8F08                    SET W(X).0        ; YES SO SET LSB OF HIGH WORD.
F79C 51                      JP $ROUND         ; GO TO ROUNDING CODE.
                     $EXINC:
                     ; NEED TO INCREASE EXPONENT BY 1. REMEMBER X POINTS TO F2-HI.
F79D F3                      X A, W(X-)        ; A CONTAINS HIGH WORD, X POINTS TO F2-LO.
F79E A6FFFAC4A8              LD A, W(SP-6)     ; GET LOW WORD.
F7A3 F1                      X A, W(X+)        ; STORE LOW WORD.
F7A4 F0                      LD A, W(X+)
F7A5 F4                      LD A, W(X)        ; GET C-EXP.C-SIGN
F7A6 B80100                  ADD A, 0100       ; INCREASE C-EXP.
F7A9 07                      IF C
F7AA B4F882                  JMPL OVRFL        ; EXPONENT OVERFLOW.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    40
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FMULT
FMULT.MAC

F7AD F3                      X A, W(X-)      ; NO OVERFLOW, SO SAVE C-EXP.C-SIGN.
                     ; ROUNDING CODE.
                     $ROUND:
F7AE B5F8ED                  JSRL SROUND
                     ; FINAL CHECK OF EXPONENT.
F7B1 D0                      LD A, M(X+)     ; X NOW POINTS TO C-EXP.
F7B2 D2                      LD A, M(X-)
F7B3 9C00                    IFEQ A, 0
F7B5 B4F866                  JMPL UNDFL
F7B8 9DFE                    IFGT A, 0FE
F7BA B4F872                  JMPL OVRFL
F7BD F2                      LD A, W(X-)
F7BE F2                      LD A, W(X-)     ; X NOW POINTS TO C-LO.
F7BF B5F8BA      +           JSR FPAK        ; PACK C.
F7C2 3FC4                    POP SP          ; SET UP SP FOR RETURN.
F7C4 3FCC                    POP B
F7C6 3FCE                    POP X
F7C8 3C                      RET
                     ; EXCEPTION HANDLING.
                     ; C IS ZERO B'COS ONE OF F1 OR F2 IS ZERO.
                     $CZERO:
F7C9 00                      CLR A
F7CA ACC8CA                  LD K, A
F7CD 3FCC                    POP B
F7CF 3FCE                    POP X
F7D1 3C                      RET
                     ;
                             .END
```

```
                                    .FORM 'FDIV.MAC'
                                    .INCLD FDIV.MAC
                                    .TITLE FDIV
                                    .LOCAL
                    ;
                    ; SUBROUTINE TO DIVIDE TWO SP FLOATING POINT NUMBERS.
                    ;    C = F1/F2
                    ;
                    ; F1 IS STORED IN THE IEEE FORMAT IN REGS K AND A.
                    ; THE HIGH WORD OF F1 WILL BE REFERRED AS F1-R1 AND IS IN K.
                    ; THE LOW WORD OF F1 WILL BE REFERRED TO AS F1-R0 AND IS IN A.
                    ;
                    ; F2 IS STORED IN THE IEEE FORMAT ON THE STACK. IF SP IS THE
                    ; STACK POINTER ON ENTRY, THEN
                    ; THE HIGH WORD OF F2, REFERRED TO AS F2-R1 IS AT SP - 4 AND
                    ; THE LOW WORD OF F2, REFERRED TO AS F2-R0 IS AT SP - 6.
                    ;
                    ; C IS RETURNED IN THE IEEE FORMAT IN REGS K AND A.
                    ;
                    FDIV:
F7D2 AFCE                   PUSH X
F7D4 AFCC                   PUSH B
                    ; SAVE ADDRESS OF F2-R0 IN TMP1.
F7D6 ACC4CE                 LD X, SP
F7D9 86FFF6CEF8             ADD X, 0FFF6    ; SUBTRACT 10.
F7DE ACCE00                 LD TMP1, X      ; AND SAVE IN TMP1.
                    ; CHECK AND SEE IF F1 IS A NAN.
F7E1 85F85C       +         JSR FNACHK
F7E4 07                     IF C
F7E5 B4F827                 JMPL FNAN       ; F1 IS A NAN.
                    ; CHECK AND SEE IF F2 IS A NAN.
F7E8 ACCACC                 LD B, K
F7EB ACC8CE                 LD X, A
F7EE A20200A8               LD A, W(TMP1+2)
F7F2 ACC8CA                 LD K, A
F7F5 AECE                   X A. X
F7F7 B5F846       +         JSR FNACHK
F7FA 07                     IF C
F7FB B4F811                 JMPL FNAN       ; F2 IS NAN.
                    ; CHECK AND SEE IF F2 IS ZERO.
F7FE B5F850       +         JSR FZCHK
F801 07                     IF C
F802 B4F7FB                 JMPL DIVBY0     ; F2 IS ZERO.
                    ; CHECK AND SEE IF F1 IS ZERO.
F805 ACCCCA                 LD K, B         ; RESTORE F1-R1 FROM B.
F808 B5F846       +         JSR FZCHK
F80B 07                     IF C
F80C 94F1                   JMP $CZERO      ; F1 IS ZERO.
                    ; GET HERE MEANS NORMAL DIVISION.
                    ; UNPACK F1 AND F2.
F80E ACC4CE                 LD X, SP        ; X POINTS TO F1-LO.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    42
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FDIV.MAC

     F811 8210C4F8                ADD SP, 010      ; MOVE SP PAST LOCAL STORAGE.
     F815 AFCE                    PUSH X           ; SAVE SP ON STACK FOR QUICK RETURN.
     F817 B5F847      +           JSR FUNPAK       ; UNPACK F1.
     F81A AC00CC                  LD B, TMP1       ; B NOW POINTS TO F2-R0
     F81D ACCE00                  LD TMP1, X       ; TMP1 NOW POINTS TO F2-LO.
     F820 E0                      LDS A, W(B+)     ; LOAD F2-R0 INTO A.
     F821 40                      NOP
     F822 AECA                    X A, K
     F824 E4                      LD A, W(B)
     F825 AECA                    X A, K           ; LOAD F2-R1 INTO K.
     F827 B5F837      +           JSR FUNPAK       ; UNPAK F2.
                         ;
                         ; ENSURE THAT F1-HI IS LESS THAN F2-HI.
                         ;
     F82A F2                      LD A, W(X-)      ; X POINTS TO F2-EXP.F2-SIGN.
     F82B F2                      LD A, W(X-)      ; X POINTS TO F2-HI.
     F82C AC00CC                  LD B, TMP1       ; B POINTS TO F2-LO.
     F82F E2                      LDS A, W(B-)     ; B POINTS TO F1-EXP.F1-SIGN.
     F830 40                      NOP
     F831 E2                      LDS A, W(B-)     ; LOAD F1-EXP.F1-SIGN.
     F832 40                      NOP              ; B POINTS TO F1-HI.
     F833 ACC8CA                  LD K, A          ; SAVE F1-EXP.F1-SIGN IN K.
     F836 F4                      LD A, W(X)       ; LOAD F2-HI.
     F837 FD                      IFGT A, W(B)     ; IS F2-HI > FI-HI ?
     F838 51                      JP $FEXSN        ; YES, SO ALL IS WELL.
                                                   ; GET HERE MEANS NEED TO SHR F1,
                                                   ; AND INCREASE ITS EXPONENT.
     F839 E0                      LOS A, W(B+)     ; GET FI-HI.
     F83A 40                      NOP              ; B POINTS TO F1-EXP.F1-SIGN.
     F83B AECA                    X A, K           ; SWAP F1-EXP.F1-SIGN AND F1-HI.
     F83D B80100                  ADD A, 0100      ; INCREASE F1-EXP BY 1.
     F840 E3                      XS A, W(B-)      ; STORE BACK IN F1-EXP.F1-SIGN.
     F841 40                      NOP              ; B POINTS TO F1-HI.
     F842 E4                      LD A, W(B)       ; LOAD F1-HI.
     F843 C7                      SHR A
     F844 E3                      XS A, W(B-)      ; STORE BACK IN F1-HI.
     F845 40                      NOP              ; B POINTS TO F1-LO.
     F846 E4                      LD A, W(B)       ; LOAD F1-LO.
     F847 D7                      RRC A
     F848 E1                      XS A, W(B+)      ; PUT IT BACK IN F1-LO.
     F849 40                      NOP              ; B POINTS TO F1-HI.
                         ;
                         $FEXSN:
                         ; DETERMINE C-EXP AND C-SIGN.
     F84A F0                      LD A, W(X+)      ; X POINTS TO F2-EXP.F2-SIGN.
     F84B E0                      LDS A, W(B+)     ; B POINTS TO F1-EXP.F1-SIGN.
     F84C 40                      NOP
     F84D F4                      LD A, W(X)       ; LOAD F2-EXP.F2-SIGN.
     F84E B9FF00                  AND A, 0FF00     ; MASK OUT THE SIGN.
     F851 C7                      SHR A            ; ALLOW 9 BITS FOR EXP CALCULATIONS.
     F852 ACC8CA                  LD K, A          ; SAVE IT IN K.
```

```
F855 E4                      LD A, W(B)       ; LOAD F1-EXP.F1-SIGN.
F856 B9FF00                  AND A, 0FF00     ; MASK OUT SIGN.
F859 C7                      SHR A
F85A 02                      SET C
F85B 96CAEB                  SUBC A, K        ; SUBTRACT THE EXPONENTS.
                                              ; NOTE THAT NOW THE MS 9 BITS
                                              ; OF A CONTAIN A 2'S COMP. INTEGER.
F85E B7000000                LD TMP1, 0
F862 E7                      SHL A
F863 07                      IF C
F864 B700FF00                LD TMP1, 0FF
F868 D7                      RRC A            ; SAVE SIGN OF NUMBER IN TMP1.
F869 B83F00                  ADD A, 03F00     ; RESTORE IEEE BIAS.
F86C E7                      SHL A            ; MAKE EXPONENT 8 BITS.
F86D 06                      IFN C            ; NO CARRY ?
F86E 49                      JP $FSIGN        ; THEN ALL IS WELL.
F86F 960010                  IF TMP1.0        ; WAS EXP NEGATIVE BEFORE ?
F872 B4F7A9                  JMPL UNDFL       ; YES, SO UNDERFLOW.
F875 B4F7B7                  JMPL OVRFL       ; OTHERWISE OVERFLOW.
                      ;
                      $FSIGN:
                      ; C-EXP HAS BEEN COMPUTED. NOW FIND C-SIGN.
                      ; BUT FIRST TAKE CARE OF SPECIAL OVER/UNDERFLOW CASES.
F878 BCFF00                  IFEQ A, 0FF00
F87B B4F7B1                  JMPL OVRFL
F87E 9C00                    IFEQ A, 0
F880 B4F79B                  JMPL UNDFL
F883 AB00                    ST A, TMP1       ; SAVE C-EXP.00000000 IN TMP1.
F885 F4                      LD A, W(X)       ; LOAD F2-EXP.F2-SIGN.
F886 99FF                    AND A, 0FF       ; MASK OUT F2-EXP.
F888 FB                      XOR A, W(B)      ; A NOW HAS F1-EXP.C-SIGN.
F889 99FF                    AND A, 0FF       ; MASK OUT F1-EXP.
F88B 9600FA                  OR A, TMP1       ; BRING IN C-EXP.
F88E F3                      X A, W(X-)       ; STORE IN F2-EXP.F2-SIGN.
                                              ; X POINTS TO F2-HI.
F88F F2                      LD A, W(X-)      ; X POINTS TO F2-LO.
F890 E2                      LDS A, W(B-)     ; B POINTS TO F1-HI.
F891 40                      NOP
                      ;
                      ; NOW DO THE MANTISSA DIVISION.
                      ;
F892 F0                      LD A, W(X+)      ; LOAD F2-LO. X POINTS TO F2-HI.
F893 ACCE00                  LD TMP1, X       ; SAVE ADDRESS OF F2-HI IN TMP1.
F896 FE                      MULT A, W(B)     ; COMPUTE F2-LO*F1-HI.
                                              ; X CONTAINS MS WORD AND A IS LS WORD.
                      ;
F897 AECC                    X A, B           ; A POINTS TO F1-HI, B CONTAINS LS WORD.
F899 AE00                    X A, TMP1        ; A POINTS TO F2-HI, TMP1 POINTS TO F1-HI.
F89B AECC                    X A, B           ; A CONTAINS LS WORD, B POINTS TO F2-HI.
                      ;
F890 EF                      .BYTE 0EF        ; DIVD A, W(B) - KLUDGED !!
```

```
                      ;
F89E ACC8CA                  LD K, A          ; SAVE QUOTIENT IN K.
F8A1 A8CC                    LD A, B          ; A POINTS TO F2-HI.
F8A3 AE00                    X A, TMP1        ; A POINTS TO F1-HI, TMP1 POINTS TO F2-HI.
F8A5 AECC                    X A, B           ; B POINTS TO F1-HI.
F8A7 E2                      LDS A, W(B-)     ; B POINTS TO F1-LO.
F8A8 40                      NOP
F8A9 E0                      LOS A, W(B+)     ; LOAD F1-LO.
F8AA 40                      NOP              ; B POINTS TO F1-HI.
F8AB 02                      SET C
F8AC 96CAEB                  SUBC A, K        ; SUBTRACT QUOTIENT SAVED IN K.
F8AF ACC8CE                  LD X, A          ; AND SAVE IN X.
F8B2 E4                      LD A, W(B)       ; LOAD F1-HI.
F8B3 06                      IFN C            ; IF C WAS NOT SET IN THE LAST SUBTRACT,
F8B4 05                      DEC A            ; ADJUST THE BORROW.
F8B5 AECE                    X A, X
F8B7 AC00CC                  LD B, TMP1       ; B POINTS TO F2-HI.
                      ;
F8BA EF                      .BYTE 0EF        ; DIVD A, W(B) - KLUDGED AGAIN !
                                              ; QUOTIENT IN A, REM IN X.
                      ;
F8BB AB00                    ST A, TMP1       ; SAVE QUOTIENT IN TMP1.
F8BD 00                      CLR A            ; ZERO A.
                      ;
F8BE EF                      .BYTE 0EF        ; DIVD A, W(B) - KLUDGED YET AGAIN !
                      ;
F8BF AE00                    X A, TMP1        ; SWAP OLD AND NEW QUOTIENTS.
                      ;
                      ; CHECK FOR NORMALIZATION. CAN BE OFF BY AT MOST 1 BIT.
F8C1 E7                      SHL A
F8C2 07                      IF C
F8C3 56                      JP $NMED         ; IT IS NORMALIZED.
                                              ; GET HERE MEANS NEED TO SHIFT LEFT ONCE.
F8C4 AE00                    X A, TMP1        ; SWAP HI AND LO WORDS.
F8C6 E7                      SHL A
F8C7 AE00                    X A, TMP1        ; HI WORD IS IN A, LO WORD IN TMP1.
F8C9 07                      IF C             ; WAS 1 SHIFTED OUT OF LO WORD?
F8CA 96C808                  SET A.0          ; YES, THEN SET LSB OF HI WORD.
F8CD ABCA                    ST A, K          ; SAVE HI WORD IN K.
F8CF E1                      XS A, W(B+)
F8D0 40                      NOP              ; B POINTS TO F2-EXP.F2-SIGN.
F8D1 E4                      LD A, W(B)       ; LOAD F2-EXP.F2-SIGN.
F8D2 B8FF00                  ADD A, 0FF00     ; SUBTRACT 1 FROM EXPONENT.
F8D5 E3                      XS A, W(B-)      ; STORE BACK IN F2-EXP.F2-SIGN.
F8D6 40                      NOP              ; B POINTS TO F2-HI.
F8D7 A8CA                    LD A, K          ; HI WORD TO A.
F8D9 E7                      SHL A
                      $NMED:
F8DA D7                      RRC A            ; RESTORE BIT OUT.
F8DB E3                      XS A, W(B-)      ; SAVE HI-WORD IN F2-HI.
F8DC 40                      NOP              ; B POINTS TO F2-LO.
```

```
F8DD A800                    LD A, TMP1
F8DF E1                      XS A, W(B+)     ; SAVE C-LO.
F8E0 40                      NOP             ; B POINTS TO F2-HI.
F8E1 ACCCCE                  LD X, B         ; MOVE ADDRESS OF F2-HI TO X.
                     ;
                     ; ROUNDING CODE.
F8E4 B5F7B7                  JSRL SROUND
                     ; FINAL CHECK OF EXPONENT.
F8E7 D0                      LD A, M(X+)     ; X NOW POINTS TO C-EXP.
F8E8 D2                      LD A, M(X-)
F8E9 9C00                    IFEQ A, 0
F8EB B4F730                  JMPL UNDFL
F8EE 9DFE                    IFGT A, 0FE
F8F0 B4F73C                  JMPL OVRFL
F8F3 F2                      LD A, W(X-)
F8F4 F2                      LD A, W(X-)     ; X NOW POINTS TO C-LO.
F8F5 B5F784      +           JSR FPAK        ; PACK C.
F8F8 3FC4                    POP SP          ; SET UP SP FOR RETURN.
F8FA 3FCC                    POP B
F8FC 3FCE                    POP X
F8FE 3C                      RET
                     ; C IS ZERO B'COS F1 IS ZERO.
                     $CZERO:
F8FF 00                      CLR A
F900 ACC8CA                  LD K, A
F903 3FCC                    POP B
F905 3FCE                    POP X
F907 3C                      RET
                     ;
                             .END
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    46
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
FDIV
FSINX.MAC

                                    .FORM 'FSINX.MAC'
                                    .INCLD FSINX.MAC
                    ;
                                    .TITLE SINX
                                    .LOCAL
                    ; A VERY DIRTY APPROXIMATION TO SIN(X).
                    ; X SHOULD BE IN RADIANS.
                    ;
                    ; ON INPUT X SHOULD BE IN IEEE FLP FORMAT IN REGS. K AND A.
                    ; ON RETURN SIN(X) IS IN IEEE FLP FORMAT IN REGS. K AND A.
                    ;
                    SINX:
F908 AFCE                   PUSH X           ; SAVE X.
F90A AFC8                   PUSH A
F90C AFCA                   PUSH K           ; X TO THE STACK.
F90E 3653           -       JSRL FMULT       ; COMPUTE X^2.
F910 AFC8                   PUSH A
F912 AFCA                   PUSH K           ; X^2 TO THE STACK.
F914 B6F994A8               LD A, W($A5LO)
F918 A4F996CAAB             LD K, W($A5HI)   ; LOAD A5.
F91D 3662           -       JSRL FMULT       ; COMPUTE A5*X^2.
F91F AFC8                   PUSH A
F921 AFCA                   PUSH K
F923 B6F998A8               LD A, W($A4LO)
F927 A4F99ACAAB             LD K, W($A4HI)   ; LOAD A4.
F92C B5FBE6                 JSRL FSUB        ; COMPUTE A4-A5*X^2.
F92F 3FCE                   POP X
F931 3FCE                   POP X
F933 3678           -       JSRL FMULT       ; COMPUTE
                                             ; X^2(A4 - A5*X^2).
F935 AFC8                   PUSH A
F937 AFCA                   PUSH K
F939 B6F99CA8               LD A, W($A3LO)
F93D A4F99ECAAB             LD K, W($A3HI)   ; LOAD A3.
F942 B5FBD0                 JSRL FSUB        ; COMPUTE
                                             ; A3 - X^2(A4 - A5*X^2).
F945 3FCE                   POP X
F947 3FCE                   POP X
F949 368E           -       JSRL FMULT       ; COMPUTE
                                             ; X^2(A3 - X^2(A4 - A5*X^2)).
F94B AFC8                   PUSH A
F94D AFCA                   PUSH K
F94F B6F9A0A8               LD A, W($A2LO)
F953 A4F9A2CAAB             LD K, W($A2HI)   ; LOAD A2.
F958 B5FBBA                 JSRL FSUB        ; COMPUTE
                                             ; A2 - X^2(A3 - X^2(A4 - A5*X^2)).
F95B 3FCE                   POP X
F95D 3FCE                   POP X
F95F 36A4           -       JSRL FMULT       ; COMPUTE
                                             ; X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
F961 AFC8                   PUSH A
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    47
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
FSINX.MAC

  50 F963 AFCA                    PUSH K
  51 F965 B6F9A4AB                LD A, W($A1LO)
  52 F969 A4F9A6CAAB              LD K, W($A1HI)   ; LOAD A1.
  53 F96E B5FBA4                  JSRL FSUB        ; COMPUTE
  54                                               ; A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))).
  55 F971 3FCE                    POP X
  56 F973 3FCE                    POP X
  57 F975 36BA            -       JSRL FMULT       ; COMPUTE
  58                                               ; X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2)))).
  59 F977 AFC8                    PUSH A
  60 F979 AFCA                    PUSH K
  61 F97B B13F80                  LD K, 03F80
  62 F97E 00                      CLR A            ; LOAD 1.0 INTO K-A.
  63 F97F B5FB93                  JSRL FSUB        ; COMPUTE
  64                                               ; 1 - ALL THE JUNK ABOVE.
  65 F982 3FCE                    POP X
  66 F984 3FCE                    POP X
  67 F986 3FCE                    POP X
  68 F988 3FCE                    POP X            ; NOW X IS AT THE TOP OF STACK.
  69 F98A 36CF            -       JSRL FMULT       ; COMPUTE
  70                                       ; X(1 - X^2(A1 - X^2(A2 - X^2(A3 - X^2(A4 - A5*X^2))))).
  71 F98C 3FCE                    POP X
  72 F98E 3FCE                    POP X
  73 F990 3FCE                    POP X
  74 F992 3C                      RET
  75                      ;
  76 F993 40                              .EVEN
  77                      ;
  78 F994 2B32            $A5LO: .WORD 0322B
  79 F996 D732            $A5HI: .WORD 032D7
  80 F998 1DEF            $A4LO: .WORD 0EF1D
  81 F99A 3836            $A4HI: .WORD 03638
  82 F99C 010D            $A3LO: .WORD 00D01
  83 F99E 5039            $A3HI: .WORD 03950
  84 F9A0 8988            $A2LO: .WORD 08889
  85 F9A2 083C            $A2HI: .WORD 03C08
  86 F9A4 ADAA            $A1LO: .WORD 0AAAD
  87 F9A6 2A3E            $A1HI: .WORD 03E2A
  88                      ;
  89                      ; A DIRTY APPROXIMATION TO COS(X) USING SIN(X).
  90                      ;
  91                      COSX:
  92 F9A8 AFCE                    PUSH X
  93 F9AA ACC8CE                  LD X, A
  94 F9AD B6F9C8A8                LD A, W($PI2LO)
  95 F9B1 AFC8                    PUSH A
  96 F9B3 B6F9CAA8                LD A, W($PI2HI)
  97 F9B7 AFC8                    PUSH A
  98 F9B9 A8CE                    LD A, X
  99 F9BB B5FB77                  JSRL FADD        ; COMPUTE X + PI/2.
 100 F9BE 3FCE                    POP X
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    48
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
FSINX.MAC

F9C0 3FCE                      POP X
F9C2 34BA              -       JSRL SINX        ; COMPUTE SIN(X+PI/2).
F9C4 3FCE                      POP X
F9C6 3C                        RET
                       ;
F9C7 40                                .EVEN
F9C8 DB0F              $PI2LO: .WORD 00FDB
F9CA C93F              $PI2HI: .WORD 03FC9
                       ;
                       ; A DIRTY APPROXIMATION TO TAN(X) USING SINX AND COSX.
                       ;
                       TANX:
F9CC AFCE                      PUSH X
F9CE AFCC                      PUSH B
F9D0 AFC8                      PUSH A
F9D2 AFCA                      PUSH K
F9D4 342C                      JSR COSX         ; COMPUTE COS(X)
F9D6 ACC8CE                    LD X, A
F9D9 ACCACC                    LD B, K
F9DC 3FCA                      POP K
F9DE 3FC8                      POP A
F9E0 AFCE                      PUSH X
F9E2 AFCC                      PUSH B
F9E4 34DC                      JSR SINX         ; COMPUTE SIN(X).
F9E6 3614                      JSR FDIV         ; COMPUTE TAN(X) = SIN(X)/COS(X).
F9E8 3FCC                      POP B
F9EA 3FCC                      POP B
F9EC 3FCC                      POP B
F9EE 3FCE                      POP X
F9F0 3C                        RET
                       ;
                               .END
                       ;
                       ;
FFFE 00F0                              .END LISTER
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    49
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
SYMBOL TABLE

A      00C8 W       ATOF   F13A *     B      00CC W       BFMUL  F0C7
COSX   F9A8         DIVBYO F000       FADD   F535         FDIV   F7D2
FMULT  F6BB         FNACHK F040       FNAN   F00F         FPAK   F07C
FPERWD 0002 W       FPTRAP F09D       FSUB   F515         FTOA   F311 *
FUNPAK F061         FZCHK  F051       ISIOK  F105         K      00CA W
LISTER F000         MUL10  F118       OVRFL  F02F         PC     00C6 W
SINX   F908         SP     00C4 W     SROUND F09E         TANX   F9CC *
TMP1   0000 W       UNDFL  F01E       X      00CE W       $A10EX F1F4
$A1HI  F9A6         $A1LO  F9A4       $A2HI  F9A2         $A2LO  F9A0
$A3HI  F99E         $A3LO  F99C       $A4HI  F99A         $A4LO  F998
$A5HI  F996         $A5LO  F994       $ACCF  F1A9         $ACCM  F177
$ADDEX F1FF         $ADDMN F612       $ADEM  F3EC         $ADJEX F685
$ANORM F64A         $ANOTO F05C       $CHKOT F113         $CHNGS F361
$CSIGN F349         $CZERO F7C9       $CZERO F8FF         $DIV10 F272
$DIV10 F370         $DOLUP F481       $DTHI  F270         $DTHI  F392
$DTLO  F26E         $DTLO  F390       $ESAVE F20F         $ESIGN F1C7
$EXACC F1C9         $EXCH2 F738       $EXCHR F1BB         $EXCLP F1D4
$EXCOL F1BA         $EXCPT F2CF       $EXIN2 F0B7         $EXINC F79D
$EXIT  F0C6         $F1CHK F56E       $F1SIN F643         $F2GTR F5EB
$TEXSN F84A         $FRCOL F18B       $FSIGN F878         $GOBAK F41F
$GOON  F42B         $HIUP  F0AF       $INCOL F15C         $INCRV F40C
$INDUN F445         $INTFY F43C       $ISNAN F04C         $ISNED F298
$ISNED F3BE         $ISNXT F17D       $JAMDN F29E         $JAMDN F3C4
$JAMIT F280         $JAMIT F3A6       $JAMLP F287         $JAMLP F3AD
$LOOP1 F147         $LOOP1 F5FB       $LOOP2 F5D7         $ML4   F3EA
$MLOG2 F33F         $MSIGN F154       $MTHI  F26C         $MTHI  F396
$MTLO  F26A         $MTLO  F394       $MUL10 F398         $NAGAS F2C3
$NAN   F4CF         $NANLP F4DC       $NEG10 F20B         $NLOOP F653
$NMED  F8DA         $NORM1 F230       $NORM2 F235         $NRDUN F247
$NRLUP F237         $NTZER F585       $OV1   F666         $OVR1  F29B
$OVR1  F3C1         $PI2HI F9CA       $PI2LO F9C8         $REMV9 F354
$RNDUP F0A7         $ROUND F6A0       $ROUND F7AE         $TRADD F66C
$VDOWN F418         $VUP   F416       $ZERLP F505         $ZERO  F4E9
$ZROF1 F607         $ZROF2 F5E2
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:    50
HPC CROSS ASSEMBLER,REV:C,30 JUL 86
SINX
MACRO TABLE

   NO WARNING LINES

   NO ERROR LINES

  2547 ROM BYTES USED

SOURCE CHECKSUM = A31F
OBJECT CHECKSUM = 2AC3

INPUT  FILE C:LISTER.MAC
LISTING FILE C:LISTER.PRN
OBJECT  FILE C:LISTER.LM
```

A Radix 2 FFT Program for the HPC

National Semiconductor
Application Note 487
Ashok Krishnamurthy

INTRODUCTION

This report describes the implementation of a radix-2, Decimation-in-time FFT algorithm on the HPC. The program, as presently set up can do FFTs of length 2, 4, 8, 16, 32, 64, 128 and 256. The program can be easily modified to work with higher FFT lengths by increasing the Twiddle Factor table.

FFT FUNDAMENTALS

If x(n), n = 0, 1, . . . , N−1 are N samples of a time domain signal, its Discrete Fourier Transform (DFT) is defined as

$$X(k) = \sum_{n=0}^{n=N-1} x(n)\, W^{nk}, \quad k = 0, 1, \ldots, N-1$$

where $W = e^{-j2\pi/N}$

The straight evaluation of the above equation requires on the order of N^2 complex multiplies. The FFT is nothing but a fast algorithm to compute the DFT that uses only on the order of N log(N) complex multiplies. Many different FFT algorithms exist (please see references 1, 2 and 3). The algorithm implemented for the HPC is the most common type of FFT — a radix-2, Decimation-in-time algorithm. This class of algorithms requires that the number of input samples, N, be a power of 2. This is usually not a problem, since the input data can be zero padded to achieve this. The development of this algorithm is described in references 1 and 2; the discussion here is brief and based on reference 1.

Separating the DFT summation above into the even-numbered points and odd-numbered points of x(n), we can rewrite the above sum as:

$$X(k) = \sum_{n\ \text{even}} x(n)\, W^{nk} + \sum_{n\ \text{odd}} x(n)\, W^{nk}$$

Using n = 2r for n even and n = 2r + 1 for n odd, we can further rewrite the above as:

$$X(k) = \sum_{r=0}^{N/2-1} x(2r)\, W^{2rk} + W^k \sum_{r=0}^{N/2-1} x(2r+1)\, W^{2rk}$$

If G(k) is the N/2 point DFT of x(2r) and H(k) is the N/2 point DFT of x(2r+1), the above equation can be written as:

$$X(k) = G(k) + W^k H(k)$$

This equation shows that a N point DFT can be written as the sum of two N/2 point DFTs. The N/2 point DFTs can be computed as the sum two N/4 point DFTs and so on until we are left with two point DFTs. The two point DFTs can be trivially evaluated by direct computation.

Figure 1, taken from reference 1, shows the decomposition for the case N = 8. With reference to this figure, we can note the following points.

1. If N is the number of points in the original sequence, where $N = 2^L$, then there are L stages in the DFT decomposition.
2. The basic computation unit is the so-called Butterfly, shown in *Figure 2*. Each stage involves the computation of N/2 butterflies.
3. The results from the computation in one stage are fed to the next stage after multiplication by some power of W. These powers of W are the so-called Twiddle Factors. Note that each power of W is really a complex number that can be represented by its real and imaginary parts. The real part of W^k is $\cos(2\pi k/N)$ and the imaginary part is $-\sin(2\pi k/N)$.
4. The number of distinct Twiddle Factors used in the first stage is 1, in the second stage is 2 etc., until the Lth stage that involves $2^{L-1} = N/2$ twiddle factors. Each twiddle factor in the first stage is involved in N/2 Butterflies, in the second stage with N/4 butterflies etc., until in the Lth stage each twiddle factor is involved with $^N/(2^L) = 1$ butterfly.
5. The input data sequence needs to be suitably scrambled if the output sequence is to be in the proper order. This scrambling is easily accomplished by using the so-called Bit-Reverse counter as outlined in reference 2.
6. The outputs from each stage can be stored back again in the same storage area as the input sequence. This gives the algorithm the in-place property. Thus the final DFT results overwrite the initial data.

THE INVERSE FFT

If X(k) k = 0, 1, . . . , N−1 is the DFT of a sequence, then its inverse DFT, x(n), is defined as follows:

$$x(n) = \left(\frac{1}{N}\right) \sum_{k=0}^{k=N-1} X(k)\, W^{-nk} \quad n = 0, 1, .., N-1.$$

Thus the Inverse FFT is the same as the forward FFT except for the following: 1. Negative powers of W are used instead of positive powers; and 2. The final sequence is scaled by 1/N. The basic FFT program can therefore be used to compute the inverse FFT with these two changes. This is the approach used in the HPC implementation.

TWIDDLE FACTOR TABLE

The brief description of the FFT in the previous section shows that the algorithm needs to use the Twiddle Factors W^k in the computation. The twiddle factors can either be computed as required, they can be computed using a recursive relation, or they can be obtained by looking up in a table (Ref. 2). The approach used in the HPC implementation is to construct a table containing the needed twiddle factors. This table is stored in ROM and values needed are looked up from this table. The length of the table needed is determined by the maximum FFT length that you want to use. The HPC FFT implementation is presently limited to a maximum length of 256. This requires that the twiddle factors W^0, W^1, . . . W^{255} be available, where

$W = e^{-j2\pi/256}$. Since $e^{jx} = \cos(x) + j\sin(x)$, the values stored in this table are cos(0), sin(0), cos(2π/256), sin(2π/256) etc., up to cos(2π × 255/256), sin (2π × 255/256). The table used in the implementation is organized as follows:

.WORD $\cos(0) \times 2^{14}$
.WORD $\sin(0) \times 2^{14}$
.WORD $\cos(2\pi/256) \times 2^{14}$
.WORD $\sin(2\pi/256) \times 2^{14}$
.
.
.
.WORD $\cos(2\pi 255/256) \times 2^{14}$
.WORD $\sin(2\pi 255/256) \times 2^{14}$

This table is available in the file TWDTBL.MAC and occupies 1024 bytes of storage.

DATA STORAGE

The data to be transformed, x(0), . . . , x(N−1) are also regarded as complex numbers with a real and an imaginary part. Let xr(i) be the real part of x(i) and xi(i) the imaginary part of x(i). Then the data needs to be stored as follows:

```
.WORD xr(0)
.WORD xi(0)
.WORD xr(1)
.WORD xi(1)
.
.
.
.WORD xr(N−1)
.WORD xi(N−1)
```

The length of this storage area obviously depends on the number of data points to be transformed. Note that the FFT program itself does not use any base page user RAM. Also, only 8 words of stack are needed. Thus the base page user RAM can be used to store the data to be transformed. Since 192 bytes are available in this area, transforms of up to 32 point in length can be in the single chip mode with no external RAM.

USING THE FFT PROGRAM

The FFT program along with test data to test the program is provided in the files FFT.MAC, TSTDAT.MAC and TWDTBL.MAC. TSTDAT.MAC contains the test data, and the output from the FFT routines. TWDTBL.MAC contains the Twiddle Factors. The FFT computation involves the use of 4 different subroutines: FFT, IFFT, BRNCNTR and SMULT. FFT does the forward FFT calculation, IFFT the Inverse FFT calculation, BRNCNTR implements the bit reversed counter, and SMULT does signed multiplication.

Two global symbols need to be defined by the user to use the FFT routines. The first, called TWSTAD should be set to the address of the start of the twiddle factor table. The second, called DTSTAD, should be set to the address of the start of the data area to be transformed. For details on the organization of these storage areas, see the preceding sections.

The actual number of data points to be transformed needs to be passed to the FFT routines. This is done as follows.

Two symbols that refer to words of on-chip RAM have been defined. The first is NUMB = W(01C0) and the second is L1 = W(01C2). Before calling the FFT routine, the user should load NUMB with N, the number of data points to be transformed, and L1 with L, $N = 2^L$.

To do a forward FFT, call FFT; to do an inverse FFT, call IFFT. In both cases, the output of the transform overwrites the input data.

INCREASING THE MAXIMUM TRANSFORM LENGTH

The maximum transform length for the FFT program is primarily limited by the size of the Twiddle Factor table. To increase the transform length, the following needs to be done.

1. Increase the Twiddle Factor table. Thus, if the maximum transform length required is 1024, the table needs to store the cosine and sine of the angles

   $$0, 2\pi/1024, 2\pi \times 2/1024, \ldots, 2\pi \times 1023/1024$$

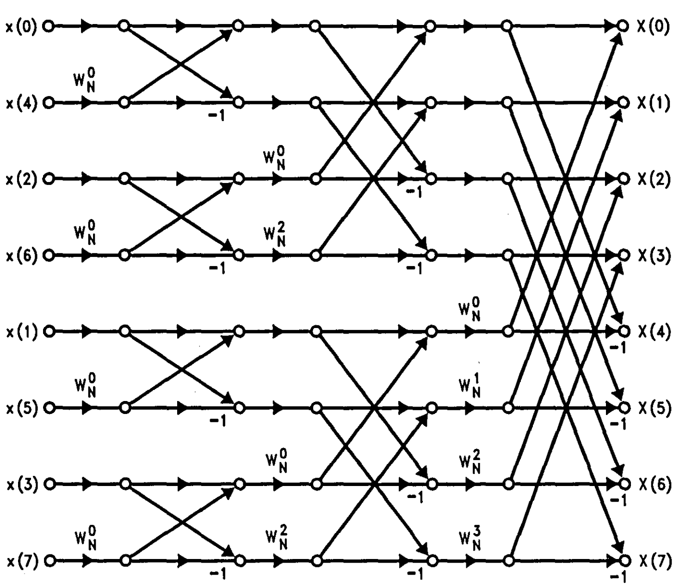
2. Change the global symbol LMAX such that the maximum transform length is 2^{LMAX}.

FFT/IFFT TEST PROGRAM

The data in the file TSTDAT.MAC can be used to test the FFT program. The data and its transform value is from reference 3. The program in reference 3 is for a Floating point FORTRAN FFT program. Since the HPC FFT program is a fixed point one, the input data needs to be suitably scaled. The scale factor chosen is 2^{10}. The file TSTDAT.MAC contains the scaled input data, and the expected transform. The input data is stored in memory words 200/27E and the expected transform is stored in memory words 280/2FE. To run the test program, do the following.

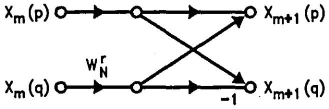
Set up the MOLE Development System with Blocks 0, 13, 14 and 15 mapped ON. Download the program to the MOLE. Set up a Breakpoint at F410. Run the program starting at F400. When the program is breakpointed, list memory words 200/27F and compare them with memory words 280/2FE.

Note that any difference between the expected DFT values and the DFT values actually computed is due to the fixed point computations in the FFT program.



TL/DD/9259-1

FIGURE 1. FFT Flow Graph for N = 8 Points



TL/DD/9259-2

FIGURE 2. The Butterfly—The Basic Computation Unit in the FFT

REFERENCES

1. A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
2. L.R. Rabiner and B. Gold, *Theory and Applications of Digital Signal Processing*, Prentice-Hall, New Jersey, 1975.
3. IEEE ASSP Society Digital Signal Processing Committee, *Programs for Digital Signal Processing*, IEEE Press, New York, 1979.

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be down loaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

APPENDIX A

Listing of FFT Program Code

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



              ;
              ; THIS PROGRAM IMPLEMENTS A RADIX-2, DECIMATION IN TIME FFT ALGORITHM.
              ;
              ;
0008                         LMAX = 08                 ; MAXIMUM FFT LENGTH IS 2^LMAX.
                                                       ;
F000                         TWSTAD = 0F000            ; TWIDDLE FACTOR TABLE START
                                                       ; ADDRESS.
0200                         DTSTAD = 0200             ; DATA STORAGE AREA START
                                                       ; ADDRESS.
              ;
01C0                         NUMB = W(01C0)            ; NUMBER OF DATA POINTS TO BE
                                                       ; TRANSFORMED.
01C2                         L1 = W(01C2)              ; NUMB IS 2^L1.
01C4                         LSHIFT = W(01C4)          ; LSHIFT = LMAX - L1. IT IS A SHIFT
                                                       ; FACTOR NEEDED TO COMPUTE THE
                                                       ; ADDRESS REQUIRED FOR TWIDDLE
                                                       ; FACTOR LOCKUP.
01C6                         NBFLY = W(01C6)           ; NUMBER OF BUTTERFLIES PER
                                                       ; TWIDDLE FACTOR PER STAGE.
01C8                         ISTEP = W(01C8)           ; IF X(1) AND X(J) ARE INVOLVED
                                                       ; IN A BUTTERFLY, THEN J=I+ISTEP.
                                                       ; IT IS ALSO THE NUMBER OF TWIDDLE
                                                       ; FACTORS IN A STAGE.
01CA                         ILEAP = W(01CA)           ; IF X(I) IS THE FIRST DATA VALUE
                                                       ; FOR THE FIRST BUTTERFLY FOR A
                                                       ; GIVEN TWIDDLE FACTOR, THEN THE
                                                       ; SUBSEQUENT BUTTERFLIES FOR THAT
                                                       ; TWIDDLE FACTOR HAVE AS THE FIRST
                                                       ; DATA VALUE X(I+N*ILEAP).
01CC                         WESTEP = W(01CC)          ; TWIDDLE FACTOR EXPONENT STEP.
                                                       ; THE TWIDDLE FACTORS FOR A GIVEN
                                                       ; STAGE ARE W^(I*WESTEP).
01CE                         NSTG = W(01CE)            ; FFT STAGE BEING EVALUATED.
                                                       ;
01D0                         ISTART = W(01D0)          ; INDEX OF THE FIRST DATA VALUE
                                                       ; FOR THE FIRST BUTTERFLY FOR A
                                                       ; GIVEN TWIDDLE FACTOR.
01D2                         WEXP = W(01D2)            ; EXPONENT VALUE FOR A GIVEN
                                                       ; TWIDDLE FACTOR.
01D4                         NTWD = W(01D4)            ; TWIDDLE FACTOR BEING EVALUATED.
                                                       ;
01D6                         COSTH = W(01D6)           ; COSINE PART OF TWIDDLE FACTOR.
                                                       ;
01D8                         SINTH = W(01D8)           ; SINE PART OF TWIDDLE FACTOR.
                                                       ;
01DA                         M1 = W(01DA)              ; INDEX OF FIRST DATA VALUE FOR
                                                       ; A BUTTERFLY.
01DC                         NBCNT = W(01DC)           ; BUTTERFLY BEING EVALUATED.
                                                       ;
01DE                         R1ADDR = W(01DE)          ; ADDRESS OF REAL PART OF FIRST
```

```
                                                            ; DATA VALUE INVOLVED IN A BUTTERFLY.
     01E0                           R2ADDR = W(01E0)        ; ADDRESS OF REAL PART OF SECOND
                                                            ; DATA VALUE INVOLVED IN A BUTTERFLY.
     01E2                           XR1 = W(01E2)           ; REAL PART OF FIRST DATA VALUE
                                                            ; INVOLVED IN A BUTTERFLY.
     01E4                           XI1= W(01E4)            ; IMAGINARY PART OF ABOVE.
                                                            ;
     01E6                           XR2 = W(01E6)           ; REAL PART OF SECOND DATA VALUE
                                                            ; INVOLVED IN A BUTTERFLY.
     01E8                           XI2 = W(01E8)           ; IMAGINARY PART OF ABOVE.
                                                            ;
     01EA                           TEMPR = W(01EA)         ; TEMPORARY STORAGE USED IN
                                                            ; A BUTTERFLY.
     01EC                           TEMPI = W(01EC)         ; SAME AS ABOVE.
                                                            ;
     01EE                           MTEMP = W(01EE)         ; TEMPORARY STORAGE USED IN SMULT.
                    ;
                                    .INCLD TSTDAT.MAC
                                    .INCLD TWDTBL.MAC
                    ;
                    ;
     F400                           . = 0F400
                    TSTFFT:
F400 B701F0C4                       LD SP, 01F0
F404 832001C0AB                     LD NUMB, 020            ; 32 POINT FFT.
F409 830501C2AB                     LD L1, 05               ; 32 = 2^5.
F40E 3049                           JSR FFT                 ; COMPUTE FFT.
F410 40                             NOP
F411 31C4                           JSR IFFT
F413 40                             NOP
F414 61                             JP .-1
                    ;
                    ;
                    ; THIS SUBROUTINE IMPLEMENTS A BIT REVERSED COUNTER AS NEEDED FOR
                    ; DATA SHUFFLING IN THE FFT ROUTINE. THE ALGORITHM IS BASED ON
                    ; THE DESCRIPTION IN:
                    ;       RABINER AND GOLD,
                    ; THEORY AND APPLICATIONS OF DIGITAL SIGNAL PROCESSING,
                    ;       PRENTICE-HALL, 1975.
                    ;
                    ; ON INPUT, X CONTAINS THE PREVIOUS BIT REVERSED COUNTER VALUE.
                    ; THE NEXT BIT REVERSED OUTPUT IS RETURNED IN X.
                    ; A IS LOST, B AND K ARE PRESERVED.
                    ;
                                    .LOCAL
                    BRCNTR:
F415 B601C0A8                       LD A, NUMB              ; GET NUMBER OF DATA SAMPLES
                                                            ; TO BE TRANSFORMED.
F419 C7                             SHR A                   ; DIVIDE BY 2.
     $REPEAT:
F41A 96CEFD                         IFGT A, X               ; IS BIT BEING TESTED A 0 ?
```

```
F41D 47                             JP $FOUND               ; YES, SO STOP CHECKING.
                                                            ; GET HERE MEANS BIT BEING
                                                            ; CHECKED IS 1.
F41E 02                             SET C
F41F A0C8CEE8                       SUBC X, A               ; ZERO OUT THE BIT.
F423 C7                             SHR A                   ; UPDATE BIT LOCATOR.
F424 6A                             JP $REPEAT
                    $FOUND:
F425 A0C8CEF8                       ADD X, A
F429 3C                             RET
                                    .LOCAL
                    ;
                    ;
                    ;
                    ; THIS SUBROUTINE MULTIPLIES TWO 16-BIT 2'S COMPLEMENT INTEGERS AND RETURNS
                    ; THE UPPER HALF OF THE RESULT. THE MULTIPLICAND IS IN A, AND THE MULTIPLIER
                    ; IN W(B). THE RESULT IS RETURNED IN A. ONE TEMPORARY WORD OF STORAGE,
                    ; ADDRESSED AS MTEMP IS USED.
                    ;
                    SMULT:
F42A 830001EEAB                     LD MTEMP, 0             ; CLEAR TEMPORARY STORAGE.
F42F A9CC                           INC B                   ; B NOW POINTS TO UPPER BYTE
                                                            ; OF MULTIPLIER.
F431 17                             IF M(B).7               ; IS IT NEGATIVE ?
F432 B601EEAB                       ST A, MTEMP             ; THEN SAVE MULTIPLICAND IN MTEMP.
F436 AACC                           DECSZ B                 ; B INTO WORD POINTER.
F438 40                             NOP
F439 B601EEAE                       X A, MTEMP              ; SWAP A AND MTEMP.
F430 B601EF17                       IF M(($MTEMP)+1).7      ; IS MULTIPLICAND NEGATIVE ?
F441 F8                             ADD A, W(B)             ; THEN ACCUMULATE MULTIPLIER.
F442 B601EEAE                       X A, MTEMP
F446 FE                             MULT A, W(B)            ; UNSIGNED MULTIPLY.
F447 AECE                           X A, X                  ; UPPER HALF IN A.
F449 02                             SET C
F44A B601EEEB                       SUBC A, MTEMP
F44E E7                             SHL A
F44F 96CF17                         IF H(X).7
F452 04                             INC A
F453 E7                             SHL A
F454 96CF16                         IF H(X).6
F457 04                             INC A
F458 3C                             RET
                    ;
                    ;
                    ;
                    ; THIS SUBROUTINE IMPLEMENTS THE FIXED POINT RADIX-2 DECIMATION-IN-TIME
                    ; FFT ALGORITHM. THE DATA IS INITIALLY PUT IN THE BIT REVERSED ORDER, AND
                    ; THEN THE FFT IS COMPUTED. FOR THE THEORY BEHIND THE ALGORITHM, CONSULT:
                    ;
                    ;       1. OPPENHEIM AND SCHAFER, DIGITAL SIGNAL PROCESSING,
                                    PRENTICE-HALL.
```

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     4
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

                    ;
                    ;       2. RABINER AND GOLD, THEORY AND APPLICATIONS OF DIGITAL SIGNAL
                    ;               PROCESSING, PRENTICE-HALL, 1975.
                    ;
                    ; THE ALGORITHM USED CLOSELY FOLLOWS THE FORTRAN PROGRAM FOREA IN
                    ;
                    ;       3. PROGRAMS FOR DIGITAL SIGNAL PROCESSING, IEEE.
                    ;
                    ;
                    FFT:
                    ;
                    ; FIRST PUT THE DATA IN BIT REVERSED ORDER.
                    ;
F459 00                             CLR A
F45A ABCC                           ST A, B                 ; SET UP NORMAL COUNTER.
F45C ABCE                           ST A, X                 ; SET UP BIT REVERSED COUNTER.
F45E A401C0CAAB                     LD K, NUMB              ; K HAS NUMBER OF DATA POINTS.
                    REVLP:
F463 A0CCCEFD                       IFGT X, B               ; IS BIT REV CNTR → NORM CNTR ?
F467 42                             JP SWAP                 ; YES, SO SWAP DATA.
F468 9421                           JMP COUNT               ; NO SO INCREMENT COUNT.
                    SWAP:
F46A AFCC                           PUSH B
F46C AFCE                           PUSH X
F46E A8CC                           LD A, B                 ; INDEX VALUE I IS IN A.
F470 E7                             SHL A
F471 E7                             SHL A
F472 B80200                         ADD A, DTSTAD           ; GET ADDR. OF XR(I).
F475 ABCC                           ST A, B                 ; SAVE IT IN B.
F477 A8CE                           LD A, X                 ; INDEX VALUE J IS IN A.
F479 E7                             SHL A
F47A E7                             SHL A
F47B B80200                         ADD A, DISTAD           ; GET ADDR. OF XR(J).
F47E ABCE                           ST A, X                 ; SAVE IT IN X.
F480 E4                             LD A, W(B)              ; A ← XR(I).
F481 F1                             X A, W(X+)              ; A ← XR(J), XR(J) ← XR(I).
F482 E1                             XS A, W(B+)             ; A ← XR(I), XR(I) ← XR(J).
F483 40                             NOP
F484 E4                             LD A, W(B)              ; A ← XI(I).
F485 F5                             X A, W(X)               ; A ← XI(J), XI(J) ← XI(I).
F486 E6                             ST A, W(B)              ; XI(I) ← XI(J).
F487 3FCE                           POP X
F489 3FCC                           POP B
                    ;
                    COUNT:
                    ;
F48B AACA                           DECSZ K                 ; DONE ?
F48D 41                             JP UPIT                 ; NO GO DO SOME MORE.
F48E 46                             JP DOFFT
                    UPIT:
F48F 347A                           JSR BRCNTR              ; COUNT UP ON BIT REV CNTR.
```

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86


F491 A9CC                         INC B                   ; COUNT UP ON NORMAL CNTR.
F493 9530                         JMP REVLP
                    DOFFT:
                    ;
                    ; DATA IS NOW STORED IN THE BIT REVERSED ORDER. COMPUTE THE FFT.
                    ;
F495 9008                         LD A, LMAX              ; A HAS MAX FFT EXPONENT.
F497 04                           INC A
F498 04                           INC A
F499 02                           SET C
F49A B601C2EB                     SUBC A, L1              ; COMPUTE LSHIFT.
F49E B601C4AB                     ST A, LSHIFT
F4A2 B601C0A8                     LD A, NUMB
F4A6 C7                           SHR A
F4A7 B601C6AB                     ST A, NBFLY             ; INITIALIZE NBFLY.
F4AB B601CCAB                     ST A, WESTEP            ; INITIALIZE WESTEP.
F4AF 830101C8AB                   LD ISTEP, 01            ; INITIALIZE ISTEP.
F4B4 830201CAAB                   LD ILEAP, 02            ; INITIALIZE ILEAP.
                    ;
                    ; SET UP L1 STAGES OF BUTTERFLIES.
                    ;
F4B9 B601C2AB                     LD A, L1
F4BD B601CEAB                     ST A, NSTG              ; LOOP L1 TIMES.
                    LOOP1:
                    ;
F4C1 00                           CLR A
F4C2 B601D0AB                     ST A, ISTART            ; INITIALIZE ISTART FOR EACH STAGE.
F4C6 B601D2AB                     ST A, WEXP              ; INITIALIZE WEXP.
                    ;
                    ; SET UP ISTEP LOOPS OF TWIDDLE FACTORS.
                    ;
F4CA B601C8A8                     LD A, ISTEP
F4CE B601D4AB                     ST A, NTWD              ; LOOP ISTEP TIMES.
                    LOOP2:
                    ;
                    ; LOOK UP THE TWIDDLE FACTOR.
                    ;
F4D2 A401C4CAAB                   LD K, LSHIFT            ; SHIFT LEFT LSHIFT TIMES.
F4D7 B601D2A8                     LD A, WEXP
                    GADLP:
F4DB E7                           SHL A
F4DC AACA                         DECSZ K                 ; DONE SHIFTING ?
F4DE 63                           JP GADLP                ; NO SO DO MORE.
F4DF B8F000                       ADD A, TWSTAD           ; ADD STARTING ADDR OF TWIDDLE
                                                          ; FACTOR TABLE.
F4E2 ABCE                         ST A, X                 ; TWIDDLE FACTOR ADDR IN X.
F4E4 F0                           LD A, W(X+)             ; GET COS(THETA).
F4E5 B601D6AB                     ST A, COSTH
F4E9 F4                           LD A, W(X)              ; GET SIN(THETA).
F4EA 01                           COMP A
F4EB 04                           INC A                   ; MAKE IT NEGATIVE.
```

```
F4EC B601D8AB                       ST A, SINTH
                    ;
F4F0 A501D001DAAB                   LD M1, ISTART           ; INITIALIZE M1.
                    ;
                    ; SET UP NBFLY BUTTERFLIES FOR THIS TWIDDLE FACTOR.
                    ;
F4F6 A501C601DCAB                   LD NBCNT, NBFLY         ; LOOP NBFLY TIMES.
                    LOOP3:
F4FC B601DAAB                       LD A, M1                ; GET INDEX OF X(I).
F500 E7                             SHL A
F501 E7                             SHL A
F502 B80200                         ADD A, DTSTAD           ; ADDR. OD XR(I).
F505 B601DEAB                       ST A, R1ADDR
F509 ABCE                           ST A, X
F50B F0                             LD A, W(X+)             ; A ← XR(I).
F50C B601E2AB                       ST A, XR1               ; STORE IN XR1.
F510 F4                             LD A, W(X)              ; A ← XI(I).
F511 B601E4AB                       ST A, XI1               ; STORE IN XI1.
F515 B601DAA8                       LD A, M1
F519 B601C8F8                       ADD A, ISTEP            ; GET INDEX OF X(J).
F51D E7                             SHL A
F51E E7                             SHL A
F51F B80200                         ADD A, DISTAD           ; ADDR. OF XR(J).
F522 B601E0AB                       ST A, R2ADDR
F526 ABCE                           ST A, X
F528 F0                             LD A, W(X+)             ; A ← XR(J).
F529 B601E6AB                       ST A, XR2               ; STORE IN XR2.
F520 F4                             LD A, W(X)              ; A ← XI(J).
F52E B601E8AB                       ST A, XI2               ; STORE IN XI2.
                    ;
F532 B201E6                         LD B, #XR2              ; B ← ADDR(XR2).
F535 B601D6A8                       LD A, COSTH             ; A ← COS(THETA).
F539 350F                           JSR SMULT               ; COMPUTE XR(J)*COS(THETA).
F53B B601EAAB                       ST A, TEMPR             ; SAVE IN TEMPR.
F53F B601D8A8                       LD A, SINTH             ; A ← SIN(THETA).
F543 3519                           JSR SMULT               ; COMPUTE XR(J)*SIN(THETA).
F545 B601ECAB                       ST A, TEMPI             ; SAVE IN TEMPI.
F549 B201E8                         LD B, #XI2              ; B ← ADDR(XI2).
F54C B601D8A8                       LD A, SINTH             ; A ← SIN(THETA).
F550 3526                           JSR SMULT               ; COMPUTE XI(J)*SIN(THETA).
F552 01                             COMP A
F553 04                             INC A
F554 B601EAF8                       ADD A, TEMPR            ; COMPUTE XR(J)*COS(THETA)-
                                                            ; XI(J)*SIN(THETA).
F558 B601EAAB                       ST A, TEMPR
F55C B601D6A8                       LD A, COSIN             ; A ← COS(THETA).
F560 3536                           JSR SMULT               ; COMPUTE XI(J)*COS(THETA).
F562 B601ECF8                       ADD A, TEMPI            ; COMPUTE XR(J)*SIN(THETA)+
                                                            ; XI(J)*COS(THETA).
F566 B601ECAB                       ST A, TEMPI
                    ;
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



307                         ;
308 F56A A401DECEAB                     LD X, R1ADDR              ; X ← ADDR(XR(I)).
309 F56F A401E0CCAB                     LD B, R2ADDR              ; B ← ADDR(XR(J)).
310 F574 F0                             LD A, W(X+)               ; A ← XR(I).
311 F575 02                             SET C
312 F576 B601EAEB                       SUBC A, TEMPR             ; A ← XR(I) - TEMPR.
313 F57A E1                             XS A, W(B+)
314 F57B 40                             NOP
315 F57C F2                             LD A, W(X-)               ; A ← XI(I).
316 F57D 02                             SET C
317 F57E B601ECEB                       SUBC A, TEMPI             ; A ←XI(J) - TEMPI.
318 F582 E6                             ST A, W(B)
319 F583 F4                             LD A, W(X)                ; A ← XR(I).
320 F584 B601EAF8                       ADD A, TEMPR              ; A ← XR(I) + TEMPR.
321 F588 F1                             X A, W(X+)
322 F589 F4                             LD A, W(X)                ; A ← XI(I).
323 F58A B601ECF8                       ADD A, TEMPI              ; A ← XI(I) + TEMPI.
324 F58E F6                             ST A, W(X)
325                         ;
326 F58F A501CA01DAF8                   ADD M1, ILEAP             ; UPDATE M1 FOR NEXT LOOP.
327                         ;
328 F595 B601DCAA                       DECSZ NBCNT               ; DONE WITH ALL BUTTERFLIES
329                                                               ; FOR THIS TWIDDLE FACTOR ?
330 F599 959D                           JMP LOOP3                 ; NO, SO GO DO SOME MORE.
331                         ;
332                         ;
333 F59B B601D0A9                       INC ISTART                ; SET UP STARTING INDEX FOR
334                                                               ; NEXT TWIDDLE FACTOR.
335 F59F A501CC01D2F8                   ADD WEXP, WESTEP          ; UPDATE TWIDDLE FACTOR
336                                                               ; EXPONENT VALUE.
337                         ;
338 F5A5 B601D4AA                       DECSZ NTWD                ; DONE WITH ALL TWIDDLES
339                                                               ; FOR THIS STAGE ?
340 F5A9 95D7                           JMP LOOP2                 ; NO, SO GO DO SOME MORE.
341                         ;
342                         ;
343 F5AB B601CAA8                       LD A, ILEAP
344 F5AF E7                             SHL A
345 F5B0 B601CAAB                       ST A, ILEAP               ; UPDATE ILEAP FOR NEXT STAGE.
346 F5B4 B601C8A8                       LD A, ISTEP
347 F5B8 E7                             SHL A
348 F5B9 B601C8AB                       ST A, ISTEP               ; UPDATE ISTEP FOR NEXT STAGE.
349 F58D B601C6A8                       LD A, NBFLY
350 F5C1 C7                             SHR A
351 F5C2 B601C6AB                       ST A, NBFLY               ; UPDATE NBFLY FOR NEXT STAGE.
352 F5C6 B601CCA8                       LD A, WESTEP
353 F5CA C7                             SHR A
354 F5CB B601CCAB                       ST A, WESTEP              ; UPDATE WESTEP FOR NEXT STAGE.
355                         ;
356 F5CF B601CEAA                       DECSZ NSTG                ; DONE WITH ALL STAGES ?
357 F5D3 B4FEEB           +             JMP LOOP1                 ; NO SO GO DO SOME MORE.
```

```
                       ;
F5D6 3C                                 RET                     ; ALL OVER.
                       ;
                       ; THE CODE BELOW IS FOR THE INVERSE FFT. THE ONLY DIFFERENCE IS THAT
                       ; THE TWIDDLE FACTORS ARE USED A LITTLE DIFFERENTLY, AND A FINAL SCALING BY
                       ; 1/NUMB IS DONE.
                       IFFT:
                       ;
                       ; FIRST PUT THE DATA IN BIT REVERSED ORDER.
                       ;
F5D7 00                                 CLR A
F5D8 ABCC                               ST A, B                 ; SET UP NORMAL COUNTER.
F5DA ABCE                               ST A, X                 ; SET UP BIT REVERSED COUNTER.
F5DC A401C0CAAB                         LD K, NUMB              ; K HAS NUMBER OF DATA POINTS.
                       IREVLP:
F5E1 A0CCCEFD                           IFGT X, B               ; IS BIT REV CNTR → NORM CNTR ?
F5E5 42                                 JP ISWAP                ; YES, SO SWAP DATA.
F5E6 9421                               JMP ICOUNT              ; NO SO INCREMENT COUNT.
                       ISWAP:
F5E8 AFCC                               PUSH B
F5EA AFCE                               PUSH X
F5EC A8CC                               LD A, B                 ; INDEX VALUE I IS IN A.
F5EE E7                                 SHL A
F5EF E7                                 SHL A
F5F0 B80200                             ADD A, DTSTAD           ; GET ADDR. OF XR(I).
F5F3 ABCC                               ST A, B                 ; SAVE IT IN B.
F5F5 A8CE                               LD A, X                 ; INDEX VALUE J IS IN A.
F5F7 E7                                 SHL A
F5F8 E7                                 SHL A
F5F9 B80200                             ADD A, DTSTAD           ; GET ADDR. OF XR(J).
F5FC ABCE                               ST A, X                 ; SAVE IT IN X.
F5FE E4                                 LD A, W(B)              ; A ← XR(I).
F5FF F1                                 X A, W(X+)              ; A ← XR(J), XR(J) ← XR(I).
F600 E1                                 XS A, W(B+)             ; A ← XR(I), XR(I) ← XR(J).
F601 40                                 NOP
F602 E4                                 LD A, W(B)              ; A ← XI(I).
F603 F5                                 X A, W(X)               ; A ← XI(J), XI(J) ← XI(I).
F604 E6                                 ST A, W(B)              ; XI(I) ← XI(J).
F605 3FCE                               POP X
F607 3FCC                               POP B
                       ;
                       ICOUNT:
                       ;
F609 AACA                               DECSZ K                 ; DONE ?
F60B 41                                 JP IUPIT                ; NO GO DO SOME MORE.
F60C 46                                 JP DOIFFT
                       IUPIT:
F60D 35F8                               JSR BRCNTR              ; COUNT UP ON BIT REV CNTR.
F60F A9CC                               INC B                   ; COUNT UP ON NORMAL CNTR.
F611 9530                               JMP IREVLP
                       DOIFFT:
```

```
                    ;
                    ; DATA IS NOW STORED IN THE BIT REVERSED ORDER. COMPUTE THE FFT.
                    ;
F613 9008                           LD A, LMAX              ; A HAS MAX FFT EXPONENT.
F615 04                             INC A
F616 04                             INC A
F617 02                             SET C
F618 B601C2EB                       SUBC A, L1              ; COMPUTE LSHIFT.
F61C B601C4AB                       ST A, LSHIFT
F620 B601C0A8                       LD A, NUMB
F624 C7                             SHR A
F625 B601C6AB                       ST A, NBFLY             ; INITIALIZE NBFLY.
F629 B601CCAB                       ST A, WESTEP            ; INITIALIZE WESTEP.
F62D 830101C8AB                     LD ISTEP, 01            ; INITIALIZE ISTEP.
F632 830201CAAB                     LD ILEAP, 02            ; INITIALIZE ILEAP.
                    ;
                    ; SET UP L1 STAGES OF BUTTERFLIES.
                    ;
F637 B601C2AB                       LD A, L1
F63B B601CEAB                       ST A, NSTG              ; LOOP L1 TIMES.
                    ILOOP1:
                    ;
F63F 00                             CLR A
F640 B601D0AB                       ST A, ISTART            ; INITIALIZE ISTART FOR EACH STAGE.
F644 B601D2AB                       ST A, WEXP              ; INITIALIZE WEXP.
                    ;
                    ; SET UP ISTEP LOOPS OF TWIDDLE FACTORS.
                    ;
F648 B601C8A8                       LD A, ISTEP
F64C B601D4AB                       ST A, NTWD              ; LOOP ISTEP TIMES.
                    ILOOP2:
                    ;
                    ; LOOK UP THE TWIDDLE FACTOR.
                    ;
F650 A401C4CAAB                     LD K, LSHIFT            ; SHIFT LEFT LSHIFT TIMES.
F655 B601D2AB                       LD A, WEXP
                    IGADLP:
F659 E7                             SHL A
F65A AACA                           DECSZ K                 ; DONE SHIFTING ?
F65C 63                             JP IGADLP               ; NO DO SOME MORE.
F65D B8F000                         ADD A, TWSTAD           ; ADD STARTING ADDR OF TWIDDLE
                                                            ; FACTOR TABLE.
F660 ABCE                           ST A, X                 ; TWIDDLE FACTOR ADDR IN X.
F662 F0                             LD A, W(X+)             ; GET COS(THETA).
F663 B601D6AB                       ST A, COSTH
F667 F4                             LD A, W(X)              ; GET SIN(THETA).
F668 B601D8AB                       ST A, SINTH
                    ;
F66C A501D001DAAB                   LD M1, ISTART           ; INITIALIZE M1.
                    ;
                    ; SET UP NBFLY BUTTERFLIES FOR THIS TWIDDLE FACTOR.
```

```
NATIONAL SEMICONDUCTOR CORPORATION          PAGE:     10
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



460                        ;
461 F672 A501C601DCAB                   LD NBCNT, NBFLY         ; LOOP NBFLY TIMES.
462                        ILOOP3:
463 F678 B601DAA8                       LD A, M1                ; GET INDEX OF X(I).
464 F67C E7                             SHL A
465 F67D E7                             SHL A
466 F67E B80200                         ADD A, DTSTAD           ; ADDR. OD XR(I).
467 F681 B601DEA8                       ST A, R1ADDR
468 F685 ABCE                           ST A, X
469 F687 F0                             LD A, W(X+)             ; A ← XR(I).
470 F688 B601E2AB                       ST A, XR1               ; STORE IN XR1.
471 F68C F4                             LD A, W(X)              ; A ← XI(I).
472 F68D B601E4AB                       ST A, XI1               ; STORE IN XI1.
473 F691 B601DAA8                       LD A, M1
474 F695 B601C8F8                       ADD A, ISTEP            ; GET INDEX OF X(J).
475 F699 E7                             SHL A
476 F69A E7                             SHL A
477 F69B B80200                         ADD A, DTSTAD           ; ADDR. OF XR(J).
478 F69E B601E0AB                       ST A, R2ADDR
479 F6A2 ABCE                           ST A, X
480 F6A4 F0                             LD A, W(X+)             ; A ← XR(J).
481 F6A5 B601E6AB                       ST A, XR2               ; STORE IN XR2.
482 F6A9 F4                             LD A, W(X)              ; A ← XI(J).
483 F6AA B601E8AB                       ST A, XI2               ; STORE IN XI2.
484                        ;
485 F6A8 B201E6                         LD B, #XR2              ; B ← ADDR(XR2).
486 F6B1 B601D6A8                       LD A, COSTH             ; A ← COS(THETA).
487 F6B5 368B                           JSR SMULT               ; COMPUTE XR(J)*COS(THETA).
488 F6B7 B601EAAB                       ST A, TEMPR             ; SAVE IN TEMPR.
489 F6BB B601D8A8                       LD A, SINTH             ; A ← SIN(THETA).
490 F6BF 3695                           JSR SMULT               ; COMPUTE XR(J)*SIN(THETA).
491 F6C1 B601ECAB                       ST A, TEMPI             ; SAVE IN TEMPI.
492 F6C5 B201E8                         LD B, #XI2              ; B ← ADDR(XI2).
493 F6C8 B601D8A8                       LD A, SINTH             ; A ← SIN(THETA).
494 F6CC 36A2                           JSR SMULT               ; COMPUTE XI(J)*SIN(THETA).
495 F6CE 01                             COMP A
496 F6CF 04                             INC A
497 F6D0 B601EAF8                       ADD A, TEMPR            ; COMPUTE XR(J)*COS(THETA) -
498                                                             ; XI(J)*SIN(THETA).
499 F6D4 B601EAAB                       ST A, TEMPR
500 F6D8 B601D6A8                       LD A, COSTH             ; A ← COS(THETA).
501 F6DC 36B2                           JSR SMULT               ; COMPUTE XI(J)*COS(THETA).
502 F6DE B601ECF8                       ADD A, TEMPI            ; COMPUTE XR(J)*SIN(THETA) +
503                                                             ; XI(J)*COS(THETA).
504 F6E2 B601ECAB                       ST A, TEMPI
505                        ;
506                        ;
507 F6E6 A401DECEAB                     LD X, R1ADDR            ; X ← ADDR(XR(I)).
508 F6EB A401E0CCAB                     LD B, R2ADDR            ; B ← ADDR(XR(J)).
509 F6F0 F0                             LD A, W(X+)             ; A ← XR(I).
510 F6F1 02                             SET C
```

```
F6F2 B601EAEB                       SUBC A, TEMPR            ; A ← XR(I) - TEMPR.
F6F6 E1                             XS A, W(B+)
F6F7 40                             NOP
F6F8 F2                             LD A, W(X-)              ; A ← XI(I).
F6F9 02                             SET C
F6FA B601ECEB                       SUBC A, TEMPI            ; A ← XI(J) - TEMPI.
F6FE E6                             ST A, W(B)
F6FF F4                             LD A, W(X)               ; A ← XR(I).
F700 B601EAF8                       ADD A, TEMPR             ; A ← XR(I) + TEMPR.
F704 F1                             X A, W(X+)
F705 F4                             LD A, W(X)               ; A ← XI(I).
F706 B601ECF8                       ADD A, TEMPI             ; A ← XI(I) + TEMPI.
F70A F6                             ST A, W(X)
                     ;
F70B A501CA01DAF8                   ADD M1, ILEAP            ; UPDATE M1 FOR NEXT LOOP.
                     ;
F711 B601DCAA                       DECSZ NBCNT              ; DONE WITH ALL BUTTERFLIES
                                                             ; FOR THIS TWIDDLE FACTOR ?
F715 959D                           JMP ILOOP3               ; NO, SO GO DO SOME MORE.
                     ;
                     ;
F717 B601D0A9                       INC ISTART               ; SET UP STARTING INDEX FOR
                                                             ; NEXT TWIDDLE FACTOR.
F71B A501CC01D2F8                   ADD WEXP, WESTEP         ; UPDATE TWIDDLE FACTOR
                                                             ; EXPONENT VALUE.
                     ;
F721 B601D4AA                       DECSZ NTWD               ; DONE WITH ALL TWIDDLES
                                                             ; FOR THIS STAGE ?
F725 95D5                           JMP ILOOP2               ; NO, SO GO DO SOME MORE.
                     ;
                     ;
F727 B601CAA8                       LD A, ILEAP
F72B E7                             SHL A
F72C B601CAAB                       ST A, ILEAP              ; UPDATE ILEAP FOR NEXT STAGE.
F730 B601C8A8                       LD A, ISTEP
F734 E7                             SHL A
F735 B601C8AB                       ST A, ISTEP              ; UPDATE ISTEP FOR NEXT STAGE.
F739 B601C6A8                       LD A, NBFLY
F73D C7                             SHR A
F73E B601C6AB                       ST A, NBFLY              ; UPDATE NBFLY FOR NEXT STAGE.
F742 B601CCA8                       LD A, WESTEP
F746 C7                             SHR A
F747 B601CCAB                       ST A, WESTEP             ; UPDATE WESTEP FOR NEXT STAGE.
                     ;
F748 B601CEAA                       DECSZ NSTG               ; DONE WITH ALL STAGES ?
F74F B4FEED        +                JMP ILOOP1               ; NO SO GO DO SOME MORE.
                     ;
                     ;
                     ; DO THE FINAL SCALING OF THE DATA BY 1/NUMB.
                     ;
F752 B04000                         LD A, 04000              ; A ← 1.0
```

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     12
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



562 F755 A401C2CAAB                    LD K, L1                  ; K ← L1.
563                     SCALLP:
564 F75A C7                            SHR A                     ; DIVIDE BY 2.
565 F75B AACA                          DECSZ K
566 F75D 63                            JP SCALLP
567                                                              ;
568                                                              ; GET HERE MEANS A IS 1/(2^L1).
569 F75E B601EAAB                      ST A, TEMPR               ; SAVE IT IN TEMPR.
570 F762 A501C001DCAB                  LD NBCNT, NUMB            ; LOOP COUNTER.
571 F768 B20200                        LD B, DTSTAD              ; B ← ADDR(XR(0)).
572                     SCALIT:
573 F76B B601EAA8                      LD A, TEMPR
574 F76F 3745                          JSR SMULT                 ; A ← XR(I)*(1/NUMB).
575 F771 E1                            XS A, W(B+)               ; XR(I) ← XR(I)*(1/NUMB).
576 F772 40                            NOP
577 F773 B601EAA8                      LD A, TEMPR               ; A ← 1/NUMB.
578 F777 374D                          JSR SMULT                 ; A ← XI(I)*(1/NUMB).
579 F779 E1                            XS A, W(B+)               ; XI(I) ← XI(I)*(1/NUMB).
580 F77A 40                            NOP
581 F77B B601DCAA                      DECSZ NBCNT               ; DONE ?
582 F77F 74                            JP SCALIT                 ; NO DO SOME MORE.
583 F780 3C                            RET                       ; ALL OVER.
584                     ;
585 FFFE 00F4                          .END TSTFFT
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     13
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

SYMBOL TABLE


A      00C8 W      B      00CC W      BRCNTR F415        COSTN  01D6 W
COUNT  F48B        DOFFT  F495        DOIFFT F613        DTSTAD 0200
FFT    F459        GADLP  F40B        ICOUNT F609        IFFT   F507
IGADLP F659        ILEAP  01CA W      ILOOP1 F63F        ILOOP2 F650
ILOOP3 F678        IREVLP F5E1        ISTART 01D0 W      ISTEP  01C8 W
ISWAP  F5E8        IUPIT  F600        K      00CA W      L1     01C2 W
LMAX   0008        LOOP1  F4C1        LOOP2  F4D2        LOOP3  F4FC
LSHIFT 01C4 W      M1     01DA W      MTEMP  01EE W      NBCNT  01DC W
NBFLY  01C6 W      NSTG   01CE W      NTWD   01D4 W      NUMB   01C0 W
PC     00C6 W      R1ADDR 01DE W      R2ADDR 01E0 W      REVLP  F463
SCALIT F76B        SCALLP F75A        SINTH  01D8 W      SMULT  F42A
SP     00C4 W      SWAP   F46A        TEMPI  01EC W      TEMPR  01EA W
TSTFFT F400        TWSTAD F000        UPIT   F48F        WESTEP 01CC W
WEXP   01D2 W      X      00CE W      XI1    01E4 W      XI2    01E8 W
XR1    01E2 W      XR2    01E6 W      $FOUND F425        $REPEA F41A
```

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     14
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

MACRO TABLE



   NO WARNING LINES

   NO ERROR LINES

 2307  ROM BYTES USED

SOURCE CHECKSUM = E9FC
OBJECT CHECKSUM = 28FC

INPUT   FILE C:FFT.MAC
LISTING FILE C:FFT.PRN
OBJECT  FILE C:FFT.LM
```

APPENDIX B

Twiddle Factor Table

```
;
; TWIDDLE FACTOR TABLE FOR USE IN THE FFT ROUTINES.
;
; TABLE SET FOR MAX FFT LENGTH OF 256.
;
; TABLE STARTS AT F000 AND OCCUPIES 1024 BYTES OF STORAGE.
;
               . = 0F000
               .WORD 16384, 0
               .WORD 16379, 402
               .WORD 16364, 804
               .WORD 16340, 1205
               .WORD 16305, 1606
               .WORD 16261, 2006
               .WORD 16207, 2404
               .WORD 16143, 2801
               .WORD 16069, 3196
               .WORD 15986, 3590
               .WORD 15893, 3981
               .WORD 15791, 4370
               .WORD 15679, 4756
               .WORD 15557, 5139
               .WORD 15426, 5520
               .WORD 15286, 5897
               .WORD 15137, 6270
               .WORD 14978, 6639
               .WORD 14811, 7005
               .WORD 14635, 7366
               .WORD 14449, 7723
               .WORD 14256, 8076
               .WORD 14053, 8423
               .WORD 13842, 8765
               .WORD 13623, 9102
               .WORD 13395, 9434
               .WORD 13160, 9760
               .WORD 12916, 10080
               .WORD 12665, 10394
               .WORD 12406, 10702
               .WORD 12140, 11003
               .WORD 11866, 11297
               .WORD 11585, 11585
               .WORD 11297, 11866
               .WORD 11003, 12140
               .WORD 10702, 12406
               .WORD 10394, 12665
               .WORD 10080, 12916
               .WORD 9760, 13160
               .WORD 9434, 13395
               .WORD 9102, 13623
               .WORD 8765, 13842
               .WORD 8423, 14053
               .WORD 8076, 14256
               .WORD 7723, 14449
               .WORD 7366, 14635
               .WORD 7005, 14811
               .WORD 6639, 14978
               .WORD 6270, 15137
               .WORD 5897, 15286
               .WORD 5520, 15426
               .WORD 5139, 15557
               .WORD 4756, 15679
               .WORD 4370, 15791
               .WORD 3981, 15893
               .WORD 3590, 15986
               .WORD 3196, 16069
               .WORD 2801, 16143
               .WORD 2404, 16207
               .WORD 2006, 16261
               .WORD 1606, 16305
               .WORD 1205, 16340
               .WORD 804, 16364
               .WORD 402, 16379
               .WORD 0, 16384
               .WORD -402, 16379
               .WORD -804, 16364
               .WORD -1205, 16340
               .WORD -1606, 16305
               .WORD -2006, 16261
               .WORD -2404, 16207
               .WORD -2801, 16143
               .WORD -3196, 16069
               .WORD -3590, 15986
               .WORD -3981, 15893
               .WORD -4370, 15791
               .WORD -4756, 15679
               .WORD -5139, 15557
               .WORD -5520, 15426
               .WORD -5897, 15286
               .WORD -6270, 15137
               .WORD -6639, 14978
               .WORD -7005, 14811
               .WORD -7366, 14635
               .WORD -7723, 14449
               .WORD -8076, 14256
               .WORD -8423, 14053
               .WORD -8765, 13842
               .WORD -9102, 13623
               .WORD -9434, 13395
               .WORD -9760, 13160
               .WORD -10080, 12916
               .WORD -10394, 12665
               .WORD -10702, 12406
               .WORD -11003, 12140
               .WORD -11297, 11866
               .WORD -11585, 11585
               .WORD -11866, 11297
               .WORD -12140, 11003
               .WORD -12406, 10702
               .WORD -12665, 10394
               .WORD -12916, 10080
```

```
.WORD -13160, 9760
.WORD -13395, 9434
.WORD -13623, 9102
.WORD -13842, 8765
.WORD -14053, 8423
.WORD -14256, 8076
.WORD -14449, 7723
.WORD -14635, 7366
.WORD -14811, 7005
.WORD -14978, 6639
.WORD -15137, 6270
.WORD -15286, 5897
.WORD -15426, 5520
.WORD -15557, 5139
.WORD -15679, 4756
.WORD -15791, 4370
.WORD -15893, 3981
.WORD -15986, 3590
.WORD -16069, 3196
.WORD -16143, 2801
.WORD -16207, 2404
.WORD -16261, 2006
.WORD -16305, 1606
.WORD -16340, 1205
.WORD -16364, 804
.WORD -16379, 402
.WORD -16384, 0


.WORD -16379, -402
.WORD -16364, -804
.WORD -16340, -1205
.WORD -16305, -1606
.WORD -16261, -2006
.WORD -16207, -2404
.WORD -16143, -2801
.WORD -16069, -3196
.WORD -15986, -3590
.WORD -15893, -3981
.WORD -15791, -4370
.WORD -15679, -4756
.WORD -15557, -5139
.WORD -15426, -5520
.WORD -15286, -5897
.WORD -15137, -6270
.WORD -14978, -6639
.WORD -14811, -7005
.WORD -14635, -7366
.WORD -14449, -7723
.WORD -14256, -8076
.WORD -14053, -8423
.WORD -13842, -8765
.WORD -13623, -9102
.WORD -13395, -9434
.WORD -13160, -9760
```

```
.WORD -12916, -10080
.WORD -12665, -10394
.WORD -12406, -10702
.WORD -12140, -11003
.WORD -11866, -11297
.WORD -11585, -11585
.WORD -11297, -11866
.WORD -11003, -12140
.WORD -10702, -12406
.WORD -10394, -12665
.WORD -10080, -12916
.WORD -9760, -13160
.WORD -9434, -13395
.WORD -9102, -13623
.WORD -8765, -13842
.WORD -8423, -14053
.WORD -8076, -14256
.WORD -7723, -14449
.WORD -7366, -14635
.WORD -7005, -14811
.WORD -6639, -14978
.WORD -6270, -15137
.WORD -5897, -15286
.WORD -5520, -15426
.WORD -5139, -15557
.WORD -4756, -15679
.WORD -4370, -15791
.WORD -3981, -15893
.WORD -3590, -15986
.WORD -3196, -16069
.WORD -2801, -16143
.WORD -2404, -16207
.WORD -2006, -16261
.WORD -1606, -16305
.WORD -1205, -16340
.WORD -804, -16364
.WORD -402, -16379
.WORD 0, -16384
.WORD 402, -16379
.WORD 804, -16364
.WORD 1205, -16340
.WORD 1606, -16305
.WORD 2006, -16261
.WORD 2404, -16207
.WORD 2801, -16143
.WORD 3196, -16069
.WORD 3590, -15986
.WORD 3981, -15893
.WORD 4370, -15791
.WORD 4756, -15679
.WORD 5139, -15557
.WORD 5520, -15426
.WORD 5897, -15286
.WORD 6270, -15137
.WORD 6639, -14978
```

```
.WORD 7005, -14811
.WORD 7366, -14635
.WORD 7723, -14449
.WORD 8076, -14256
.WORD 8423, -14053
.WORD 8765, -13842
.WORD 9102, -13623
.WORD 9434, -13395
.WORD 9760, -13160
.WORD 10080, -12916
.WORD 10394, -12665
.WORD 10702, -12406
.WORD 11003, -12140
.WORD 11297, -11866
.WORD 11585, -11585
.WORD 11866, -11297
.WORD 12140, -11003
.WORD 12406, -10702
.WORD 12665, -10394
.WORD 12916, -10080
.WORD 13160, -9760
.WORD 13395, -9434
.WORD 13623, -9102
.WORD 13842, -8765
.WORD 14053, -8423
.WORD 14256, -8076
.WORD 14449, -7723
.WORD 14635, -7366
.WORD 14811, -7005
.WORD 14978, -6639
.WORD 15137, -6270
.WORD 15286, -5897
.WORD 15426, -5520
.WORD 15557, -5139
.WORD 15679, -4756
.WORD 15791, -4370
.WORD 15893, -3981
.WORD 15986, -3590
.WORD 16069, -3196
.WORD 16143, -2801
.WORD 16207, -2404
.WORD 16261, -2006
.WORD 16305, -1606
.WORD 16340, -1205
.WORD 16364, -804
.WORD 16379, -402

.END
```

APPENDIX C

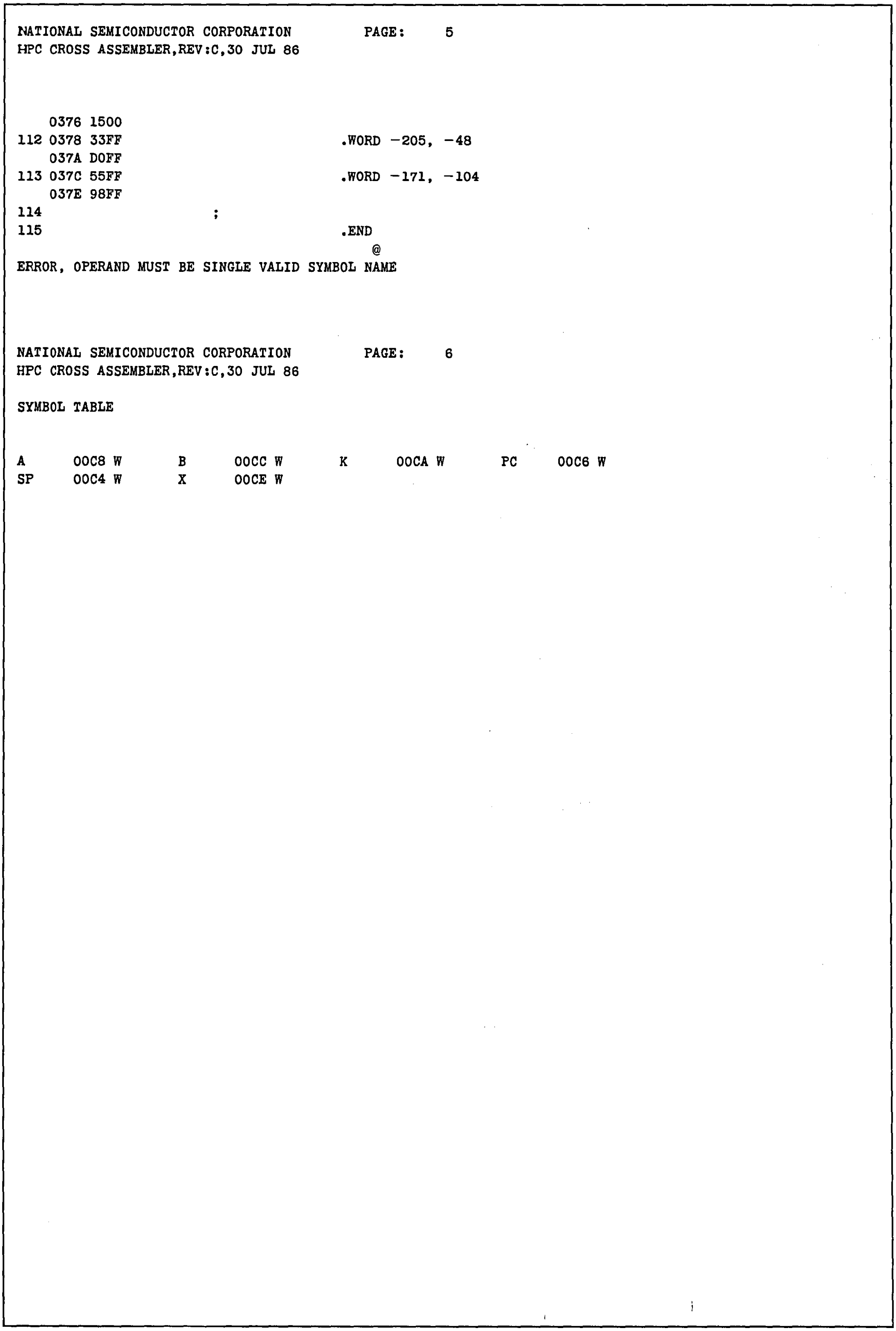
Test Data and Expected Results

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     1
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



                    ;
                    ;
                    ; TEST DATA FOR FFT ROUTINES.
                    ; OBTAINED FROM : PROGRAMS FOR DIGITAL SIGNAL PROCESSING, IEEE PRESS,
                    ; CHAPTER 1 BY GOLD.
                    ;
                    ;
     0200                               . = 0200
0200 0004                               .WORD 1024, 0
0202 0000
0204 9A03                               .WORD 922, 307
0206 3301
0208 E102                               .WORD 737, 553
020A 2902
020C F201                               .WORD 498, 719
020E CF02
0210 E800                               .WORD 232, 796
0212 1C03
0214 E2FF                               .WORD -30, 786
0216 1203
0218 F9FE                               .WORD -263, 699
021A BB02
021C 42FE                               .WORD -446, 550
021E 2602
0220 C9FD                               .WORD -567, 361
0222 6901
0224 96FD                               .WORD -618, 155
0226 9800
0228 A5FD                               .WORD -603, -46
022A D2FF
022C EFFD                               .WORD -529, -222
022E 22FF
0230 67FE                               .WORD -409, -359
0232 99FE
0234 FBFE                               .WORD -261, -446
0236 42FE
0238 9BFF                               .WORD -101, -479
023A 21FE
023C 3500                               .WORD 53, -462
023E 32FE
0240 BA00                               .WORD 186, -400
0242 70FE
0244 1F01                               .WORD 287, -304
0246 D0FE
0248 5E01                               .WORD 350, -187
024A 45FF
024C 7301                               .WORD 371, -64
024E C0FF
0250 6101                               .WORD 353, 54
0252 3600
0254 2001                               .WORD 301, 154
```

```
    0256 9A00
 31 0258 E100                         .WORD 225, 229
    025A E500
 32 025C 8600                         .WORD 134, 274
    025E 1201
 33 0260 2600                         .WORD 38, 287
    0262 1F01
 34 0264 CCFF                         .WORD -52, 269
    0266 0D01
 35 0268 81FF                         .WORD -127, 227
    026A E300
 36 026C 49FF                         .WORD -183, 166
    026E A600
 37 0270 2AFF                         .WORD -214, 95
    0272 5F00
 38 0274 23FF                         .WORD -221, 21
    0276 1500
 39 0278 33FF                         .WORD -205, -48
    027A DDFF
 40 027C 55FF                         .WORD -171, -104
    027E 98FF
 41                    ;
 42                    ; THESE ARE THE EXPECTED DFT RESULTS.
 43                    ;
 44 0280 C702                         .WORD 711, 3584
    0282 000E
 45 0284 2B0B                         .WORD 2859, 8244
    0286 3420
 46 0288 9D25                         .WORD 9629, -9354
    028A 76DB
 47 028C 7707                         .WORD 1911, -3926
    028E AAF0
 48 0290 8704                         .WORD 1159, -2288
    0292 10F7
 49 0294 9F03                         .WORD 927, -1571
    0296 DDF9
 50 0298 3303                         .WORD 819, -1167
    029A 71FB
 51 029C F502                         .WORD 757, -903
    029E 79FC
 52 02A0 CE02                         .WORD 718, -715
    02A2 35FD
 53 02A4 B202                         .WORD 690, -572
    02A6 C4FD
 54 02A8 9D02                         .WORD 669, -457
    02AA 37FE
 55 02AC 8C02                         .WORD 652, -361
    02AE 97FE
 56 02B0 7F02                         .WORD 639, -279
    02B2 E9FE
 57 02B4 7302                         .WORD 627, -205
```

```
   0286 33FF
58 02B8 6902                           .WORD 617, -139
   02BA 75FF
59 02BC 6002                           .WORD 608, -77
   02BE B3FF
60 02C0 5802                           .WORD 600, -18
   02C2 EEFF
61 02C4 5102                           .WORD 593, 39
   02C6 2700
62 02C8 4A02                           .WORD 586, 95
   02CA 5F00
63 02CC 4302                           .WORD 579, 152
   02CE 9800
64 02D0 3C02                           .WORD 572, 210
   02D2 0200
65 02D4 3502                           .WORD 565, 271
   02D6 0F01
66 02D8 2E02                           .WORD 558, 336
   02DA 5001
67 02DC 2702                           .WORD 551, 408
   02DE 9801
68 02E0 2002                           .WORD 544, 488
   02E2 EB01
69 02E4 1802                           .WORD 536, 581
   02E6 4502
70 02E8 1002                           .WORD 528, 691
   02EA B302
71 02EC 0702                           .WORD 519, 827
   02EE 3B03
72 02F0 FE01                           .WORD 510, 1004
   02F2 EC03
73 02F4 F701                           .WORD 503, 1248
   02F6 E004
74 02F8 F701                           .WORD 503, 1615
   02FA 4F06
75 02FC 1202                           .WORD 530, 2241
   02FE C108
76                       ;
77                       ;
78                       ; TEST DATA FOR FFT ROUTINES.
79                       ; OBTAINED FROM :
80                       ;
81                       ;
82 0300 0004                           .WORD 1024, 0
   0302 0000
83 0304 9A03                           .WORD 922, 307
   0306 3301
84 0308 E102                           .WORD 737, 553
   030A 2902
85 030C F201                           .WORD 498, 719
   030E CF02
```

```
 86 0310 E800                               .WORD 232, 796
    0312 1C03
 87 0314 E2FF                               .WORD -30, 786
    0316 1203
 88 0318 F9FE                               .WORD -263, 699
    031A BB02
 89 031C 42FE                               .WORD -446, 550
    031E 2602
 90 0320 C9FD                               .WORD -567, 361
    0322 6901
 91 0324 96FD                               .WORD -618, 155
    0326 9B00
 92 0328 A5FD                               .WORD -603, -46
    032A D2FF
 93 032C EFFD                               .WORD -529, -222
    032E 22FF
 94 0330 67FE                               .WORD -409, -359
    0332 99FE
 95 0334 FBFE                               .WORD -261, -446
    0336 42FE
 96 0338 9BFF                               .WORD -101, -479
    033A 21FE
 97 033C 3500                               .WORD 53, -462
    033E 32FE
 98 0340 BA00                               .WORD 186, -400
    0342 70FE
 99 0344 1F01                               .WORD 287, -304
    0346 D0FE
100 0348 5E01                               .WORD 350, -187
    034A 45FF
101 034C 7301                               .WORD 371, -64
    034E C0FF
102 0350 6101                               .WORD 353, 54
    0352 3600
103 0354 2D01                               .WORD 301, 154
    0356 9A00
104 0358 E100                               .WORD 225, 229
    035A E500
105 035C 8600                               .WORD 134, 274
    035E 1201
106 0360 2600                               .WORD 38, 287
    0362 1F01
107 0364 CCFF                               .WORD -52, 269
    0366 0D01
108 0368 81FF                               .WORD -127, 227
    036A E300
109 036C 49FF                               .WORD -183, 166
    036E A600
110 0370 2AFF                               .WORD -214, 95
    0372 5F00
111 0374 23FF                               .WORD -221, 21
```



```
NATIONAL SEMICONDUCTOR CORPORATION        PAGE:    5
HPC CROSS ASSEMBLER,REV:C,30 JUL 86



    0376 1500
112 0378 33FF                       .WORD -205, -48
    037A D0FF
113 037C 55FF                       .WORD -171, -104
    037E 98FF
114                    ;
115                                 .END
                                        @
ERROR, OPERAND MUST BE SINGLE VALID SYMBOL NAME




NATIONAL SEMICONDUCTOR CORPORATION        PAGE:    6
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

SYMBOL TABLE


A      00C8 W       B      00CC W       K      00CA W       PC     00C6 W
SP     00C4 W       X      00CE W
```

```
NATIONAL SEMICONDUCTOR CORPORATION         PAGE:     7
HPC CROSS ASSEMBLER,REV:C,30 JUL 86

MACRO TABLE



   NO WARNING LINES

    1 ERROR LINES

  384  ROM BYTES USED

SOURCE CHECKSUM = 7A03
OBJECT CHECKSUM = 0705

INPUT   FILE C:TSTDAT.MAC
LISTING FILE C:TSTDAT.PRN
```

AN-497

Expanding the HPC Address Space

National Semiconductor
Application Note 497
Joe Cocovich

INTRODUCTION

The maximum address range of the HPC family of 16-bit High Performance microControllers is 64k bytes using the external address/data bus to interface with external memory. This application note describes a method to increase the amount of memory in a system to 544k bytes utilizing bank switching techniques. Block diagrams are presented to aid in circuit design. Software examples are given for memory and bank management.

HPC ADDRESSING

Program memory addressing is accomplished by the 16-bit Program Counter on a byte basis (instructions are always fetched a byte at a time). Memory can be addressed as words or bytes directly by instructions or indirectly through the B, X and SP registers. Words are always addressed on even-byte boundaries. The HPC uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

The external address/data bus of the HPC is 16 bits wide. This means the maximum address that the bus can hold is FFFF for a maximum address range of 64K bytes (65,536). Keep in mind, this uses the external address/data bus (A0:A15 for Address/Data and B10, 11, 12, 15) for Control.

BANK SWITCHING

If more than 64k of addressing is needed in the HPC system, the following method of increasing memory space can be used. Divide the total address range into two halves (32k bytes each). One half of this address range will be the MAIN memory address space. The MAIN memory address space will contain logical addresses (those addresses which the Program Counter can generate) in the range 8000 to FFFF and is accessed when A15 is a '1'. This includes the Interrupt vectors' and the Reset vector memory locations. The other half of the address range will be the BANK memory address space. The BANK memory address space will contain logical addresses in the range 0000 to 7FFF and is accessed when A15 is a '0'. This includes the on-chip I/O, registers, and RAM at locations 0000 to 01FF.

Now, four additional address lines are created using Port B pins (B8, B9, B13, B14). This prevents the use of the four timer synchronous outputs TS0–TS3 which are the alternate functions for these pins. The BANK memory is now addressed using A0:A14, B8, B9, B13, B14 and is accessed when A15 is a '0'. The BANK memory address space is now expanded to 512k bytes broken down into 16 individually selectable banks of 32k bytes each selected by these four bits of Port B.

A look at Table 1 and *Figure 1* quickly tells you that only one bank in the BANK memory space can share the logical address range 0000:7FFF at any one time. Therefore, programs running in the BANK memory address space can only directly access data and programs in the MAIN memory address space or in it's own bank (selected by B8, B9, B13, B14). On chip resources, which include RAM, I/O, and registers are mapped into logical addresses 0000 to 01FF. These logical addresses are in the BANK memory address space, but, since these addresses are considered to be always on-chip by the HPC, it never looks at the external address/data bus and will not read external memory in this range. Therefore, the first 256 bytes in each bank of memory in the BANK memory space will not be accessible by the HPC, but this address range (on chip resources) is directly accessible by any bank of memory in the BANK memory address space. This is why *Figure 1* shows a total available memory of 536.5k.

The interrupt vectors are mapped into logical addresses FFF0 to FFFF which are in the MAIN memory address space. Interrupts are handled properly if they occur while executing a program out of one of the banks of memory in BANK memory space, since the interrupt vector locations have A15 set to '1' which will allow access to the MAIN memory space. However, these interrupt vectors must either point to a routine in the MAIN memory address space which performs the interrupt service or point to code that selects the appropriate bank of memory in the BANK memory space and go there if the interrupt service routine is located there.

The stack must be located so that it can be directly accessible from anywhere in memory. It can be placed in the MAIN memory space or in the on-chip RAM. Programs and data storage that must be shared and directly accessed by all memory banks in the BANK memory space should also reside in the MAIN memory space.

HPC OPERATING MODES

The HPC must be configured to run in one of it's Expanded modes of operation by setting the EA bit in the PSW to be able to address the BANK memory range of 0000 to 7FFF. This memory expansion addressing scheme will work if the HPC is configured in either the Normal Expanded mode (EXM pin tied low) or ROMless Expanded mode (EXM pin tied high). The Normal mode differs from the ROMless mode only by the fact that the HPC will access the on-chip ROM for addresses in the range of E000 to FFFF (in the case of the HPC16083) and will access the external MAIN memory for addresses in the range of 8000 to DFFF.

The external data bus size is determined once, at reset, by sampling the state of $\overline{\text{HBE}}$ (B12). If $\overline{\text{HBE}}$ is high when sampled, the HPC enters 8-bit mode. In 8-bit mode, only pins A0–A7 are used to transfer data and pins A8–A15 continue to hold the most-significant eight bits of the address. So, only the lower eight bits of the address need to be latched externally (*Figure 2*). If HBE is low when sampled, the HPC enters 16-bit mode. In 16-bit mode, all 16 pins of Port A are used to transfer data as well as addresses. Two octal latches are then required externally to hold each address as it is issued by the HPC. The signal ALE from the HPC clocks the latches (*Figure 3*).

Keep in mind that if the external memory is configured as 8-bit memory, then the program stack must be in internal on-chip RAM because it has to be accessible as 16-bit words. If the external memory is configured as 16-bit memory then the stack can be in external RAM but must be in the MAIN memory address space to be directly accessible by all banks.

PROGRAMMING CONVENTIONS

A convention must be followed for maintaining linkages between the programs and data running in the MAIN memory space and the programs and data running in the BANK memory space. For the following discussion, the MAIN memory space will be referred to as just another bank of memory.

MAIN bank reserved portion

A portion of the MAIN memory bank should be reserved for Jump instructions to subroutines in the MAIN memory bank that need to be called by programs running in any selected bank in the BANK memory space. These Jump instructions serve as entry points for programs and subroutines. Typically, common functions that are required by programs running in several banks would be put in the MAIN memory bank. These could include: interrupt service routines, I/O drivers, and data handling and conversion routines. This portion also contains address pointers to tables of data in the MAIN memory bank that also are required by programs running in any selected bank in the BANK memory space. See Listing 1 for an example.

BANK memory reserved portion

A portion of each bank in the BANK memory space should be reserved for Jump instructions to subroutines in that bank that need to be called by programs running in the MAIN memory bank. These Jump instructions serve as entry points for programs and subroutines. For example, each bank in the BANK memory space could contain routines that perform unique but related functions. One bank could be reserved for math routines; another bank could perform message handling; and yet another could contain diagnostic routines. All of these functions could be scheduled and executed from some sort of Supervisor running in the MAIN memory bank performing the linkages to all these routines thru the entry points. This reserved portion of each bank also contains address pointers to tables of data in that bank that also are required by programs running in the MAIN memory bank. In the case of a bank running message handling routines, address pointers could be inserted to point to buffers that programs running in MAIN memory need to access. See Listing 2 for an example.

Linkage areas

These reserved portions of each memory bank (MAIN space or BANK space) must be fixed and known to each other memory bank that requires access to programs and data in that bank. Therefore, one other requirement in each bank is a set of labels that are assigned the values of the pointer locations to subroutines and tables in the bank of interest (see Listings 3 and 4).

One last requirement in the MAIN memory bank, if it is to perform bank to bank moves and for general housekeeping, is to reserve two byte locations to be used to keep track of the bank currently selected (high byte value on Port B) being used in the transfer of data (see Listing 5).

From the MAIN memory bank, the user can access all memory in the system. He can call subroutines in any bank in the BANK memory space and read/write data to the entire memory. From any bank in the BANK memory space, the user can call subroutines in the MAIN memory bank and read/write data to the MAIN memory bank in addition to his own local bank.

The basic procedure used to call a program in the BANK memory space from the MAIN memory bank is merely to set the proper value on the Port B select lines and execute a Jump to SubRoutine through a pointer in the selected bank:

Interrupts

Regardless of where the interrupt service routine actually resides, an image of the bank selected must be retained by the service routine to allow it to return to the appropriate bank when complete. If the interrupt service routine is in the MAIN memory bank, the linkage is handled in the normal fashion where the interrupt vector points to the service routine. The interrupt service can reside in the BANK memory space and takes a little extra overhead for the linkage.

To call a program in the MAIN memory bank from the BANK memory space, merely execute a Jump to SubRoutine through a pointer in the MAIN memory bank:

```
    JSRL CMPBLNK       ;see Listing 1 and 4
```

EXAMPLE SOFTWARE

Now that a convention has been established for communicating between the MAIN memory space and the BANK memory space, let's take a look at some sample code that can be used to move data between these memory spaces. In order to make the selection of bank memory efficient, it is important to keep in mind that the four bits of the high byte of Port B that are used to select a bank of memory in the BANK memory space can be written to directly since the other 4 bits of this byte of Port B are used for memory control outputs (the external control bus) and are not affected by a write to the high byte of Port B.

Bank to Bank data transfer by MAIN

Listing 6 shows the setup required to initialize the linkage area in order to perform a transfer of data from one bank to another bank in the BANK memory space by a program running in the MAIN memory space. This involves setting up the RAM locations that are used to 'select' the source bank and the destination bank, select the source bank to determine the starting address of the area to move, select the destination bank to determine the starting address of the area to move data into, then finally calling the subroutine in MAIN memory that performs the move. After the setup portion, the subroutine that performs the transfer is presented. This code assumes that the external memory is configured in 16-bit mode.

Bank to MAIN data transfer by Bank

Listing 7 presents a similar example for moving blocks of data from a bank in BANK memory to MAIN memory by a program running in that bank. This code also assumes that the external memory is configured in 16-bit mode.

External 8-bit mode

If the external memory is configured in 8-bit mode, the setup portion changes because the initialization of the RAM address pointers SSTART, DSTART and DEND requires building word address pointers from word pointers in the external reserved areas of each bank. In 8-bit mode, this requires two 8-bit transfers compared to one 16-bit transfer in 16-bit mode (see Listing 8). Once these address pointers have been built, however, the subroutine that actually performs the move does not have to change because 1) word transfers are allowed between On-chip RAM and registers regardless of the mode and 2) the subroutine performs byte moves. To improve speed in the 16-bit mode, this subroutine can be modified to perform 16-bit moves. However, keep in mind that this will impose the restriction on the address pointers in the linkage areas of requiring that addresses be on word boundaries. Listing 9 presents a similar example for moving blocks of data from a bank in BANK memory to MAIN memory by a program running in that bank.

PROGRAM DEVELOPMENT

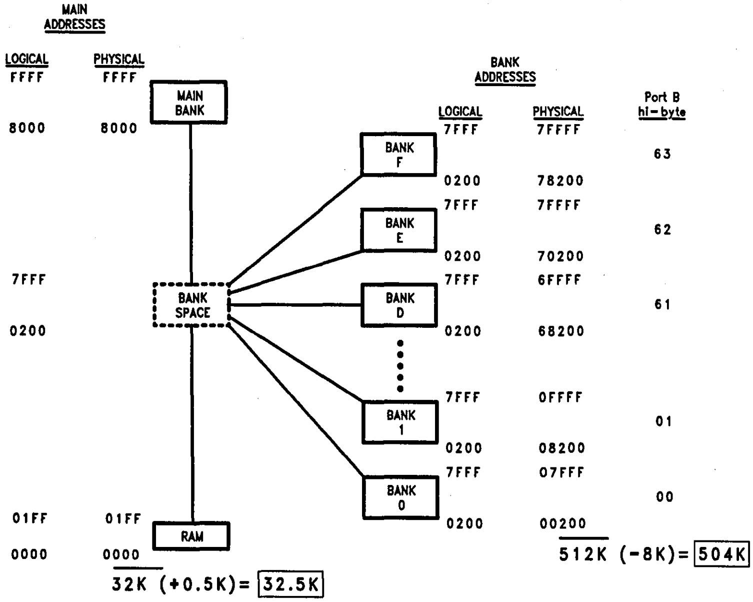
The MOLE monitor software can support the development of HPC programs in multiple banks of memory. It provides the means of qualifying a trigger condition, as set in Trace or Breakpoint functions, with the memory bank number. The BANK command will allow a trigger only when executing in the memory bank of interest. The MOLE supports a total of 16 memory banks which are normally selected by 4 bits of Port B as described earlier. See the HPC Personality Board User's Manual for further detail on this command.

CONCLUSION

What has been presented is a method to expand the memory space of the HPC to 544k. Although this method utilized four bits of Port B to accomplish the extra addressing, theoretically, the remaining 8 bits could have been used if not required for other purposes. This could mean a maximum addressability for the HPC of greater than 128 Megabytes. However, the MOLE will only support the fixed definition of four extra address lines. Clever utilization of existing resources can enable you to get the most out of hardware and software limited only by one's imagination.

TABLE I. Logical Addresses vs Physical Memory Locations

Logical Address	Bank #	Hi Byte Port B	Physical Address	
0000:7FFF	0	00	00000:07FFF	(BANK)
0000:7FFF	1	01	08000:0FFFF	
0000:7FFF	2	02	10000:17FFF	
0000:7FFF	3	03	18000:1FFFF	
0000:7FFF	4	20	20000:27FFF	
0000:7FFF	5	21	28000:2FFFF	
0000:7FFF	6	22	30000:37FFF	
0000:7FFF	7	23	38000:3FFFF	
0000:7FFF	8	40	40000:47FFF	
0000:7FFF	9	41	48000:4FFFF	
0000:7FFF	A	42	50000:57FFF	
0000:7FFF	B	43	58000:5FFFF	
0000:7FFF	C	60	60000:67FFF	
0000:7FFF	D	61	68000:6FFFF	
0000:7FFF	E	62	70000:77FFF	
0000:7FFF	F	63	78000:7FFFF	
8000:FFFF	—	—	08000:0FFFF	(MAIN)



TL/DD/9342-1

FIGURE 1. How BANK Memory is Mapped into the HPC Address Space

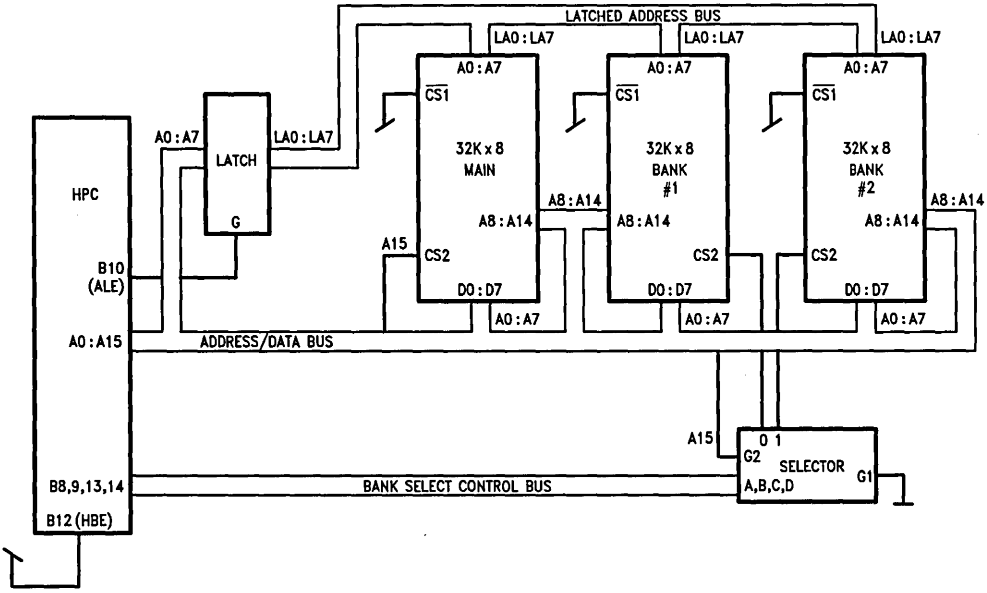
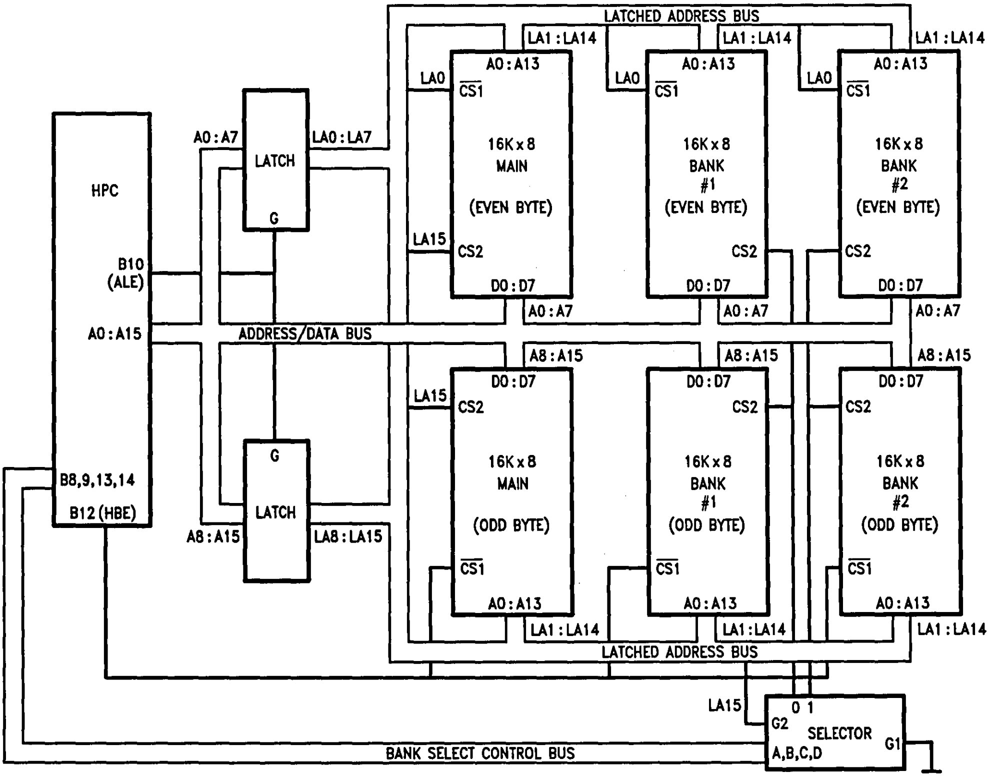


FIGURE 2. HPC in 8-Bit Mode

TL/DD/9342-2



TL/DD/9342-3

FIGURE 3. HPC in 16-Bit Mode

```
            .=08000    ;set PC counter to 8000
;This code resides in the MAIN memory bank
;
;     The following address pointers are inserted to allow
;     programs running in BANK memory to find these
;     locations. They represent the starting and ending
;     location for code in MAIN memory.
;
      .WORD INIT  ;addr pointer to first location in bank
      .WORD PROGEND     ;addr pointer to last location in bank
;
;     The following Jump instructions are inserted to allow
;     programs running in BANK memory to call these
;     subroutines. They represent subroutines that compare
;     blocks of memory in MAIN memory space with blocks of
;     memory in BANK memory space or compare blocks of memory
;     in BANK memory for zeros.
;
      JMPL CMPM         ;entry for compare blocks (MAIN-BANK)
      JMPL CMPBFB       ;entry for compare BANK cleared
```

LISTING 1. MAIN Bank Reserved Portion

```
            .=0200     ;set PC counter to 200
;This code resides in any bank in BANK memory
;
;     The following address pointers are inserted to allow
;     programs running in MAIN memory to find these
;     locations. They represent the ending location for code
;     in this bank of BANK memory.
;
      .WORD PROGEND     ;addr pointer to last loc in this bank
;
;     The following Jump instructions are inserted to allow
;     programs running in MAIN memory to call these
;     subroutines. They represent subroutines that compare
;     blocks of memory in MAIN memory space with blocks of
;     memory in this bank, diagnostic routines, and interrupt service routine.
;
      JMPL CMPMB  ;entry for comp blocks (MAIN-this bank)
      JMPL BTEST  ;entry for this bank's diag routines
      JMPL BINTS  ;entry for this bank's interrupt service routine
```

LISTING 2. Typical Bank Reserved Portion

```
;This code resides in the MAIN memory bank
;
;     linkages to Bank 0
;
B0START = 0200          ;addr of pointer to first avail loc
;
CMPMB0 = 0202           ;addr of JMPL to routine that compares
;                            move results
B0TEST = 0205           ;addr of JMPL to test routines
;
;     linkages to Bank 1
;
B1START = 0200          ;addr of pointer to first avail loc
;
CMPMB1 = 0202           ;addr of JMPL to routine that compares
;                            move results
B1TEST = 0205           ;addr of JMPL to test routines
;
;     linkages to Bank 2
;
B2START = 0200          ;addr of pointer to first avail loc
;
CMPMB2 = 0202           ;addr of JMPL to routine that compares
;                            move results
       = 0205           ;addr of JMPL to test routines
;
B2INTS = 0208           ;addr of JMPL to interrupt service routine
```

LISTING 3. MAIN Memory Bank Linkage Area

```
;This code resides in any bank in BANK memory
;
;     linkages to MAIN memory
;
MSTART = 08000          ;addr of pointer to first avail loc
MEND = 08002            ;addr of pointer to last avail loc
;
CMPM = 08004            ;addr of JMPL to routine that compares
;                            move results
CMPBLNK = 08007         ;addr of JMPL to routine that compares
;                            if a block in selected BANK is zero
```

LISTING 4. Typical Bank Linkage Area

```
;This code resides in the MAIN memory bank
;
; The following locations are used for bank to bank moves
; and compares
      BANKS = 01C0            ;source bank byte value
      BANKD = 01C1            ;destination bank byte value
;
      BANK0 = 0         ;Port B high byte value to select bank 0
      BANK1 = 1         ;                                      1
      BANK2 = 2         ;                                      2
      BANK3 = 3         ;                                      3
      BANK4 = 020       ;                                      4
      BANK5 = 021       ;                                      5
      BANK6 = 022       ;                                      6
      BANK7 = 023       ;                                      7
      BANK8 = 040       ;                                      8
      BANK9 = 041       ;                                      9
      BANKA = 042       ;                                      10
      BANKB = 043       ;                                      11
      BANKC = 060       ;                                      12
      BANKD = 061       ;                                      13
      BANKE = 062       ;                                      14
      BANKF = 063       ;                                      15
;
;     Main Memory Bank is logical and physical address range
;     8000:FFFF. Switched Memory Banks are logical addresses
;     in the range 0000:7FFF combined with the
;     Port B(14,13,9,8) bits to create physical addresses in
;     the range 00000:7FFFF
;
```

LISTING 5. BANK Memory Management

```
         LD M(0E3),BANK1;set bank select lines to select bank 1
         JSRL B1TEST    ;see Listing 2 and 3
                                          •
                                          •
                                          •
INT35:
LD       BANKS,M(0E3)   ;save bank interrupted from
LD       M(0E3),BANK2   ;set bank select lines to select bank 2
JSRL     B2INTS         ;see listing 2 and 3
                                          •
                                          •
                                          •
LD       M(0E3),BANKS   ;restore bank interrupted from
RETI
                                          •
                                          •
                                          •
.IPT     2,INT35        ;set interrupt vector
```

```
;This code resides in the MAIN memory bank
;
      LD M(BANKS),BANK0       ;prepare to move data from Bank 0
      LD M(BANKD),BANK1       ;to Bank 1
      LD M(0E3),BANK0          ;select Bank 0
      LD W(SSTART),W(B0START) ;set starting address in source bank
      LD M(0E3),BANK1          ;select Bank 1
      LD W(DSTART),W(B1START) ;set starting address in destination bank
      LD W(DEND),W(B1START)   ;set ending address in destination bank
      ADD W(DEND),1023         ;to 1K greater than starting address
      JSRL MOVBB                ;do it
            •
            •
            •
            •
            •
;
;  This subroutine moves data from bank memory to bank memory
;  where the source bank is defined by the contents of the byte
;  at RAM location BANKS and the destination bank is defined by
;  the contents of the byte at RAM location BANKD. In addition,
;  the following locations must be set up before calling:
;
;  SSTART  →  RAM location containing source bank start address
;  DSTART  →  RAM location containing destination bank start address
;  DEND  →  RAM location containing destination bank end address
;
MOVBB:
      LD B,W(DSTART)          ;B  ←  starting address (destination)
      LD K,W(DEND)            ;K  ←  ending address (destination)
      LD X,W(SSTART)          ;X  ←  starting address (source)
LOOPBB:
      LD M(0E3),M(BANKS)      ;select source BANK
      LD A,M(X+)                ;byte at source into A
                                 ;increment source pointer
      LD M(0E3),M(BANKD)      ;select destination BANK
      XS A,M(B+)                ;A into byte at destination, bump pntr
      JP LOOPBB                 ;back for more if B less than K
      RET
```

LISTING 6. Move Data by MAIN from BANK to BANK (16-Bit Mode)

```
;This code resides in any bank in BANK memory
;
      LD W(SSTART),TABLE1      ;starting address of table in this memory
      LD W(DSTART),W(MSTART)  ;starting address in main memory
      LD W(DDEND),TABLE1+1023 ;ending address in main memory
      JSRL MOVE               ;do it
              •
              •
              •
              •
              •
;
;  This subroutine moves data from this bank to main memory
;
;  SSTART  →  RAM location containing source memory start address
;  DSTART  →  RAM location containing destination memory start addr
;  DEND →  RAM location containing destination memory end address
;
MOVE:
      LD B,W(DSTART)          ;B  ←  starting address (destination)
      LD K,W(DEND)            ;K  ←  ending address (destination)
      LD X,W(SSTART)          ;X  ←  starting address (source)
LOOPBM:
      LD A,M(X+)              ;byte at source into A
                               ;increment source pointer
      XS A,M(B+)              ;A into byte at destination, bump pntr
      JP LOOPBM               ;back for more if B less than K
      RET
```

LISTING 7. Move Data by BANK from BANK to MAIN (16-Bit Mode)

```
;This code resides in the MAIN memory bank
;
      LD M(BANKS),BANK0        ;prepare to move data from Bank 0
      LD M(BANKD),BANK1        ;to Bank 1
      LD M(0E3),BANK0           ;select Bank 0
      LD M(SSTART),M(B0START) ;set starting address in source bank
      LD M(SSTART+1),M(B0START+1)
      LD M(0E3),BANK1           ;select Bank 1
      LD M(DSTART),M(B1START)  ;set starting address in destination bank
      LD M(DSTART+1),M(B1START+1)
      LD M(DEND),M(B1START)   ;set ending address in destination bank
      LD M(DEND+1),M(B1START+1)
      ADD M(DEND),L(1023)      ;to 1K greater than starting address
      ADC M(DEND+1),H(1023)
      JSRL MOVBB                 ;do it
                  •
                  •
                  •
                  •
                  •
;
;  This subroutine moves data from bank memory to bank memory
;  where the source bank is defined by the contents of the byte
;  at RAM location BANKS and the destination bank is defined by
;  the contents of the byte at RAM location BANKD. In addition,
;  the following locations must be set up before calling:
;
;  SSTART  →  RAM location containing source bank start address
;  DSTART  →  RAM location containing destination bank start address
;  DEND  →  RAM location containing destination bank end address
;
MOVBB:
      LD B,W(DSTART)           ;B ← starting address (destination)
      LD K,W(DEND)             ;K ← ending address (destination)
      LD X,W(SSTART)           ;X ← starting address (source)
LOOPBB:
      LD M(0E3),M(BANKS)       ;select source BANK
      LD A,M(X+)                 ;byte at source into A
                                  ;increment source pointer
      LD M(0E3),M(BANKD)       ;select destination BANK
      XS A,M(B+)                 ;A into byte at destination, bump pntr
      JP LOOPBB                  ;back for more if B less than K
      RET
```

LISTING 8. Move Data by MAIN from BANK to BANK (8-Bit Mode)

```
;This code resides in any bank in BANK memory
;
      LD M(SSTART),L(TABLE1)  ;starting address of table in this memory
      LD M(SSTART+1),H(TABLE1)
      LD M(DSTART),M(MSTART)  ;starting address in main memory
      LD M(DSTART+1),M(MSTART+1)
      LD M(DEND),M(MSTART)    ;set ending address in main memory
      LD M(DEND+1),M(MSTART+1)
      ADD M(DEND),L(1023)      ;to 1K greater than starting address
      ADC M(DEND+1),H(1023)
      JSRL MOVE                  ;do it
            •
            •
            •
            •
            •
;
;  This subroutine moves data from this bank to main memory
;
;  SSTART  →  RAM location containing source memory start address
;  DSTART  →  RAM location containing destination memory start addr
;  DDEND  →  RAM location containing destination memory end address
;
MOVE:
      LD B,W(DSTART)          ;B  ←  starting address (destination)
      LD K,W(DEND)            ;K  ←  ending address (destination)
      LD X,W(SSTART)          ;X  ←  starting address (source)
LOOPBM:
      LD A,M(X+)              ;byte at source into A
                               ;increment source pointer
      XS A,M(B+)              ;A into byte at destination, bump pntr
      JP LOOPBM               ;back for more if B less than K
      RET
```

LISTING 9. Move Data by BANK from BANK to MAIN (8-Bit Mode)

The code listed in the App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

Assembly Language Programming for the HPCTM

National Semiconductor
Application Note 510
Steve McRobert

HOW TO WRITE SHORT, EFFICIENT, BUT UNDERSTANDABLE ASSEMBLER PROGRAMS

INTRODUCTION

One of the design objectives of the HPC family was that it should be very easy to use. With this in mind the instruction set has been designed so that it obeys a very simple set of rules. Once these rules have been learned, the programmer can write code with very little reference to instruction manuals.

The HPC is fully memory mapped. Every piece of hardware attached to an HPC core appears as a byte or a word in a linear 64K byte address space. Any data movement or arithmetic instruction can operate on any memory location and everything in the HPC has a memory location, including the accumulator. All of the I/O ports, the peripheral control registers, RAM and ROM are treated in exactly the same fashion as far as the assembly language programmer is concerned.

The HPC assembly language syntax can be explained by describing the instruction codes and the addressing modes. The instruction code tells the processor what operation it is performing, such as an add, a subtract, a multiply, a divide or a data movement instruction. The addressing mode is the way that the programmer specifies the value or values to be operated on to the microprocessor itself.

ADDRESSING MODES

Operations can be performed on any memory location. One can, for example, increment or decrement any byte or word of any memory location in the HPC. Increment and decrement are examples of single address instructions. These are instructions which have only one operand. Other examples are the bit set, bit test and bit clear instructions. These five instructions are good examples of the basic thinking behind the HPC instruction set. All of these instructions use the same four addressing modes.

Direct

The simplest addressing mode to understand is that known as direct. In this mode the address of the variable to be operated on is included as part of the sequence of bytes that comprises the entire instruction. For example, in order to perform a decrement on memory location 0F0 this value is included in the string of bytes that forms the instruction.

```
Examples:
  DECSZ   0F0.B
  INC     0F0.W
```

The increment instruction, like most other instructions with HPC, can operate on either a byte or a word. A byte access is specified by putting a B after the address of the variable, a word access by writing W.

Register Indirect

This addressing mode usually generates less bytes of code than any other. HPC has two 16-bit registers, B and X, which can be used as general purpose memory locations but also have a specific function as pointers to memory. These instructions take up very little ROM space because the address of the variable to be operated on is contained in the pointer register and the pointer register to be used is specified as part of the instruction. An instruction such as increment, using register indirect, can thus be only 1 byte long as it does not need to be followed by a byte specifying the address of the variable.

```
Examples:
  INC     [B].B ;byte increment, B pointer
  INC     [X].W ;word increment, X pointer
```

Indirect

B and X provide two 16-bit pointers to memory. Programmers will often wish to have more than two pointers in use at any one time. HPC therefore provides indirect addressing mode. In this mode a 16-bit pointer to the location to be accessed is stored in the basepage of the HPC. The instruction, therefore, is followed by a single byte which specifies the address of this 16-bit pointer. The bottom 192 bytes of RAM are on chip with the HPC and are in the so-called base page. The base page is normally used for storing frequently accessed variables as only a single byte of address is required to access a base page variable. When using indirect addressing mode, the 16-bit pointer value must always be in the base page.

```
Examples:
  DECSZ  [0].W    ;decrement a word
  INC    [0FE].B ;increment a byte
```

The base page is in the region of 0 to 0FF bytes. This area also contains the most frequently used registers such as the accumulator. The programmer can thus use indirect addressing mode with registers such as the accumulator acting as the pointer. This is an example of the simplicity of the HPC instruction set. Any operation can be performed on any HPC register simply by invoking its address in the HPC 64 kbyte addressing space.

Indexed

The last of the four basic addressing modes is indexed mode. Indexed is very similar to indirect except that an 8- or 16-bit immediate value precedes the address of the 16-bit pointer and is added to it to generate the address of the variable to be accessed. This allows a table of values to be located anywhere in memory and the pointer register need only be incremented or decremented to move through the table of values.

```
Examples:
  INC    0FF00 [4].W  ;increment a word
  DECSZ  02 [2].B     ;decrement a byte
```

Bit Operations

The bit operations of the HPC allow any bit in the memory of the HPC to be accessed. The addressing modes for these three operations, SBIT, RBIT and IFBIT, always refer to the memory location as a byte. The individual bit of the byte to be tested, using the four addressing modes already described, is actually coded into the opcode itself. This could be described as an implied addressing mode but this definition is not normally used in HPC. The way this works can be seen from the opcode map in the programmers guide of the HPC, where it can be seen that there are in fact eight opcodes shown for each of the three different bit instructions.

Example:

```
SBIT 5, 2.B  ;set bit 5 of byte
             ;at address 2.
```

Double Register Indirect

A rule of thumb when trying to decide which addressing mode one can use with which opcode in HPC is that you can use any combination of addressing mode and opcode that is sensible. An example of this is a special addressing mode which works only for the bit instructions. This addressing mode is known as double register indirect and uses a combination of the B and X registers to index into any bit of a 64k bit string, the lower boundary of which can be located anywhere in memory.

When using this addressing mode the B register points to the lowest byte of this 8k byte string, while the most significant 13 bits of the X register point at the individual byte in the string that is being accessed. The three least significant bits of the X register point at the bit of the byte that the instruction is pointing at. By using this addressing mode, words of any length can be scanned for whether individual bits are set or cleared. This addressing mode, while unusual, fits into the scheme of things as it clearly is only of any relevance to the individual bit instructions.

Examples:

```
SBIT X,   [B].B; Set bit
IFBIT X,  [B] B; test bit
```

Note that the bit instructions only operate on bytes, to allow operations on words would require twice as many opcodes for no gain.

Two Address Instructions

The five instructions described so far have only one operand. There are many more instructions in the HPC instruction set which have two operands, such as arithmetic instructions, the comparison instructions and data movement instructions. The HPC instruction set allows any of these instructions to use any of the four addressing modes already described. An instruction such as multiply, for example, when written in the HPC assembler syntax as shown below shows the opcode followed by the destination operand, which is then followed by the source operand. The result of the operation in all cases except the comparison instructions winds up in the destination operand. The comparison instructions, IFEQ and IFGT do not affect the values of any memory location but, like all other two operand instructions, can operate on any two words or bytes in the HPC addressing space.

Examples:

```
MUL A, [B].B
MUL 0.W,2.W
```

The destination operand in HPC may be either the accumulator or a byte or word of memory accessed using the direct addressing mode. If the destination operand is the accumulator, the source operand may be addressed using direct, register indirect, indirect or indexed addressing modes as well as the familiar immediate addressing mode. The programmer can thus load the accumulator with an 8- or 16-bit immediate value which follows the opcode, multiply the accumulator with that value, divide the accumulator by that value or compare the accumulator by that value. Using the accumulator as the destination operand gives maximum flexibility in the choice of addressing mode for the source operand and also tends to produce a shorter instruction in terms of its length in bytes as the opcode does not have to include the address of the destination operand.

Examples:

```
LD A, #37              ;load A With
                       ;immediate value.
add OFE.W,#  OF000 ;Add immediate to
                       ;memory.
```

Instruction Lengths

Tables are provided in the HPC users manual to allow the user to estimate the number of bytes an instruction will use and the time this instruction will take to execute. To use these tables the programmer must be aware of the name of the addressing mode he is using. This is perfectly clear for the single address instructions described at the beginning of this note but perhaps needs some explanation for two operand instructions.

For two operand instructions with the accumulator as the destination, the addressing mode is named after that used for the source operand. For example, load accumulator using a value pointed at by indirect addressing mode is referred to simply as indirect addressing mode.

Operations on Direct Memory

There are two addressing modes which allow operations to be performed directly on memory locations. If the destination operand is directly addressed memory, then the source operand may be directly addressed memory or an immediate value. These two are the only combinations of addressing modes that can be used where the destination operand is a memory location.

Examples:

```
DIV   O10.W, OF000.W
direct-direct mode
DIV   OFO.B,#10
immediate direct mode.
```

Special Symbols

Some special symbols have been allocated in the HPC cross assembler. These are A, B, K, X, PC and SP. The programmer can also define his own symbols using the equals directive of the assembler. The way that the symbols described above would be defined using the equals directive are shown below by way of example.

Example:

```
 A = OC8.W
 B = OCC.W
 X = OCE.W
 K = OCA.W
PC = OC6.W
SP = OC4.W
```

Note that these symbols cannot be redefined so the above set of definitions should never be included in a user program.

IMPLIED ADDRESSING MODES

Some of the HPC's opcodes have been shortened by using implied addressing mode. A few examples have already been shown. This section describes some more special cases. It could be said that accumulator as destination is an example of an implied addressing mode, where the address of the destination is coded into the instruction. There are some special purpose instructions which use implied addressing mode for instructions which are used very frequently. In most cases these instructions look exactly the same to the programmer as instructions using the addressing modes described earlier. For example there is a special opcode for load B with an immediate value. The programmer could do this using the immediate direct addressing mode but a special opcode has been provided to make this instruction shorter.

Load B and K is a special immediate load which loads both the B and K registers in one operation.

Carry Flag

The carry flag may be accessed using the standard bit test instructions because it can be read in the processor status word, but as carry must so often be set and tested, special instructions to do this have been included which do not require the address of the carry flag.

Multiply and Divide

Finally, the divide double and multiply instructions both have to manipulate 32-bit values. These therefore have to store an operand in two concatenated registers. The HPC instruction set cannot specify two registers with one address. Therefore these instructions default to using the X register as the high word of their 32-bit value.

The source and destination of a multiply instruction are specified as normal except that the 32-bit answer is stored in the destination operand with the 16 high bits of the answer stored in the X register. The divide double instruction basically performs the inverse of multiply, taking the 32-bit value formed by X concatenated with the destination value and dividing it by the source value. Divide double, like divide, yields a 16-bit result and a 16-bit remainder. For both divide double and divide the remainder is stored in the X register. In both cases the K register is used for intermediate value storage and is cleared as a result of this operation.

As the result of divide double can only be a 16-bit value, a full 32-bit divide is performed by following a 16-bit divide with a 32-bit divide as shown below. The example below shows how the divide instructions work together and also highlights the combinations of addressing modes that can and cannot be used with HPC.

```
                LD          B,#11
                DIV         HIGH.W,#10
LOOP:           DIVD        LOW.W,#10
                LD A, X
                ST A, [B].B
                DECSZ B
                JP LOOP
```

This example shows the conversion of a 32-bit binary value in words low and high into a 10-digit BCD number in the 10 bytes starting from 1. The conversion is performed one digit at a time and the B register is used to point at the byte's location where the digit is to be stored. The first instruction of the programme therefore is to initialize the B register. The divide instruction divides word high by 10 using immediate direct addressing mode and stores the answer back in word high. The remainder is stored in the X register. The divide double instruction then divides X concatenated with word low by 10. Because X contains a remainder, the result of this division will always be a 16-bit value and can thus be stored in word low. The remainder is stored in X and is in fact the modulus and is thus the BCD digit that we have derived on this pass through the numbers.

We now wish to store the remainder into one of our BCD digit locations using register indirect mode. We need to load the value into the accumulator from X. The X register is nothing special in this application, so load A with word X is in fact an example of direct addressing mode.

Now that our BCD value is in the accumulator, we can store this in the byte location using B register indirect addressing mode.

The next instruction is decrement skip on zero. This uses direct addressing mode to decrement the B register. This instruction is an example of many in HPC which perform more than one function. As well as decrementing the memory location specified, this instruction also compares it with zero after the decrement has been performed. If the result is zero, the instruction following the decrement skip on zero instruction is skipped. That is to say it is ignored and control passes to the instruction following it. In this example the final instruction of the routine is a single byte jump back to the divide instruction. The overall loop is executed ten times in order to perform the conversion. On the final pass through the loop, B becomes zero and execution of this algorithm is terminated.

Auto Increment/Decrement Instructions

This multi-function instruction capability is best illustrated by the four special addressing modes register increment or decrement with or without conditional skip, which work only with the data movement instructions load and exchange. The load instruction in general uses any of the five two-address modes or the two combination modes to transfer data from one location to another.

The exchange instruction is similar except that the destination must always be the accumulator. Exchange not only takes the source and puts the value into the destination but also takes the value from destination and puts it into source. Clearly there is no immediate addressing mode for exchange as a destination cannot be stored into an immediate value.

When load and exchange are used with the X register as a pointer and register indirect mode, a suffix + or − can be added after the X. In this case, once the data movement operation has been performed, the X register is incremented or decremented by one or two according to whether there has been a byte or a word access respectively. A further refinement on this is provided by the load and exchange with conditional skip instructions, LDS and XS respectively. These only work with the B register as the pointer and perform two more operations rather similar to the decrement skip on zero instruction. Once the increment or decrement has been performed, the B register is compared with the K register, otherwise known as the limit register. If an increment has been performed and B is greater than K, the instruction following the movement instruction will be skipped. If a decrement is performed, the instruction is skipped if B is less than K.

An example of how these specialized instructions are used is given by the block move routine shown below;

```
         LD  X,#START
         LD  BK,#BEGIN,#END
LOOP:    LD  A, [X+].W
         XS  A, [B+].W
         JP  LOOP
```

This routine moves a block of data from one location to another. The X register is initialized first and is used as a pointer to the first value to be moved in the source block. The B and K registers point to the first and last values respectively in the destination block. The loop itself consists of only three bytes. The first instruction loads the accumulator with the word pointed to by the X register and increments X by two. A second instruction exchanges the accumulator with the word pointed to by the B register, increments the B register by two and compares it with K. If B is greater than K, the jump instruction is skipped and this loop is terminated.

The example shows how HPC code can perform a great deal with very few instructions and use up very few bytes of code while doing so.

These auto increment/decrement instructions are the only examples where an addressing mode cannot be used for any instruction where it might make sense. It is however fairly easy to remember which addressing modes these can be used with. Auto increment/decrement can be used with the load and exchange instructions for the X register. Auto increment or decrement with conditional skip can be used with load and exchange instructions using the B register as a pointer. No other combinations are allowed.

We have not provided specific string move or search instructions but the auto increment/decrement operations provide building blocks allowing the programmer to assemble his own stock. In the block move instruction shown above, the value being moved is in the accumulator in between the load and exchange instructions. The programmer can then compare this value with anything he wishes, fill BCD to ASCII, pack BCD, unpack BCD or perform any operation he likes on a string of data.

HPC ASSEMBLY CODE

The addressing modes usable for each opcode are described in a shorthand form.

```
Example:
                ADD  MA < MA + MemI
```

In the above syntax MA means directly addressed memory or the accumulator and MemI means memory addressed using any of the four basic single-address addressing modes or an immediate value. This would be better written as shown below:

```
          A < A + MemI
or        M < M + M
or        M < M + I
```

Expanding the syntax highlights that the flexible addressing modes such as register indirect may only be used if the destination is the accumulator. It also shows that if the destination is direct memory the source may only be an immediate value or another direct memory location.

When writing assembly code the programmer writes the same mnemonic whether a memory location is a piece of RAM or ROM or an I/O port or the accumulator. In general any source or destination variable may be a byte or a word and combinations are allowed. Care must be taken when storing word into a byte location that the programmer really wishes to truncate that value to byte and throw away the upper 8 bits of the value. When loading a byte into a word location the upper 8 bits of the word location will be filled with zeros. If memory external to the HPC is used, this may be 8 or 16 bits wide. The programmer must be aware of this when writing his assembly language as HPC cannot cope with the programmer requesting a 16-bit access to 8-bit wide external memory. The HPC will not convert this to two sequential 8-bit accesses.

The only exception to this rule is that a pointer word in indirect or indexed addressing modes must always be in the base page. This is because only one byte has been allowed in the overall length of the instruction for the address of the pointer.

For all other addressing modes there is no difference in the assembly language the programmer writes between accessing a variable that is in the base page and a variable that is above address 0FF.

The programmer should be aware however that variables in the base page consume less bytes per access and the instruction will execute more quickly than non-base page variables. When studying the data sheet to see how long an instruction is, the programmer will see that the table result is different according to whether variables are base page or not. The programmer should therefore allocate base page to variables which are used most often.

EXECUTION SPEED

There are 64 bytes of RAM above the base page. These, like the base page RAM, require zero wait states to access even when the processor is running at full speed. They do however require 2 bytes of code for their addresses. These

64 bytes may best be made use of by using them as the stack area as the 16-bit stack pointer contains the full address and therefore there is no penalty in instruction length in putting the stack in this non-base page on-chip RAM.

Note that there is no difference in execution time between byte and word accesses, that is to say accesses to byte or word variables. When studying the data sheet, differences in program length and therefore in execution time will be observed according to whether the address of a directly addressed variable is a byte or a word. It is important to understand the difference between the width of the variable and the width of the address that is used to access that variable.

The cycles per instruction table is not always clear about the number of wait states applied to different variables. The HPC includes a wait state register which sets the number of wait states to be used when accessing external memory, the internal ROM, or internal registers associated with ports A and B. Wait states may be applied to these on-chip registers to allow compatibility with development tools such as the MOLE™ and HPC Designer Kit board, as when these tools are run on high clock speeds wait states must be applied for accesses to the port recreation logic. The HPC needs wait states for accessing slow external memory and when running at high clock rates.

These wait states may be applied in order that the MOLE can provide a perfect emulation of a single-chip HPC. In the MOLE the HPC is running with external memory and thus the A port and some of the B port are used for address/data and control lines respectively. The A port and part of the B port must therefore be recreated external to the HPC. In the case of the MOLE this is done using a large array of PAL®s. Because they are external to the HPC, one wait state must be applied when accessing these externally recreated ports at high clock speeds. If wait states could not be applied to these ports in a masked ROM HPC, the MOLE would not be able to provide full speed emulation. This is just one example of how the design of the HPC has been influenced by the need to emulate it 100% exactly at full speed. Apart from this no wait states are applied to any access to address locations below 200 HEX, regardless of the addressing mode used.

The HPC data sheet does not make it clear how many wait states are applied when register indirect addressing mode is used. It implies that wait states are always applied when register indirect or similar addressing modes are used, but this is not the case.

The best way to time a piece of code is to write the code and then run it through the cross assembler to generate a source plus object listing. The number of bytes generated by each instruction can then be easily read and only the cycles and accesses table need be looked up in order to calculate how long each instruction takes to execute.

Note that accesses to internal ROM are subject to at least one wait state for exactly the same reason as accesses to the A or B ports.

SUMMARY

The HPC is fully memory mapped. The I/O Ports, Peripheral Control Registers, RAM and ROM are treated exactly the same. This makes the HPC easy to program. The HPC instruction set has relatively few opcodes but allows any of these opcodes to be used with any addressing mode so as to provide an Instruction Set with great power and flexibility.

Once the contents of this note have been understood, HPC code can be written without referring to any document more lengthy than the HPC Instruction Set description in the data sheet.

A Software Driver for the HPC Universal Peripheral Interface Port

National Semiconductor
Application Note 550
Brian Marley

ABSTRACT

This application note covers the use of the National Semiconductor HPC46083 High-Performance microController as an intelligent Peripheral Interface and Interrupt controller for another "Host" CPU, using its 8-bit or 16-bit parallel UPI (Universal Peripheral Interface) Port. Included in the discussion is the source text of an HPC driver program, which can be tailored as an "executive" for a wide variety of HPC tasks. A simple application is built from this software, which interfaces a National NS32CG16 CPU to a typical front panel (LED indicators, LCD alphanumeric display, pushbuttons and beeper).

1.0 INTRODUCTION

The National Semiconductor HPC family of microcontrollers includes as a feature the ability to be slaved to another "Host" processor over that processor's memory bus. This feature, called the Universal Peripheral Interface (or UPI) Port, allows:

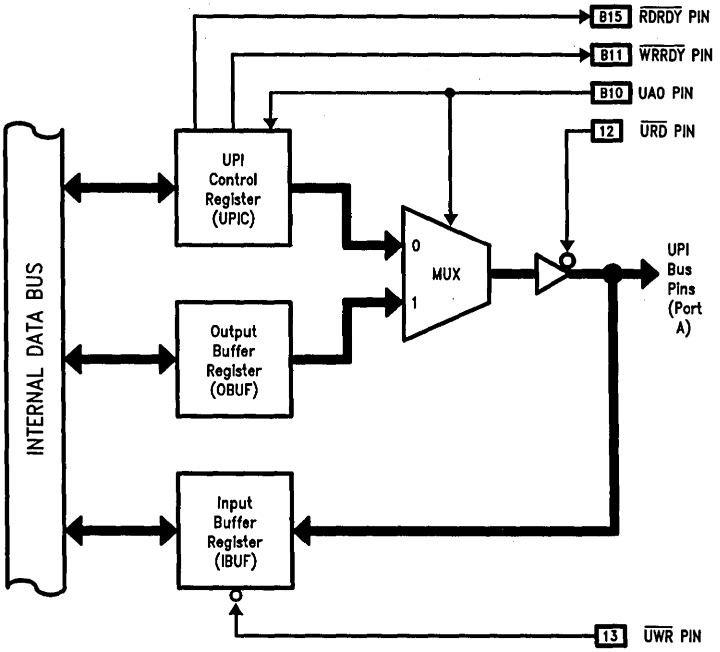
1. Transfer of either 8-bit or 16-bit data in a single bus transaction,
2. Polling to determine the status of the port from either side (Ready for Write/Ready for Read), and
3. Interruption of the host by the HPC with full vectoring.

The HPC, then, can serve as a front-end controller for the host, freeing it from control and/or communication tasks that might burden its capacity for interrupt service, and providing vectored interrupting for higher-level (and therefore less frequent) communication.

2.0 THE UPI PORT

2.1 Internal Structure

Figure 1 shows the internal structure of the UPI Port. It connects via three registers to the HPC's on-chip data bus, and via a set of pins (Port A) to the host's bus. The control interface between the HPC and the host consists of two low-active strobe signals ($\overline{\text{URD}}$ and $\overline{\text{UWR}}$) and an address signal (UA0) output by the host, and two handshake signals ($\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$) output from the HPC.



TL/DD/9976-1

FIGURE 1. UPI Internal Structure

The UPI Port may be configured either as a 16-bit bus (using all of Port A: pins A0–A15) or as an 8-bit bus (pins A0–A7), allowing pins A8–A15 to be used as general-purpose bit-programmable I/O pins. This selection is made by HPC firmware.

2.2 Basic Operations

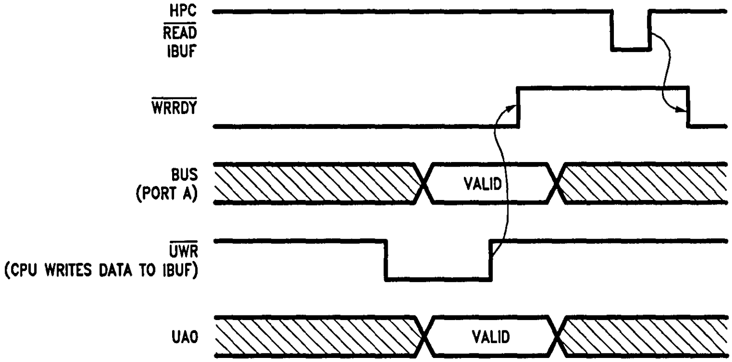
Three types of operation may be performed over the UPI Port:

1. Transfer of a byte or word of data from the host to the HPC's IBUF register. This is called a "UPI Write" operation.
2. Transfer of a byte or word of data from the HPC's OBUF register to the host. This is called a "UPI Read" operation.
3. Polling by the host to determine whether the HPC is ready for the next UPI Write or UPI Read operation. This involves the host reading the UPIC (UPI Control) register, which contains the states of the WRRDY and RDRDY pins as two of its bits.

As shown in *Figure 2*, whenever the host writes to the HPC (by pulsing the UWR signal low) data is latched into the HPC's IBUF register. At this time also, the value on the UA0 pin is latched into the UPIC (UPI Control) register, allowing HPC firmware to route the data just written. (For example, this bit can be used by the HPC firmware to distinguish between commands and data written to it.) The rising edge of UWR is detected by an edge-trigger circuit on-chip, which may be used to trigger an interrupt or for polling, to alert the HPC firmware to the presence of new data. The WRRDY handshake signal, normally low, goes high until the HPC firmware has sampled the data written to it (by reading internally from the IBUF register).

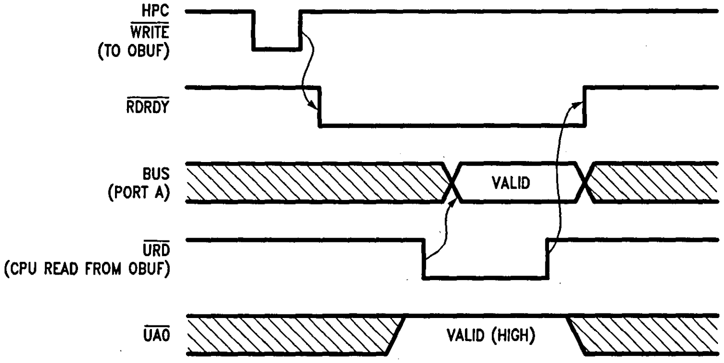
Figure 3 shows the sequence of events in reading data from the HPC. The transfer starts when the HPC writes a value to the internal OBUF register. The RDRDY handshake signal, normally high, goes low to indicate that data is present for the host. (This pin can be used to interrupt the host as well.) By pulsing the URD pin low while holding the UA0 pin to a "1", the host reads the contents of the OBUF register, and the RDRDY pin goes back high.

The polling operation *(Figure 4)* allows the host to monitor the RDRDY and WRRDY conditions as data bits, by pulsing the URD pin low with a "0" held on the UA0 pin. This effectively reads from the UPIC register; the WRRDY condition appears on bit 0 (the least-significant bit), and the RDRDY condition appears on bit 1 (the next most significant bit). Polling in this manner does not affect the state of the RDRDY bit.



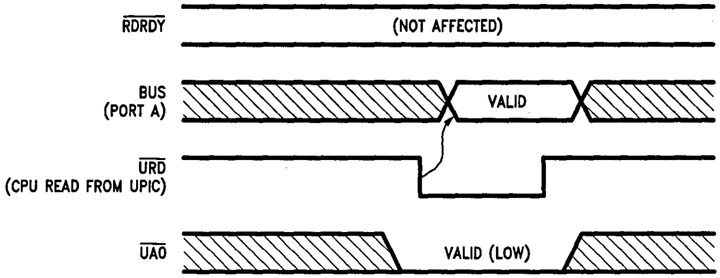
TL/DD/9976-2

FIGURE 2. UPI Write Operation



TL/DD/9976-3

FIGURE 3. UPI Read Data Operation



TL/DD/9976-4

FIGURE 4. UPI Poll Operation

2.3 Typical Hardware Configurations

Typical connections between the host and the HPC are shown in *Figures 5* through *7*.

2.3.1 Polled Synchronization

In the simplest case *(Figure 5)*, the $\overline{\text{WRRDY}}$ and $\overline{\text{RDRDY}}$ signals are not used, and the host synchronizes itself with the HPC strictly by polling the UPIC register for the Read Ready and Write Ready conditions. The only additional logic always required is a pair of OR gates to activate $\overline{\text{URD}}$ and $\overline{\text{UWR}}$ only when the HPC is selected by the host's address decoder. Depending on the host, it may also be necessary to add WAIT states, as is often required in peripheral interfaces to match the bus timing characteristics of the two ends.

Sophisticated synchronization schemes are not available using this simple an interface, but it does save the HPC $\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$ pins for any other general-purpose I/O functions.

2.3.2 Interrupt-Driven Synchronization

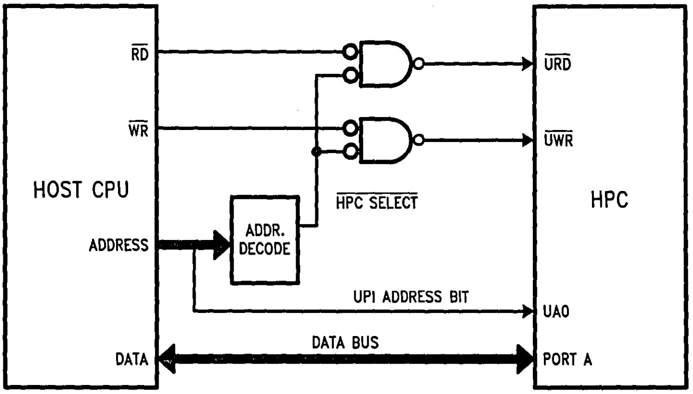
Assuming that the host has interrupt control capability, the circuit above can be enhanced to implement an interrupt-driven synchronization scheme, as shown in *Figure 6*. A falling edge on either $\overline{\text{RDRDY}}$ or $\overline{\text{WRRDY}}$ will trigger an interrupt to the host, informing it when the HPC becomes ready for either direction of data transfer. No additional logic is required (except for possible buffering or inversion), but only dedication of the $\overline{\text{WRRDY}}$ and/or $\overline{\text{RDRDY}}$ pins for the interrupt function. It is not necessary for both RDRDY and WRRDY conditions to trigger interrupts; one can be polled and the other interrupt-driven, as dictated by the requirements of the system and the structure of the host and HPC software. Also, depending on the host, it is often possible for the HPC itself to provide interrupt vectoring, thus eliminating the need for an external interrupt controller entirely. The approach taken in the driver program, described below, implements the HPC as the interrupt controller, with interrupts asserted only by the $\overline{\text{RDRDY}}$ pin.

2.3.3 Hardware Synchronization

Figure 7 shows the connections required to implement hardware synchronization between the host and the HPC. In this scheme, there is no host software involved in synchronizing with the HPC; if the host attempts a UPI transfer for which the HPC is not prepared, the host is held in "Wait states" until the HPC is ready. Note that the UPIC register is an exception; Wait states are not to be inserted when the CPU polls the UPI port's status (UA0 = 0).

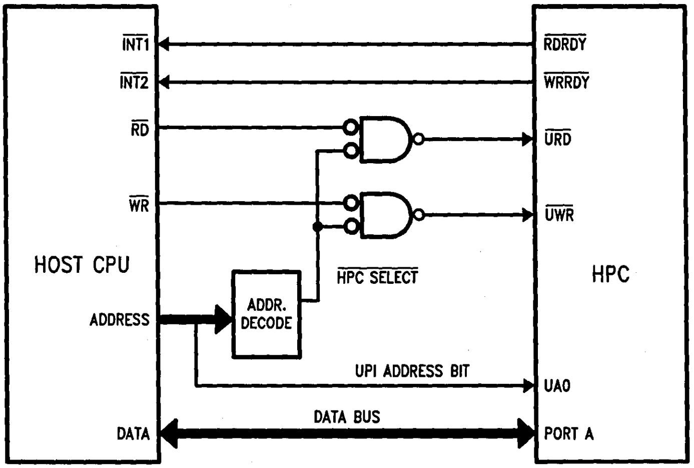
The main advantage of this scheme is speed: the CPU and HPC transfer data as fast as they can both run the transfer loop. (One will generally find that the HPC stays ahead of the CPU; the CPU tends to be in the critical path due to more complex buffer management algorithms.) The main disadvantage is that if the HPC is allowed to be interrupted in the middle of the transfer, the CPU is not free to do anything else at all, including servicing its own interrupts.

In addition to the logic to detect when to hold the host (at the bottom of the figure), additional gating is required on the $\overline{\text{UWR}}$ signal, to prevent it from being asserted until the $\overline{\text{WRRDY}}$ signal is active. This is required because the IBUF register of the HPC is a fall-through latch, and its contents would be lost if $\overline{\text{UWR}}$ were allowed to go active too soon.



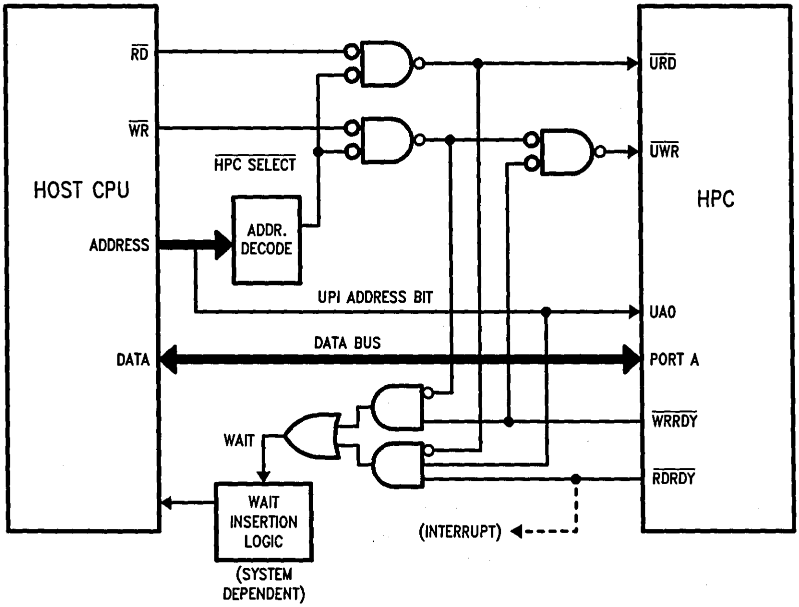
TL/DD/9976-5

FIGURE 5. Polling Interface



TL/DD/9976-6

FIGURE 6. Interrupt-Driven Interface

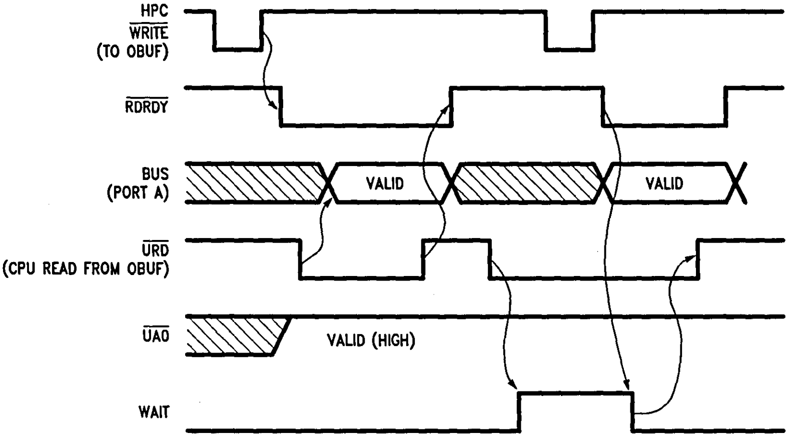


TL/DD/9976-7

FIGURE 7. Hardware-Synchronized Interface

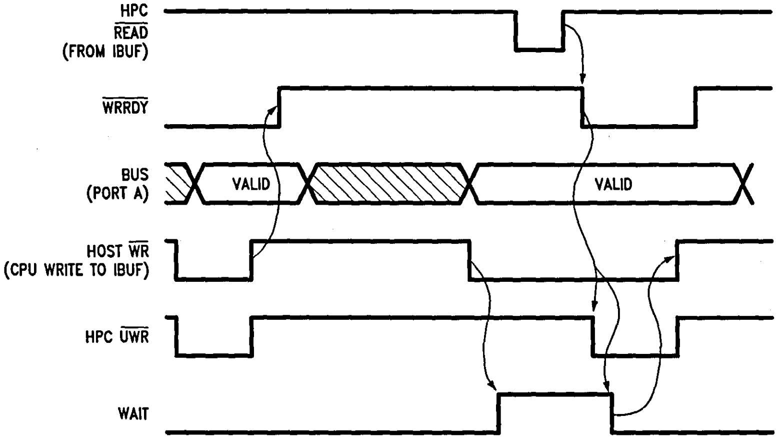
Figures 8 and *9* illustrate the timing involved in hardware synchronization. *Figure 8* shows the host attempting two UPI Read accesses in quick succession; the second Read access is held pending until the HPC has supplied the data. *Figure 9* shows the host attempting two UPI Write accesses in quick succession; it is held in Wait states (with the $\overline{\text{UWR}}$ signal suppressed) until the HPC has emptied the first value from the IBUF register.

This scheme and the interrupt-driven scheme above are not mutually exclusive; as shown in *Figure 6*, one might tie $\overline{\text{RDRDY}}$ or $\overline{\text{WRRDY}}$, or both, to CPU interrupts. The application hardware described implements both schemes, leaving CPU software the option of using hardware synchronization or not. The driver program in the HPC operates the same, independent of the option used.



TL/DD/9976-8

FIGURE 8. Hardware Synchronization: Read Operations



TL/DD/9976-9

FIGURE 9. Hardware Synchronization: Write Operations

3.0 A UPI DRIVER AND SAMPLE APPLICATION

The circuit and program described below implement an interface between the HPC and a National microprocessor, the NS32CG16, as the host CPU. The UPI port is configured to be 8 bits wide. The hardware supports both interrupt-driven ($\overline{RDRDY}$ only) and hardware synchronization, as well as polling.

In order to demonstrate some real commands to support, a set of simple interfaces is attached to the HPC, typical of a front panel.

— Up to 8 pushbuttons

— Up to 8 LED indicators

— A 16-character alphanumeric LCD display

— A speaker for "beeps" on alert conditions or input errors

— A real-time clock interrupt function, giving the CPU the means to measure time intervals accurately.

This application by itself is admittedly not enough to justify the presence of an HPC in a system, but it is a simple application, and we expect that this will often be part of the HPC's job. For a much more comprehensive application, which includes this one as a subset, see the next application note in this series: "The HPC as a Front-End Processor".

We will describe in this section a specific set of hardware and software, and a UPI command and response protocol to make these interfaces play.

3.1 UPI Port Connections to NS32CG16

The attached schematic shows the HPC UPI port as it has been used a real application. On Sheet 1, a block diagram is given, showing the components involved. The CPU is an NS32CG16 microprocessor, running at a 15 MHz clock rate (crystal frequency 30 MHz). The HPC component is the HPC46083, running at a crystal frequency of 19.6608 MHz.

It would be unrealistic to present only the UPI interface section, since tradeoffs and implementation considerations abound when dealing with fast processors and large addressing spaces. For this reason, we include on sheets 5, 6 and 7 the circuitry involved in NS32CG16 address decoding and dynamic RAM control.

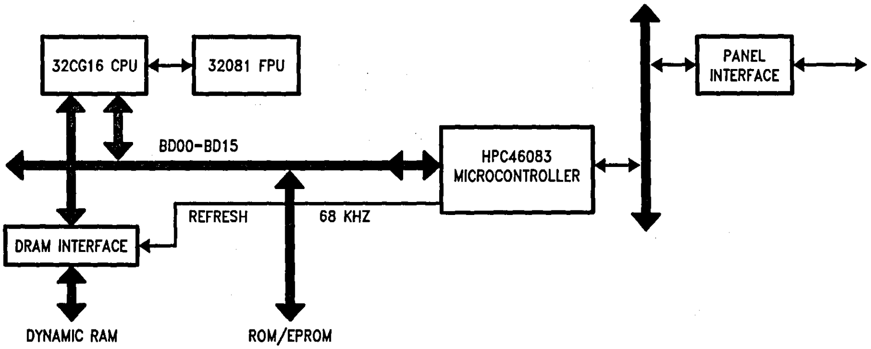
The $\overline{UREAD}$ and $\overline{UWRITE}$ UPI strobes are generated for the HPC in area B1 of Sheet 6. In addition, the latched CPU address bit BA09 is used as the UA0 addressing bit.

Hardware and Interrupt synchronization are accomplished as follows. On Sheet 6, area D8, the HPC signals $\overline{URDRDY}$ and $\overline{UWRRDY}$ enter a synchronizer, and emerge as $\overline{URDRDYS}$ and $\overline{UWRRDYS}$. The $\overline{URDRDYS}$ signal goes to the CPU as its Maskable Interrupt signal (Sheet 5, area C8). After gating, which yields $\overline{URDRDYSQ}$ and $\overline{UWRRDYSQ}$, they enter the PAL16L8 in area C7 of Sheet 6. This PAL's relevant outputs are $\overline{WAIT1}$ and $\overline{WAIT2}$, which go to the CPU for Wait State generation, and $\overline{ACWAIT}$, which also goes to the CPU (as $\overline{CWAIT}$) after passing through the PAL20R8 device in area D4 of Sheet 6.

In addition, the HPC provides from Timer T4 a square wave at approximately 68 kHz, which triggers refreshes of dynamic RAM. The signal involved is called "68 kHz", and goes from the HPC on Sheet 4, area D1, to Sheet 6, area D8. Note that the detector in area D7 is held on at Reset, to preserve RAM contents by continuous refreshing while the HPC is being reset.

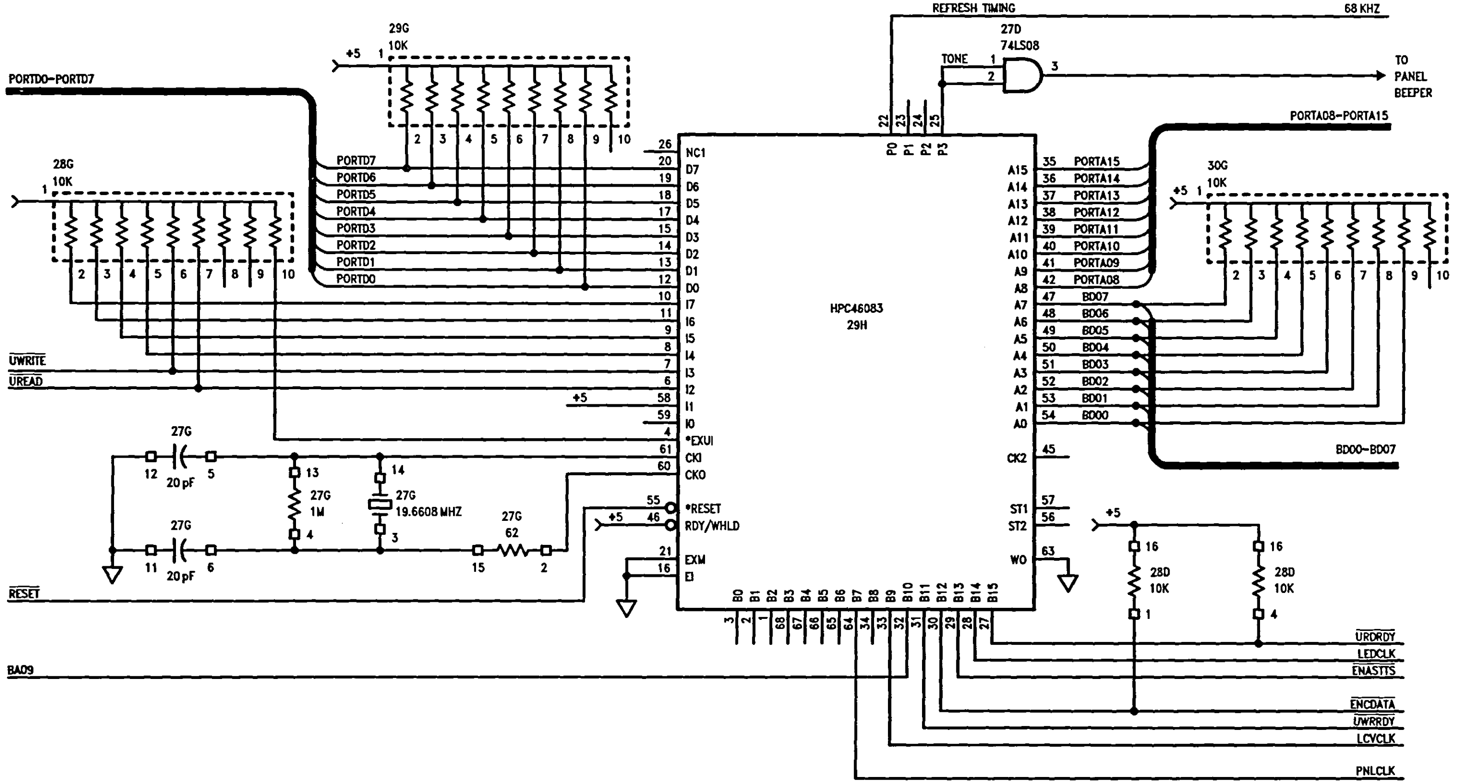
3.1.1 Schematic

UPI Demo Functional Block Diagram



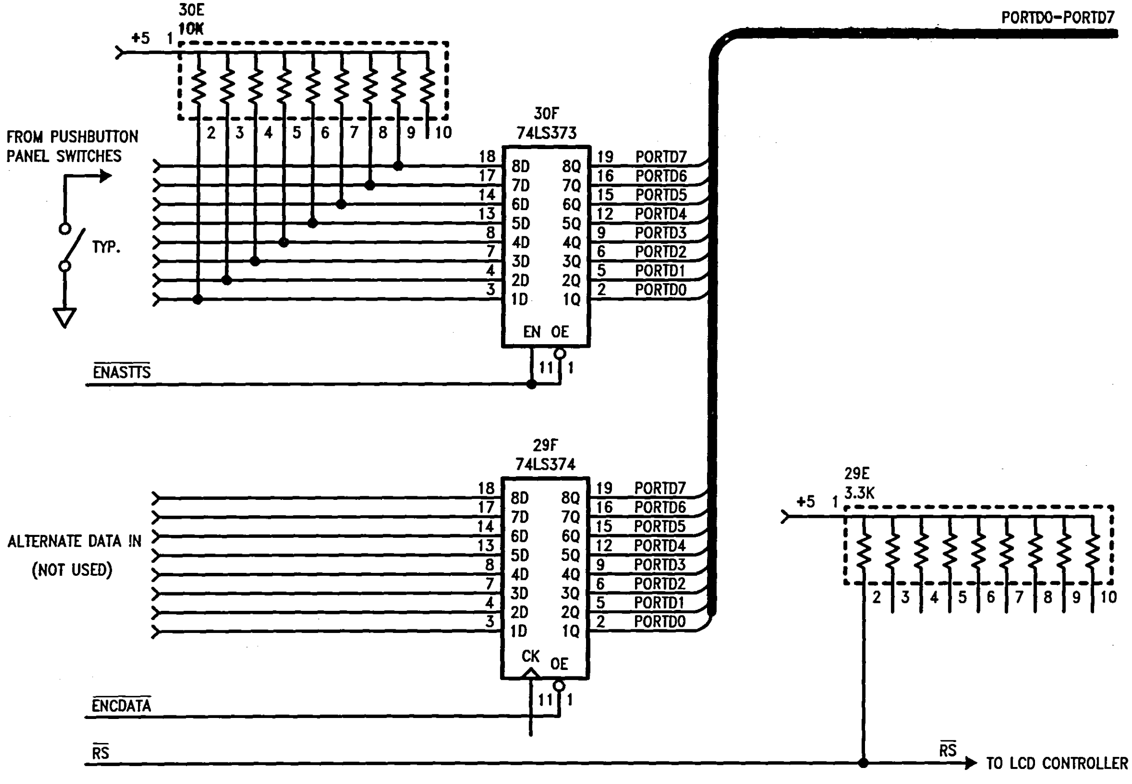
TL/DD/9976-10

HPC Microcontroller



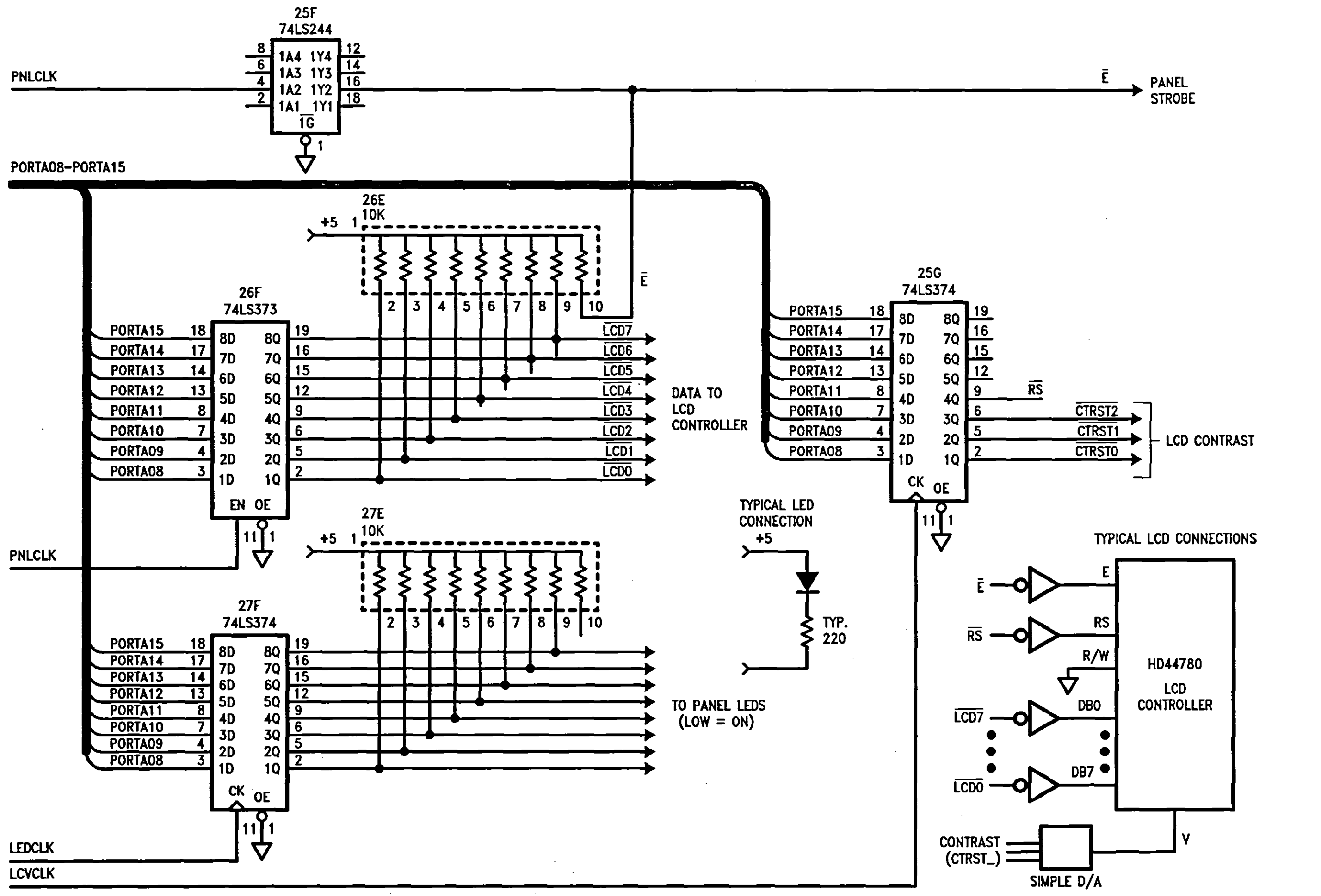
TL/DD/9976-11

HPC I/O



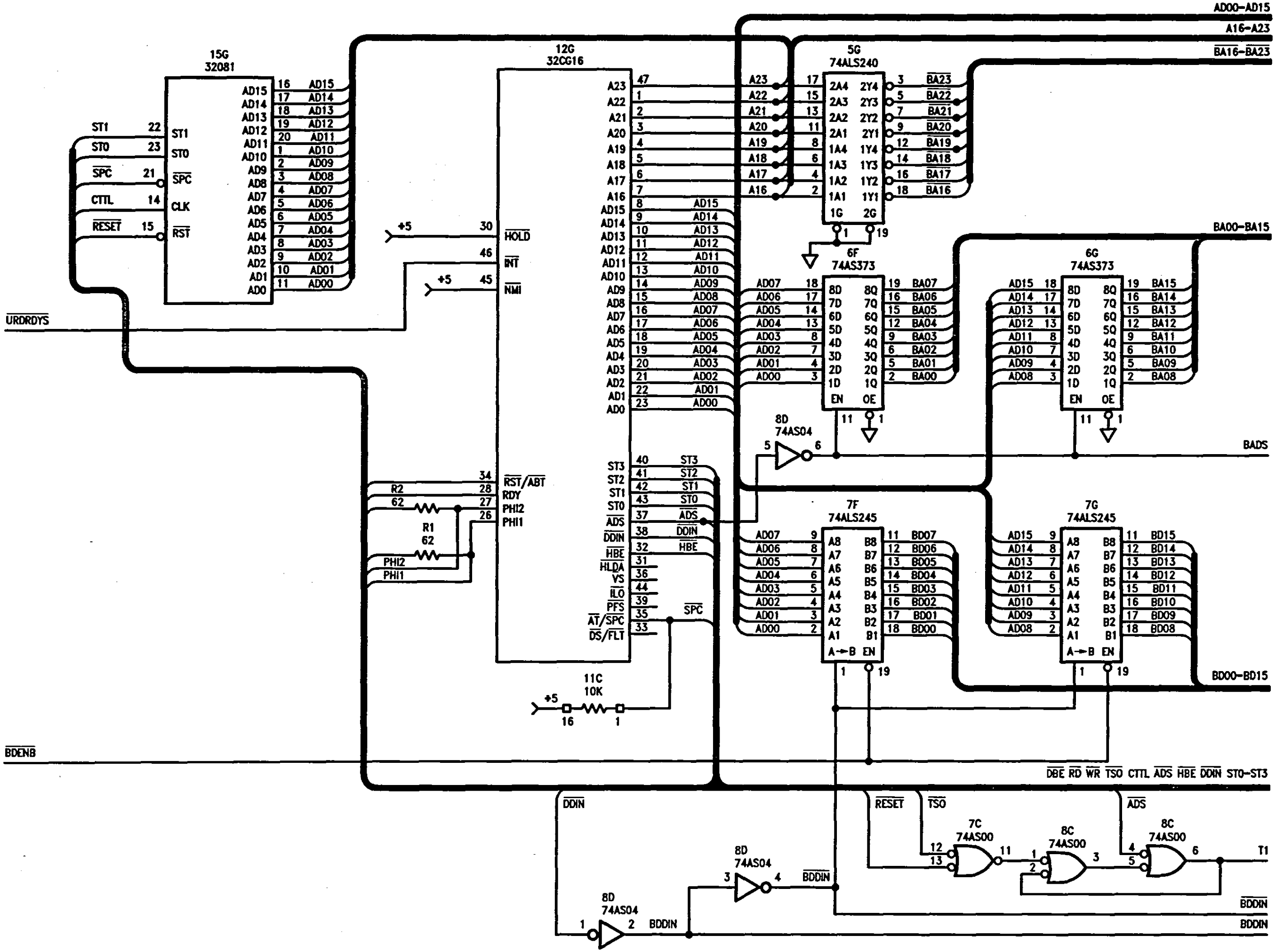
TL/DD/9976-12

Panel I/O Interface



TL/DD/9976-13

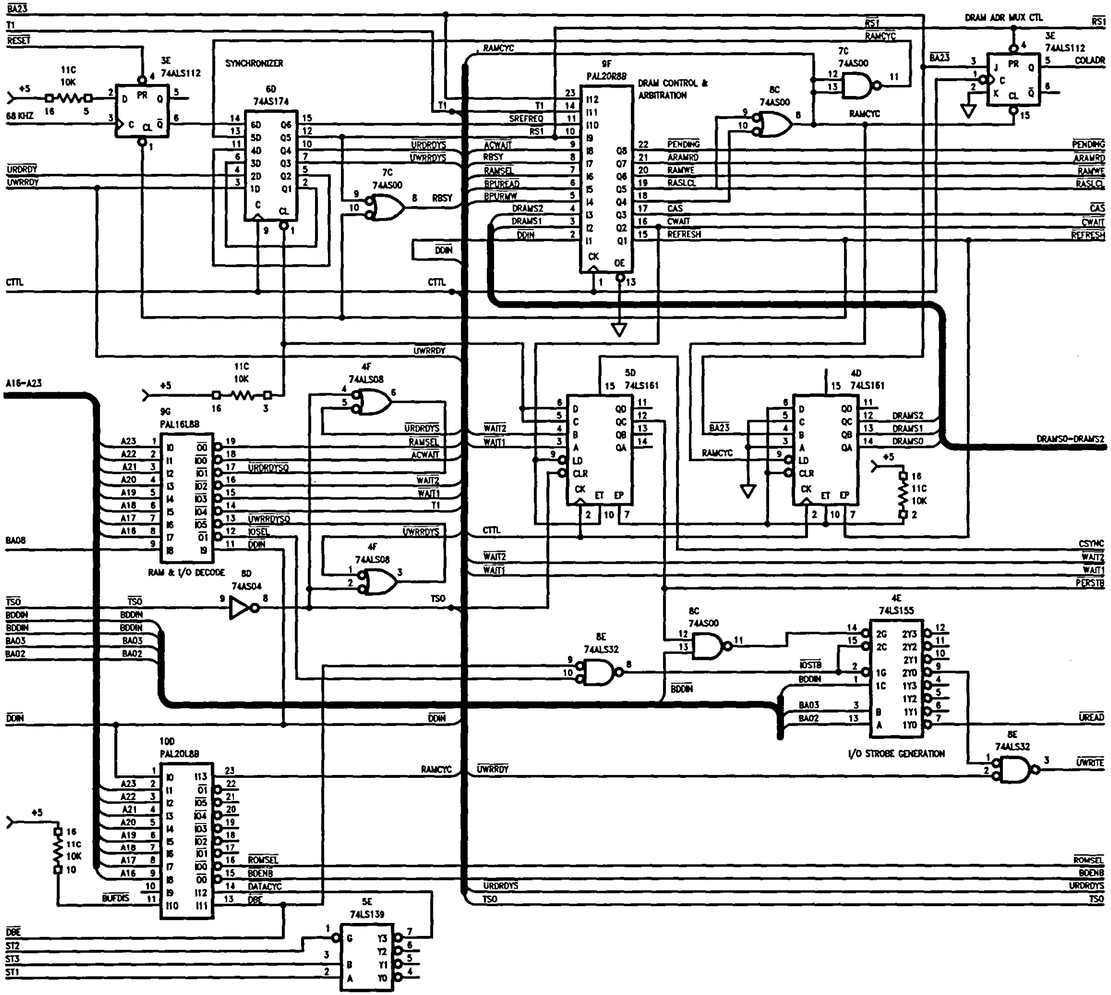
CPU Cluster and Buffering



AD00–AD15
A16–A23
BA16–BA23
BA00–BA15
BADS
BD00–BD15
15G
32081
ST1
ST0
SPC
CLK
RST
CTTL
RESET
AD15 AD14 AD13 AD12 AD11 AD10 AD9 AD8 AD7 AD6 AD5 AD4 AD3 AD2 AD1 AD0
12G
32CG16
+5
HOLD
INT
NMI
URDRDYS
RST/ABT
RDY
PHI2
PHI1
R2
62
R1
62
A23 A22 A21 A20 A19 A18 A17 A16
ST3
ST2
ADS
DDIN
HBE
HLDA
VS
ILO
PFS
AT/SPC
DS/FLT
11C
10K
5G
74ALS240
2A4 2A3 2A2 2A1 1A4 1A3 1A2 1A1
2Y4 2Y3 2Y2 2Y1 1Y4 1Y3 1Y2 1Y1
1G
2G
BA23 BA22 BA21 BA20 BA19 BA18 BA17 BA16
6F
74AS373
6G
74AS373
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN
OE
BA07 BA06 BA05 BA04 BA03 BA02 BA01 BA00
BA15 BA14 BA13 BA12 BA11 BA10 BA09 BA08
8D
74AS04
7F
74ALS245
7G
74ALS245
A8 A7 A6 A5 A4 A3 A2 A1
B8 B7 B6 B5 B4 B3 B2 B1
A→B
BD07 BD06 BD05 BD04 BD03 BD02 BD01 BD00
BD15 BD14 BD13 BD12 BD11 BD10 BD09 BD08
BDENB
DBE RD WR TSO CTTL ADS HBE DDIN ST0–ST3
RESET
TSO
7C
74AS00
8C
74AS00
T1
BDDIN
BDDIN

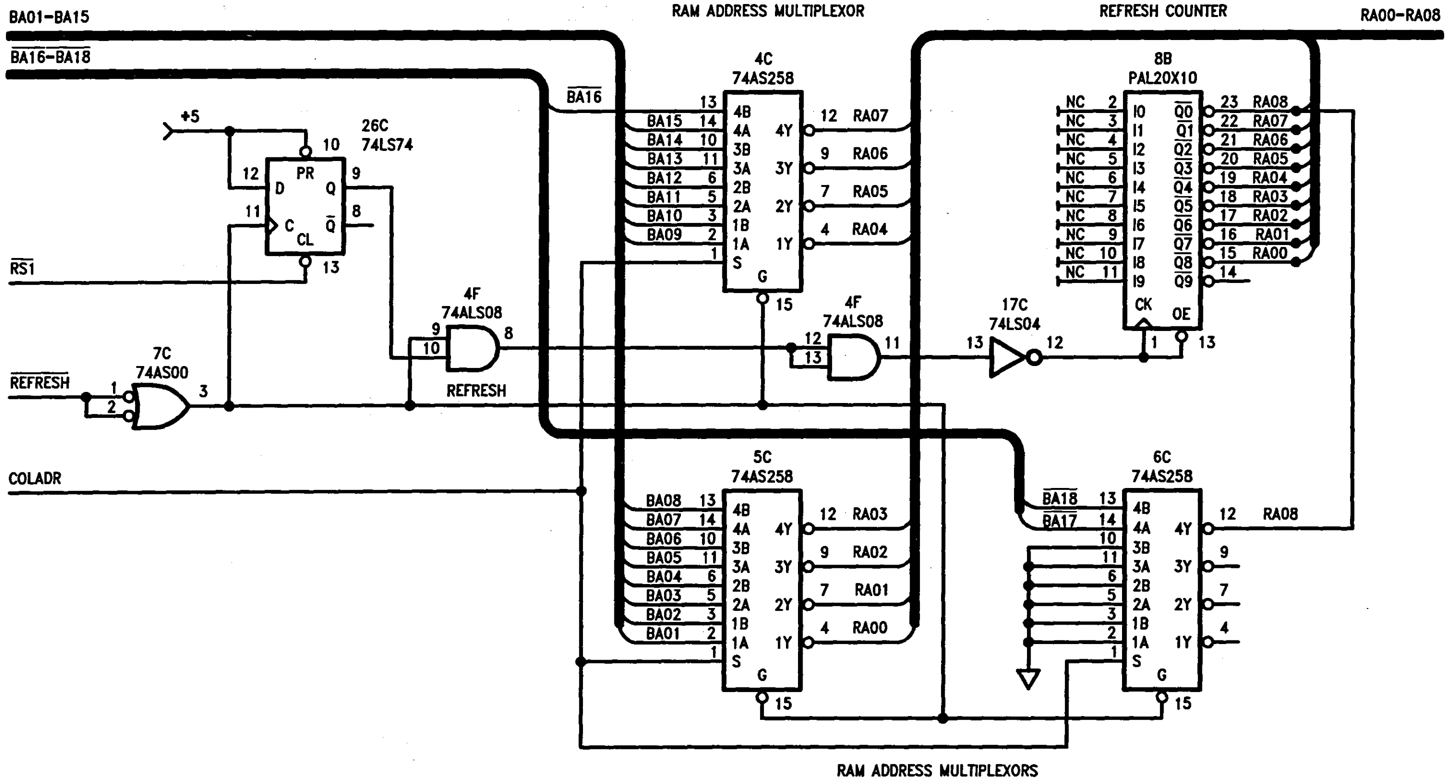
TL/DD/9976-14

Address Decoders and Timing Control Logic



TL/DD/9976-15

DRAM Address Generation



TL/DD/9976–16

3.1.2 PAL Equations

Schematic Sheet 7, Area 3D

```
Name        REFRESH.PLD;
Partno      XXXXX;
Date        05/19/87;
Revision    1A;
Designer    FOX;
Company     NSC;
Assembly    X7A;
Location    8B;
Device      p20x10;
/********************************************************************************************/
/*                                                                                          */
/*  REFRESH:  9 BIT REFRESH COUNTER                                                         */
/*                                                                                          */
/********************************************************************************************/
/*  Allowable Target Device Types: PAL20X10                                                 */
/********************************************************************************************/
/**  Inputs  **/
Pin     1       = !refresh     ;/* refresh pulse                                            */

/**  Outputs  **/
Pin     [15..23]= [ra0..8]     ;/* ram refresh address                                      */
Pin     14      = !refron      ;/* refresh enabled output                                   */
/**  Declarations and Intermediate Variable definitions **/
$define |       #

/**  Logic Equations  **/
!ra0.d  =        ra0;
!ra1.d  =        !ra1 $ ra0;
!ra2.d  =        !ra2 $ ra0 & ra1;
!ra3.d  =        !ra3 $ ra0 & ra1 & ra2;
!ra4.d  =        !ra4 $ ra0 & ra1 & ra2 & ra3;
!ra5.d  =        !ra5 $ ra0 & ra1 & ra2 & ra3 & ra4;
!ra6.d  =        !ra6 $ ra0 & ra1 & ra2 & ra3 & ra4 & ra5;
!ra7.d  =        !ra7 $ ra0 & ra1 & ra2 & ra3 & ra4 & ra5 & ra6;
!ra8.d  =        !ra8 $ ra0 & ra1 & ra2 & ra3 & ra4 & ra5 & ra6 & ra7;
refron.d=       'b'1;
```

Schematc Sheet 6, Area 5D

```
Name        RAM.PLD;
Partno      XXXXX;
Date        07/25/87;
Revision    1A;
Designer    FOX;
Company     NSC;
Assembly    X7A;
Location    9F;
Device      p20r8;
/*****************************************************************************************/
/*                                                                                       */
/*  RAM CONTROL: HARDWARE RMW BPU CYCLE, SEPARATE BUSES                                  */
/*  6/17: Two States of refadr                                                           */
/*  6/19: Invert rsl                                                                     */
/*****************************************************************************************/
/*  Allowable Target Device Types: PAL20R8B                                              */
/*****************************************************************************************/

/**  Inputs  **/

Pin    1    = cttl     ;    /* clock input                                               */
Pin    2    = !ddin    ;    /* data direction in signal                                  */
Pin    3    = dramsl   ;    /* DRAM state counter, bit 1                                 */
Pin    4    = drams2   ;    /* DRAM state counter, bit 2                                 */
Pin    5    = !bpurmw  ;    /* BPU read modify write cycle                               */
Pin    6    = !bpuread ;    /* BPU source read (comb.)                                   */
Pin    7    = !ramsel  ;    /* Any RAM address decode                                    */
Pin    8    = busy     ;    /* DRAM busy indication (rsl | refresh)                      */
Pin    9    = !acwait  ;    /* Advanced CWAIT from ROM, or I/O                           */
Pin    10   =!rsl      ;    /* ram cycle delayed by one Tstate                           */
Pin    11   = !srefreq ;    /* Refresh Request                                           */
Pin    14   = tl       ;    /*Processor Tl state                                         */
Pin    23    !a23      ;    /* Address 23                                                */

/**  Outputs  **/
Pin    15   = !refresh ;    /* refresh cycle                                             */
Pin    16   = !cwait   ;    /* 32C201 cwait                                              */
Pin    17   = !cas     ;    /* CAS, local & cartridge                                    */
Pin    18   = !rascart ;    /* RAS for DRAM cartridge                                    */
Pin    19   = !raslcl  ;    /* RAS for local DRAM                                        */
Pin    20   = !ramwe   ;    /* DRAM Write enable                                         */
Pin    21   = !aramrd  ;    /* DRAM read                                                 */
Pin    22   = !pending ;    /* DRAM cycle requested, but ctl busy                        */
min [refresh, cwait, cas, rascart, raslcl, ramwe, aramrd, pending] = 2;
/** Declarations and Intermediate Variable Definitions **/
field waitseq = [pending, cwait];
$define widle    0      /* wait sequencer idle */
$define busywt   3      /* wait sequencer waiting for busy DRAM */
$define cextwt   1      /* wait sequencer waiting for cycle extension */
```

Schematc Sheet 6, Area 5D (Continued)

```
field ctl = [refresh,cas,raslcl,rascart];
$define idle    00
$define cras    01
$define crascas 05
$define casend  04
$define lras    02
$define lrascas 06
$define refadr  08
$define refras  0b
$define |       #
field drscount = [drams2..dramsl];
/**  Logic Equations  **/
        lcl_sel         = ramsel & !a23;
        cart_sel        = ramsel & a23;

        lclread         = !a23 & ddin;
        lclwrite        = !a23 & !ddin;

        holdoff         rsl;
/*      busy            = refresh | holdoff;   (generated externally)                    */
        cart_start      = cart_sel & (tl | pending) & !holdoff;
        local_start     = lcl_sel & (tl | pending) & !holdoff;
        ram_start       = cart_start | local_start;
        drrco           = drscount: [6..7] & ramwe;
sequence waitseq {
/*      acwait & ramsel are mutually exclusive conditions  */
present widle     if (ramsel | bpurmw & bpuread) & busy & tl    next busywt;
                  if acwait | (ramsel & !busy & tl & !bpurmw)
                         next cextwt;
                  default next widle;
present busywt    if busy                                 next busywt;
                  if !busy & (bpurmw)                     next widle;
                  if !busy & !(bpurmw)                    next cextwt;
present cextwt    if ramsel & drscount: [0..1] | acwait   next cextwt;
                  default next widle;
}
sequence ctl {
present idle      if cart_start                           next cras;
                  if local_start                          next lras;
                . if !ram_start & srefreq                 next refadr;
                  default next idle;
present cras      if !rsl                                 next cras;
                  if rsl                                  next crascas;
present crascas   if ( !bpurmw & drscount: [4..7]) | (bpurmw & drrco)
                                                          next casend;
                  default next crascas;
present lras                                              next lrascas;
```

Schematc Sheet 6, Area 5D (Continued)

```
present lrascas if ( !bpurmw & drscount: [4..7]) | (bpurmw & drrco)
                                                       next casend;
                 default next lrascas;
present casend   if srefreq                            next refadr;
                 if !srefreq                           next idle;
present refadr   if srefreq                            next refadr;
                 if !srefreq & !rsl                    next refras;
                 if !srefreq & rsl                     next idle;
present refras   if ramwe                              next refadr;
                 default next refras;
}
/*  remember ramwe & aramrd are delayed by one t-state  */
ramwe.d = !refresh & (bpurmw & drscount: [6..7] & !ramwe
                         | !bpurmw & !ddin & (ram_start | ctl: cras
                                 | (cart_sel & drscount: [0..3]) | ctl:lras)
                 )
        | ctl:refras & rsl & !ramwe;

aramrd.d = (bpurmw & drscount: [0..3] | !bpurmw & ddin)
        & (ctl:cras | ctl:crascas | ctl:lras | ctl:lrascas);
```

Schematic Sheet 6, Area 7C

```
Name        DCD1.PLD;
Date        07/03/87;
Revision    1A;
Designer    FOX;
Company     NSC;
Assembly    X7A;
Location    9G;
Device      p1618;
/*********************************************************************************************/
/*                                                                                           */
/*  DECODE 1: I/O DECODE, PROM & HPC I/F WAIT CONTROL                                        */
/*  6/3: two waits for hpc write                                                             */
/*  6/4: 1 wait min. for ALL i/o, including HPC                                              */
/*  6/4: 3 wait min. for i/o                                                                 */
/*                                                                                           */
/*********************************************************************************************/
/*  Allowable Target Device Types: PAL16L8B                                                  */
/*********************************************************************************************/
/**  Inputs  **/
Pin         [1..8]    = [a23..16]     ;/* high order address bus                             */
Pin         9         = ba8           ;/* address bit 8                                      */
Pin         11        = !ddin         ;/* cpu ddin/                                          */
Pin         13        = !uwrrdys      ;/* (HPC) UWRRDY/, synchronized                        */
Pin         14        = t1            ;/* T1 state of CPU                                    */
Pin         17        = !urdrdys      ;/* (HPC) URDRDY/, synchronized                        */

/**  Outputs  **/
Pin         12        = !iosel        ;/* I/O select decode                                  */
Pin         15        = !wait1o       ;/* WAIT1 output                                       */
Pin         16        = !wait2o       ;/* WAIT2 output                                       */
Pin         18        = !acwait       ;/* Advance CWAIT for RAM ctl                          */
Pin         19        = !ramsel       ;/* DRAM address decode                                */

/**  Declarations and Intermediate Variable Definitions  **/
$define |             #
field address         = [a23..16]     ;/* address field                                      */
field waitv           = [acwait,wait2o,wait1o]; /* wait value field                          */
$define nowaits       "b'000
$define wait1v        ("b'100 & t1)
/*  note use of # in next 3 defines because $define not nestable                             */
$define wait2v                        ("b'101 & ("b'011 # "b'100 & t1))
$define wait3v                        ("b'110 & ("b'011 # "b'100 & t1))
$define wait4v                        ("b'111 & ("b'011 # "b'100 & t1))
$define cwaitonly "b'100

/**  Logic Equations  **/

ramsel                = address: [0780000..07fffff] | address: [0800000..0bfffff];
iosel                 = address: [0fd0000..0ffffff] & !ba8;
waitv                 = wait3v & address: [0000000..00fffff] /* main rom, 3 waits */
                      | wait4v & address: [0200000..05fffff] /* font rom, 4 waits */
                      | wait3v & address: [0fd0000..0ffffff] & !ba8 /* i/o, 1 wait */
                      | cwaitonly & address: 0ff0000 & !ba8 &
                                      ( !urdrdys & ddin | !uwrrdys & !ddin);
```

Schematic Sheet 6, Area 7A

```
Name        DCD2.PLD;
Partno      XXXXX;
Date        07/27/87;
Revision    1C;
Designer    FOX;
Company     NSC;
Assembly    X7A;
Location    10D;
Device      p2018;
/********************************************************************************************/
/*                                                                                          */
/*  DECODE 2: ROM DECODE, BUFFER CONTROL, BPU DECODE
/*  5/24: included enbpu in bpucyc generation                                               */
/*  5/28: added bpucyc to rdenb                                                             */
/*  5/31: added fcxxxx to bdenb                                                             */
/*  6/23: added buffer disable term for SPLICE                                              */
/*  7/25: reconfigured for bpurmw & bpuread                                                 */
/*  7/27: inverted polarity of enbpu ≥ enablebpu (for master enb)                           */
/********************************************************************************************/
/*  Allowable Target Device Types: PAL20B                                                   */
/********************************************************************************************/
/**  Inputs  **/
Pin         1       = !ddin      ;/* ddin/ from cpu                                         */
Pin         = [2..9]=[a23..16]   ;/* high order address bus                                 */
Pin         10      = !enablebpu ;/* BPU enable, static bit                                 */
Pin         11      = !bufdis    ;/* buffer disable                                         */
Pin         13      = !dbe       ;/* dbe/ from tcu                                          */
Pin         14      = !datacyc   ;/* data cycle status decode                               */
Pin         23      = ramcyc     ;/* ram cycle in progress                                  */
/**  Outputs  **/
Pin         15      = !bdenb     ;/* BD bus enable                                          */
Pin         16      = !romsel    ;/* Main rom select                                        */
Pin         17      = !romcart   ;/* rom cartridge select                                   */
Pin         18      = !bpurmw    ;/* BPU read modify write                                  */
Pin         19      = !bpuread   ;/* BPU read cycle (comb.)                                 */
Pin         20      = !vramsel   ;/* video ram select                                       */
Pin         21      = rdbufin    ;/* RAM data bus direction (in)                            */
Pin         22      = !rdenb     ;/* RAM data bus enable                                    */
/**  Declarations and Intermediate Variable Definitions  **/

field address       = [a23..16]  ;/* address field                                          */
romspace            = address: [0000000..05fffff];
ramspace            = address: [0780000..0bfffff];
stack               = address: [0780000..078ffff];

$define  |          #
            min b_ddin = 0;
/**  Logic Equations  **/
romsel = address: [0000000..00fffff];   /* main rom                                        */
romcart = address: [0200000..05fffff];  /* font rom                                        */
```

Schematic Sheet 6, Area 7A (Continued)

```
vramsel = address: [0f00000..0f0ffff];                      /* video ram (scan buffer)     */
/*
/*          bpucyc & b_ddin are D latches implemented in the PAL
/*
/*          basic d latch equation (w/o set or clear) is:
/*                     Q = (G & D) | ( !G & Q) | (D & Q)
/*
/*          The b_ddin latch is fall through while ramcyc not asserted,
/*          latched while ramcyc is asserted, therefore, for both latches:
*/
            g          = !ramcyc;
/*
/*          The bpurmw latch d input is ""bpurange'', defined as:
*/
            bpurange=  address:   [0000000..05fffff]        /* rom                         */
                       | address: [0790000..0bfffff];       /* dram, less stack            */
/*          This ""d'' input would use too many terms. The bpucyc output,
/*          however, need only be latched when it is asserted, as this is
/*          the situation that can allow the cpu and ram control to
/*          not be synchronized. This simplification allows the simplification
/*          of the latch to:
/*                     Q = D | ( !G & Q)
*/
            bpurmw = enablebpu & ( !ddin & bpurange & datacyc | ( !g & bpurmw));
            bpuread = enablebpu &  ddin & bpurange & datacyc;
/*          rdenb enables cpu access to the ram data bus
*/
            rdenb      = dbe & bufdis &
                       ( !bpurmw & bpuread & romspace       /* buffer must be off for bpu
                                                            /* but on for source in rom    */
                        /* no DRAM or bpu control writes are permitted                     */
                        /* while in inner loop of bitblt                                   */
                        /* (within interrupt ok due to vector read!)
                        | ramspace
                        | address: [0fe0000..0feffff]);     /* i/o access to bpu           */
            !rdbufin   = (ramspace | address: [0fe0000..0feffff]) & !ddin
                       | romspace & bpuread;
            bdenb      = dbe & !bufdis & (romspace          /* any rom                     */
                                | address: [0f00000..0f0ffff] /* scan buffer               */
                                | address: [0fd0000..0fdffff] /* cmnd/status               */
                                | address: [0ff0000..0ffffff] /* non-bpu i/o               */
```

3.2 Application Connections

The connections made to the HPC are shown in schematic sheets 2 through 4.

3.2.1 LCD Data

An 8-bit parallel interface connects the upper half of Port A, through buffers and latches on Sheet 4, to a Hitachi HD44780 alphanumeric LCD display controller. The signals in our application are inverted with respect to the HD44780 documentation, due to the nature of the front panel module we used.

Sending data from the HPC to the LCD display involves the following procedure:

1. Setup the $\overline{\text{RS}}$ signal: 1 for a command, 0 for data. This is done by setting up LCD Contrast status on the high-order byte of Port A (pins A8–A15), with the desired $\overline{\text{RS}}$ state on pin A11, then pulsing the signal LCVCLK (pin B9) high, the low.
2. Setup the panel data on HPC pins A8–A15.
3. Set the PNLCLK signal (pin B7) low for 1.2 μs, then high. This clocks the data into the LCD display controller. Note that the latch in area B6 of Sheet 4 is effectively serving only as a buffer; the PNLCLK Enable signal, being normally high, allows data to fall through whenever it changes when used as described here.
4. Since the handshaking capability of the HD44780 is not being used here, it is necessary for the HPC to use an internal timer to determine when the controller is ready after sending a command or data. The delay time is either 120 μs or 4.9 ms, depending on the type of command sent.

3.2.2. LCD Contrast (LCD Voltage)

A three-bit value is presented for LCD contrast on signals $\overline{\text{CTRST0}}$ through $\overline{\text{CTRST2}}$. A value of 000 is highest contrast, and 111 is lowest contrast. To change the contrast, the value is placed on HPC pins A8 (LSB), A9 and A10 (MSB), the LCVCLK (pin B9) is pulsed high, then low.

Note that some other bits within this latch have other functions: bit 3 (from HPC pin A11) is the $\overline{\text{RS}}$ signal to the LCD controller, and bit 7 (from pin A15) is used by the HPC firmware as a Fatal Error flag. These bits must be setup correctly whenever the LCD Contrast latch is written to.

3.2.3 LEDs

Up to 8 LED indicators may be connected, through the latch in area A6 of Sheet 4, to the upper byte of Port A. The LED's are assumed to be connected already to their own current-limiting resistors.

The desired data is setup on Port A pins A8–A15, then a pulse is presented on the LEDCLK signal (pin B14); high and then low. Data is presented in complemented form by the HPC (0 = on, 1 = Off). Any or all (or none) of the latch bits may be connected to drive LEDs.

3.2.4 Speaker (Beeper)

A tone is produced on a speaker by enabling Port P pin P3 as the Timer T7 output, and running Timer T7 so as to produce a 3 kHz square wave. Since timer outputs toggle on underflows, this corresponds to a timer underflow rate of 6 kHz. The tone signal is shown is area D1 of Sheet 2.

3.2.5 Pushbutton Switches

Up to eight pushbuttons may be connected to the HPC's Port D pins, through the buffer in area D6 of Sheet 3. Each pushbutton is assumed to be an SPST switch, shorting to ground when depressed. The pull-up resistors present a "1" level otherwise. The HPC must de-bounce the inputs in its firmware before issuing them to the CPU.

The pushbuttons are examined every 10 ms, by setting the $\overline{\text{ENASTTS}}$ signal (pin B13) low while ensuring that $\overline{\text{ENCDATA}}$ (pin B12) is high. This presents the switch outputs onto Port D. Unused bits should be pulled high to avoid triggering spurious pushbutton events.

3.3 Protocol Between CPU and HPC

The scheme supported by the UPI Driver program is asynchronous full-duplex communication with CPU. That is, either side is allowed to speak at any time. To avoid confusion, however, any message is restricted to send data in only one direction: in sequences initiated by the CPU ("Command" sequences), only the CPU talks, and in sequences initiated by the HPC ("Interrupt" sequences), only the HPC talks. Thus, a Command sequence and an Interrupt sequence can be in progress simultaneously without confusion.

Acknowledgement of a Command or an Interrupt sequency is possible; a Command can trigger an acknowledgement Interrupt sequence, and an Interrupt sequence can result in a subsequent Command sequence. The critical distinction, though, is that the acknowledgement need not come immediately. If, for example, the HPC is already in the process of sending an Interrupt message, and receives a Command, it will complete the current Interrupt sequence before acknowledging the Command with a new Interrupt.

Command sequences (from the CPU to the HPC) consist of a one-byte command code, followed by any argument values necessary to complete the command. Each byte written to the HPC triggers an internal interrupt (I3); the HPC buffers up these bytes until a full command has been received, then acts on it in the last byte's interrupt service routine. Commands taking a significant amount of processing time can be scheduled within the HPC using interrupts, either from external events or from one of the HPC's eight timers; each interrupt triggering the next step of the command.

Interrupt sequences (from the HPC to the CPU) operate similarly, but with a small difference. Only the first byte presented by the HPC causes an interrupt to the CPU; this byte is the interrupt vector value, which triggers the interrupt (through the $\overline{\text{RDRDY}}$ pin) and selects the CPU's service routine. The CPU remains in its interrupt service routine until the transfer of data associated with that interrupt event is finished, then returns to its previous task. This is not to say that the CPU must keep all other interrupts disabled during an Interrupt sequence, but only that no other interrupt occurring during this time may cause the CPU to read from the HPC, or to terminate reading, until the current Interrupt sequence is complete. With the NS32C016 processor as host, the main challenge is to keep the Interrupt Acknowledge bus cycles from other interrupts, which appear as Read cycles, from causing $\overline{\text{URD}}$ pulses to the HPC. It is possible to distinguish a Non-Maskable Interrupt from a Maskable Interrupt by the address asserted by the CPU in acknowledging the interrupt, and in a larger kind of system containing an NS32202 Interrupt Control Unit, the NS32000 Cascaded Interrupt feature can be used to prevent unwanted reads from the HPC from occurring as a result of other Maskable interrupts as well. In our application hardware, the only type of extraneous interrupt occurring is the Non-Maskable Interrupt; address decoding logic isolates the HPC's UPI port from these.

3.4 Commands

The first byte (command code) is sent to address FFFC00, and any argument bytes are then written to address FFFE00. The CPU may poll the UPIC register at address FD0000 to determine when the HPC can receive the next byte, or it can simply attempt to write, in which case it will be held in Wait states until the HPC can receive it. Unless noted, the CPU may send commands continuously, without waiting for acknowledgement interrupts from previous commands.

00 INITIALIZE — This command has two functions. The first INITIALIZE command after a hardware reset (or RESET command) enables the !RTC and !BUTTON-DATA interrupts. The INITIALIZE command may be re-issued by the CPU to either start or stop the !RTC interrupts. There is one argument:

RTC-Interval: One-byte value. If zero, !RTC interrupts are disabled. Otherwise, the !RTC interrupts occur at the interval specified (in units of 10 ms per count).

01 SET-CONTRAST — The single argument is a 3-bit number specifying a contrast level for the LCD panel (0 is least contrast, 7 is highest contrast). There is no response interrupt. Does not require INITIALIZE command first.

02 SEND-LCD — This writes a string of up to 8 bytes to the LCD panel. Arguments are:

flags: a single byte, containing the RS bit associated with each byte of data. The first byte's RS value is in the least-significant bit of the FLAGS byte.

#bytes: The number of bytes to be written to the LCD display.

byte[1]—byte[#bytes]: The data bytes themselves.

The HPC determines the proper delay timing required for command bytes (RS = 0) from their encodings. This is either 4.9 ms or 120 μs.

The response from the HPC is the !ACK-SEND-LCD interrupt, and this command must not be repeated until the interrupt is received. This command does not require an INITIALIZE command first.

03 SEND-LED — The single argument is a byte containing a "1" in each position for which an LED should be lit.

There is no response interrupt, and this command does not require the INITIALIZE command first.

04 BEEP — No arguments. This beeps the panel for approximately one second. No response interrupt. If a new BEEP command is issued during the beep, no error occurs (the buzzer tone is extended to one second beyond the most recent command). Does not require INITIALIZE command first.

A5 RESET-HPC — Resets the HPC if it is written to address FFFC00. It may be written at any time that the UPI port is ready for input; it will automatically cancel any partially-entered command. The CPU's Maskable Interrupt must be disabled before issuing this command.

After issuing this command, the CPU should first poll the UPIC register at address FD0000 to see that the HPC has input the command (the least-significant bit [Write Ready] is zero). It must then wait for at least 25 μs, then read a byte from address FFFE00. The HPC now begins its internal re-initialization. The CPU must wait for at least 80 μs to allow the HPC to re-initialize the UPI port. Since part of the RESET procedure causes Ports A and B to float briefly (this includes the CPU's Maskable Interrupt input pin), the CPU should keep its maskable interrupt disable during this time. It also must not enter a command byte during this time because the byte may be lost.

3.5 Interrupts

The HPC interrupts the CPU, and provides the following values as the interrupt vectors for the CPU hardware. The CPU then reads data from the HPC at address FFFE00. All data provided by the HPC must be read by the CPU before returning from the interrupt service routine, otherwise the HPC would either hang or generate a false interrupt. The CPU may poll the UPIC register at address FD0000 to determine when each data byte is ready, or it may simply attempt to read from address FFFE00, and it will be held in Wait states until the data is provided by the HPC.

Note: All CPU interrupt service routines, including the NMI interrupt routines, must return using the "RETT 0" instruction. Do NOT use "RETI".

00–0F (Reserved for CPU internal traps and the NMI interrupt.)

11 !RTC — Real-Time Clock Interrupt. No data returned. Enabled by INITIALIZE command if interval value supplied is non-zero. Note: this version of HPC firmware issues a non-fatal !DIAG interrupt if the CPU fails to service each !RTC interrupt before the next one becomes pending.

17 !ACK-SEND-LCD — This is the response to the SEND-LCD command, to acknowledge that data has all been written to Panel LCD display. No other data is provided with this interrupt. Always enabled, but occurs only in response to a SEND-LCD command.

18 !BUTTON-DATA — Pushbutton status has changed: one or more buttons have been either pressed or released. The new status of the switches is reported in a data byte, encoded as follows:

Any pushbutton that is depressed is presented as a "1". All other bit positions, including unused positions, are zeroes. The pushbuttons are debounced before being reported to the CPU. This interrupt is enabled by the first INITIALIZE command after a reset.

1D !DIAG Diagnostic Interrupt. This interrupt is used to report failure conditions and CPU command errors. There are five data bytes passed by this interrupt:

Severity
Error Code
Data in Error (passed, but contents not defined)
Current Command (passed, but contents not defined)
Command Status (passed, but contents not defined)

The Severity byte contains one bit for each severity level, as follows:

x	x	x	F	x	x	C	N

N (Note): least severe. The CPU missed an event; currently only the !RTC interrupt will cause this.

C (Command): medium severity. Not currently implemented. Any command error is now treated as a FATAL error (below).

F (Fatal): highest severity: the HPC has recognized a non-recoverable error. It must be reset before the CPU may re-enable its Maskable Interrupt. In this case, the remaining data bytes may be read by the CPU, but they will all contain the value 1D (hexadecimal). The CPU must issue a RESET command, or wait for a hardware reset. See below for the procedure for FATAL error recovery.

The Error Code byte contains, for non-FATAL errors, a more specific indication of the error condition:

RTC	(Reserved for COMMAND)						

RTC = Real-Time Clock overrun: CPU did not acknowledge the RTC interrupt before two had occurred.

The other bits are reserved for details of Command errors, and are not implemented at this time.

The remaining 3 bytes are not yet defined, but are intended to provide details of the HPC's status when an illegal command is received.

Note: Except in the FATAL case, all 5 bytes provided by the HPC *must* be read by the CPU, regardless of the specific cause of the error.

Fatal Error Recovery:

When the HPC signals a !DIAG error with FATAL severity, the CPU may use the following procedure to recover:

1. Write the RESET command (A5 hex) to the HPC at address FFFC00.
2. By inspecting the UPIC register at address FD0000, wait for the HPC to read the command (the *WRRDY bit will go low).
3. Wait an additional 25 μs.
4. Read from address FFFE00. This will clear the OBUF register and reset the Read Ready status of the UPI port. The HPC will guarantee that a byte of data is present; it is not necessary to poll the UPIC register. This step is necessary because only a hardware reset will clear the Read Ready indication otherwise (HPC firmware cannot clear it).
5. Wait at least 80 μs. This gives the HPC enough time to re-initialize the UPI port.
6. After Step 5 has been completed, the CPU may re-enable the Maskable Interrupt and start issuing commands. Since the HPC is still performing initialization, however, the first command may sit in the HPI IBUF register for a few milliseconds before the HPC starts to process it.

4.0 SOURCE LISTINGS AND COMMENTARY

4.1 HPC Firmware Guide

Refer to this section for help in following the flow of the HPC firmware in the listing below. Positions in the code are referenced by assembly language labels rather than by page or line numbers.

The firmware for the HPC is almost completely interrupt-driven. The main program's role is to poll mailboxes that are maintained by the interrupt service routines, and to send an interrupt to the CPU whenever an HPC interrupt routine requests one in its mailbox.

On reset, the HPC firmware begins at the label "start". However, the first routine appearing in ROM is the Fatal Error routine. This was done for ease of breakpointing, to keep this routine at a constant address as changes were made elsewhere in the firmware.

4.1.1 Fatal Error Routine

At the beginning of the ROM is a routine (label "hangup") that is called when a fatal error is detected by the HPC. This routine is usually called as a subroutine (although it never returns). It disables HPC internal interrupts, and then sets bit 7 of the LCD Contrast Latch as a trigger for a logic analyzer, MOLE or ISE system.

Its next action is to display its subroutine return address in hexadecimal on the LCD panel. This address shows where the error was detected. The HPC then enters an infinite loop, which continuously presents the !DIAG interrupt. It may be terminated either by a hardware reset or by sending the RESET command from the CPU. On receiving the RESET command, the HPC jumps to label "xreset", which is within the command processing routine. The "xreset" routine waits for the CPU to read from the UPI port, then clears a set of registers to simulate a hardware reset and jumps to the start of the program.

4.1.2 Initialization

On receiving a Reset signal, the HPC begins execution at the label "start". A required part of any application is to load the PSW register, to select the desired number of Wait states (without this step, the Reset default is 4 Wait states, which is safe but usually unnecessary).

Other initializations here are application-dependent, and so they relate to our application system and front-panel operations.

At label "srfsh", the program starts the Refresh clock pulses running for the dynamic RAM on our application hardware, from HPC pin P0 (controlled by Timer T4). For debugging purposes, a circuit within the RAM controller section performs continuous refreshes during Reset pulses, so data in dynamic RAM is never lost unless power is removed.

At "supi", the UPI port is initialized for transfers between the HPC and the CPU.

At label "sram", all RAM within the HPC is initialized to zero. This is done for debugging purposes, to help ensure that programming errors involving uninitialized data will have more consistent symptoms.

At "sskint", the stack pointer is initialized to point to the upper bank of on-chip RAM (at address 01C0). The address of the fatal error routine "hangup" is then pushed, so that it will be called if the stack underflows. This is not necessary in all applications, since the Stack Pointer starts at address 0002, but for our purposes it was more convenient to relocate it.

At "tminit", the timers T1–T3 are stopped and any interrupts pending from timers T0–T3 are cleared.

In addition, some miscellaneous port initializations are performed here. The upper byte of Port A is set as an output port (for data going to the LCD and LED displays), and the Port B pins which select pushbutton data are initialized.

At "sled", the LED control signals are initialized, and all LED indicators on the panel are turned off.

At "stmrs", all timers are loaded with their initial values, and timers T5–T7 are stopped and any interrupts pending from them are cleared. (Timer T4 keeps running for dynamic RAM refresh.)

At "sled", the panel LCD display is initialized to a default contrast level of 5, then commands are sent to initialize it to 8-bit, 2-line mode, with the cursor visible and moving to the right by default. This section calls a subroutine "wrpnl", located at the end of the program, which simply writes the character in the accumulator out to the LCD display and waits for approximately 10 ms. Note that if the CPU fails to initialize the LCD display further, a single cursor (underscore) character is all that appears: a recognizable symptom of a CPU problem.

The program now continues to label "minit", which performs some variable initializations which are necessary for operation of the UPI Driver itself (as opposed to the application). This much must always be present, but any other initializations required by the application should appear here as well. For our front-panel application, there are no such initializations required.

At label "runsys", the necessary interrupts are enabled (from the timers, and from pin I3, which is the UPI port interrupt from the CPU), and the program exits to the Main Program loop at label "mainlp".

4.1.3 Main Program (UPI Output to CPU)

The Main Program is the portion of the UPI Driver that runs with interrupts enabled. It consists of a scanning loop at label "mainlp", calling a set of subroutines (explained below). It is responsible for interrupting the CPU and passing data to it. The HPC is allowed to write data to the CPU only after interrupting it. The main loop scans a bit-mapped variable in on-chip RAM that is set up by interrupt service routines (a word called "alert") to determine whether any conditions exist that should cause an interrupt to the CPU.

The "alert" word contains one bit for each interrupt that the HPC can generate. If a bit is set (by an interrupt service routine), the Main Program jumps to an appropriate subroutine to notify the CPU. Each subroutine first checks whether the UPI interface's OBUF register is empty, and if not, it waits (by calling the subroutine "rdwait"). It then writes the 32000 interrupt vector number to the OBUF register. This has the effect of interrupting the CPU (Because the pin $\overline{\text{URDRDY}}$ goes low), and the CPU hardware reads the vector from the OBUF register. If there is more information to give to the CPU, the HPC places it, one byte at a time, into the OBUF register, waiting each time for OBUF to be emptied by the CPU. This technique assumes that the CPU remains in the interrupt service routine until all data has been transferred. If the CPU were to return from interrupt service too early, the next byte of data given to it would cause another interrupt, with the data value taken as the vector number. (Note, however, that a Non-Maskable interrupt is allowed. It simply delays the process of reading data from the HPC. Since the HPC is running its main program at this point, with its internal interrupts still enabled, it is not stalled by this situation.)

Subroutines called from the Main Program loop are:

sndrtc: sends a Real-Time Clock interrupt to the CPU. No data is transferred; only the interrupt vector.

sndlak: interrupts the CPU to acknowledge that a string of data (from a SEND-LCD command) has been written to the LCD display. No data is transferred for this interrupt.

sndbtn: interrupts the CPU to inform it that a pushbutton has been pressed or released. A data byte is transferred from variable "swlsnt", which shows the new states of all the pushbuttons.

sndiag: interrupts the CPU to inform it of a !DIAG interrupt condition, when it is of NOTE severity. (Other !DIAG conditions are handled at label "hangup".)

4.1.4 Interrupt Service Routines

All of the remaining routines are entered by the occurrence of an interrupt.

4.1.4.1 UPI Port Input from CPU (Interrupt I3)

This interrupt service routine, at label "upiwr", accepts commands from the CPU. Each byte of a command triggers an interrupt on the I3 pin. When the last byte is received, the command is processed before the I3 interrupt routine returns. The HPC is therefore immediately ready to start collecting another command.

Any command that involves waiting is only initiated before the I3 routine returns, and interrupts are set up to activate more processing when the time is right. Therefore, this interrupt service routine returns promptly, even for time-consuming commands.

At any time, the "upiwr" routine may be in one of the following states:

1. Waiting for the first byte of a command. In this state, the variable "curcmd" (Current Command) has its top bit ('cmdemp") set, meaning that it is empty. When a byte is received from the CPU in this state, this routine jumps to the label "firstc". The byte is placed in the "curcmd" byte (clearing the top bit), and then a multi-way branch (jidw) is performed, whose destination depends on the contents of the byte. The possible destinations have labels starting with the letters "fc". If the command has only one byte (for example, the command BEEP), it is processed immediately in the "fc" sequence, and the "curcmd" variable is set empty again. If, however, the command is longer than one byte, its "fc" routine will place a value into the variable "numexp", which gives the number of additional bytes that are expected for this command, and then will return from the interrupt. Note that the "curcmd" byte now appears to be full, because its top bit is no longer set.
2. Collecting bytes of a command. The code that is relevant in this state is between the labels "upiwr" and "lastc". This state is in effect while the "cmdemp" bit of "curcmd" is zero and the "numexp" variable is non-zero. Each I3 interrupt causes the routine to place the command byte into a buffer ("cpubuf", with pointer variable "cpuad"), decrement the "numexp" variable, and return if the result is non-zero. If the result is zero, then the routine has collected an entire command, and it goes to the label "lastc", and enters state (3) below.
3. In this state, the requested number of bytes has been collected, and this usually means that the entire command, except for the first byte, is in the "cpubuf" area of RAM. The code for this state is at label "lastc". First, the "curcmd" byte is checked to see whether "extended collection" is being performed (bit 6 set: see below). If not, the "curcmd" byte is set empty. A multiway branch is then performed (jidw), which transfers control depending on the command byte in "curcmd". All routines that are destinations of this branch start with the letters "lc". The "lc" routine for each command uses the data in "cpubuf" to process the current command. In some cases, this processing is completed very quickly. For example, at label "lcsled", a value is simply transferred from "cpubuf" to a latch that drives the LEDs on the front panel, and this interrupt service routine returns. But a more complex command can move data out of "cpubuf" to other variables in RAM, and start a timer to sequence the process of executing the command.

In some commands (for example, SEND-LCD), state (3) above is entered twice. This is called "extended collection", and occurs when a command has variable length. State (3) is entered once to collect enough information to determine the exact length of the command. It then sets up the "numexp" variable again, re-entering state (2) to collect the remainder of the command. When state (3) is entered the second time, it processes the command. A bit in the "curcmd" variable (bit 6, called "getcnt") is set in state (1), which indicates that another collection will be performed, and prevents state (3) from setting the "curcmd" byte empty the first time it is entered.

Command Processing Routines

INITIALIZE　I3 interrupt labels:　State 1 = fcinit　State 3 = lcinit

SET-CONTRAST　I3 interrupt labels:　State 1 = fcslcv　State 3 = lcslcv

At label "lcslcv" (Set LCD Voltage), the LCD Contrast latch is loaded from the value supplied by the CPU.

SEND-LCD　I3 interrupt labels:　State 1 = fcslcd　State 3 = lcslcd

This command uses the "extended collection" feature. At label "fcslcd", two bytes are requested for collection, but the "getcnt" bit of "curcmd" is set, meaning that these are not the last bytes of the command. At label "lcslcd" (jumping to label 'lcslc1"), the length of the instruction is determined from the #bytes value supplied by the CPU, and a second collection of bytes is requested, this time with the "getcnt" bit off. When the last byte has been collected, control is transferred to the label "lcslcd", then to "lcslc2". Here, the data bytes for the panel are unloaded from the CPU buffer area "cpubuf" into the LCD string buffer "lcdbuf". The flag (RS) bits are loaded into variable "lcdsfg", and the number of bytes to be sent to the LCD display is placed into variable "lcdsct'. Timer T6 is now started, to provide scheduling interrupts for writing the bytes from the LCD string buffer to the LCD display.

On occurrence of each T6 interrupt (labels "t6int" and "t6nxtc"), one byte is written to the LCD display. Depending on the state of the RS flag for that byte, and the value sent to the panel, T6 may run for either 120 μs or 4900 μs before it triggers the next transfer. When the last character has been transferred, and Timer T6 has provided the proper delay after it, the bit "alcdak" is set in the "alert" word, requesting the main program to send an !ACK-SEND-LCD interrupt to the CPU.

SEND-LED	I3 interrupt labels:	State 1 = fcsled	·	State 3 = lcsled
At label "lcsled", the byte provided by the CPU is written to the LED latch.

BEEP	I3 interrupt labels:	State 1 = fcbeep	State 3 = (none)
At label "fcbeep", Port P pin P3 is enabled to toggle on each underflow of Timer T7, which has been initialized at the beginning of the program (label "stmrs") to underflow at a rate of 6 kHz. Pin P3, then, presents a 3 kHz square wave to the panel buzzer. To time out the duration of the beep tone, interrupts from Timer T0 are enabled, which then occur once every 53 ms. The variable "beepct" is set up with the number of T0 interrupts to accept, and is decremented on each T0 interrupt. When it has been decremented to zero (meaning that one second has elapsed), pin P3 is reset to a constant zero to turn off the tone.

4.1.4.2 Background Timer (T1) Task

The Timer T1 interrupt service routine represents a task that is not triggered directly by CPU commands. Its functions are to interrupt the CPU periodically for the Real-Time Clock function, and to present the !BUTTON-DATA interrupt whenever the pushbutton inputs change state.

Timer T1 is loaded with a constant interval value which is used to interrupt the HPC at 10 ms intervals. When the Timer T1 interrupt occurs (labels 'tmrint", to "t1poll", to "t1int"), then if the real-time interrupt is enabled, the variable "rtccnt" is decremented to determine whether an !RTC interrupt should be issued to the CPU. If so, the bit "artc" in the "alert" word is set, requesting the main program to issue the interrupt. The main program, at label "sndrtc", actually interrupts the CPU. No other data is passed to the CPU with the interrupt.

At label "kbdchk" the panel pushbutton switches are also sampled. If the pattern matches the last sample taken (saved in variable "swlast") then it is considered to be stable, and it is then compared to the last switch pattern sent to the CPU (in variable "swlsnt"). If the new pattern differs, then it is placed in "swlsnt", and the bit "abutton" in variable "alert" is set, requesting the main program to send a !BUTTON-DATA interrupt. The main program, at label "sndbtn", triggers the interrupt and passes the new pattern to the CPU from variable "swlsnt".

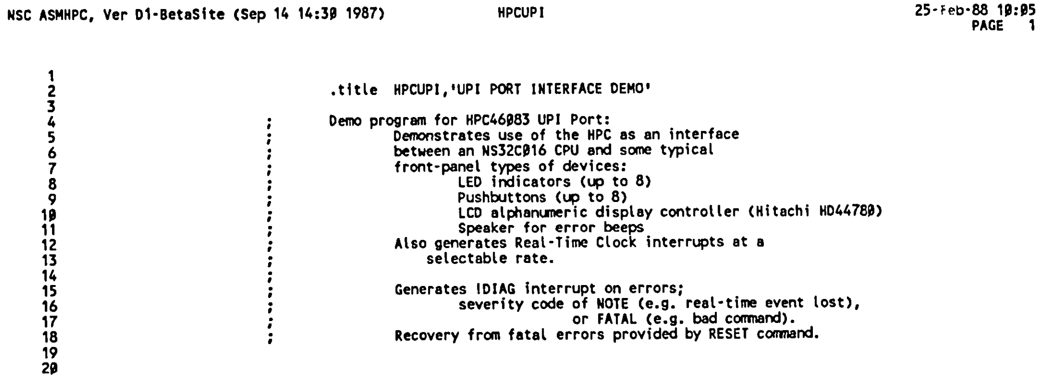
4.1.4.3 Timer T6 Interrupt

Because the LCD controller's command acknowledgement capability was not used in our application, Timer T6 is used to time out the LCD controller's processing times. See the description of the SEND-LCD command above.

4.1.4.4 Timer T0 Interrupt

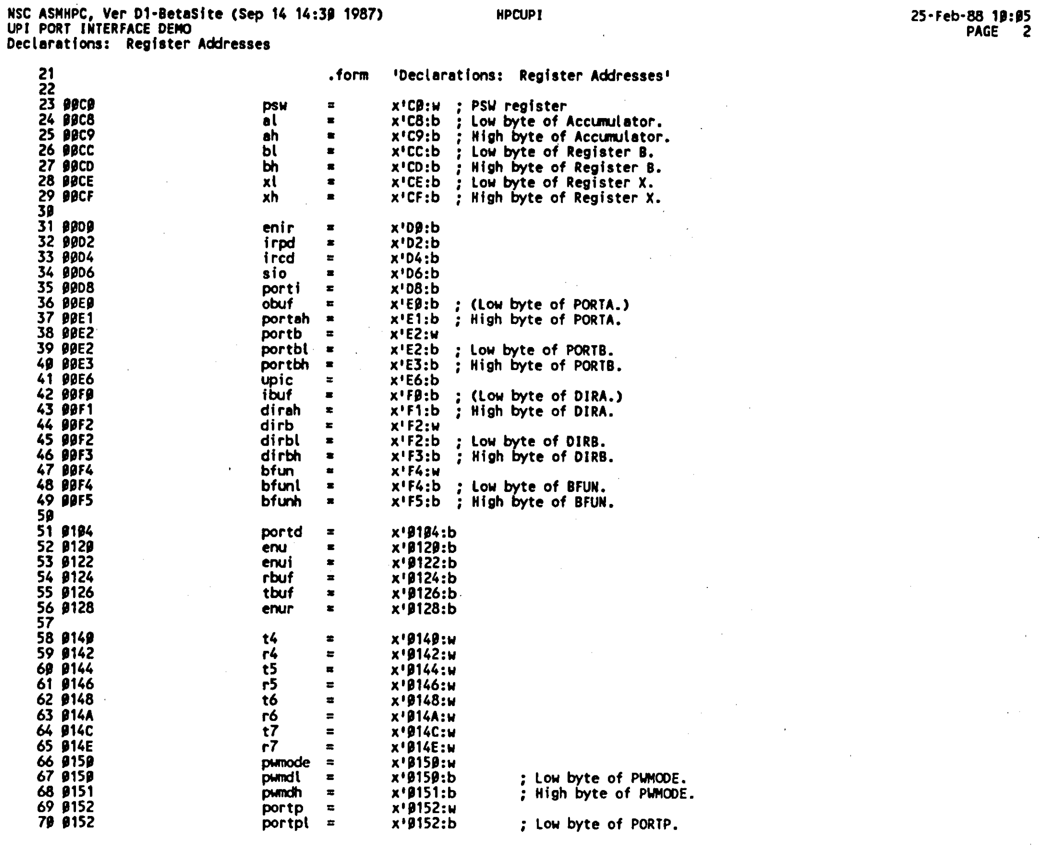
The interrupt service routine for Timer T0 (labels "tmrint", to "t0poll", to "t0int") is used simply to provide timing for the duration of the speaker tone. The interrupt is enabled in response to the BEEP command from the CPU, and is disabled on occurrence of the interrupt. It provides an interval of approximately one second.

4.2 HPC Firmware Listing



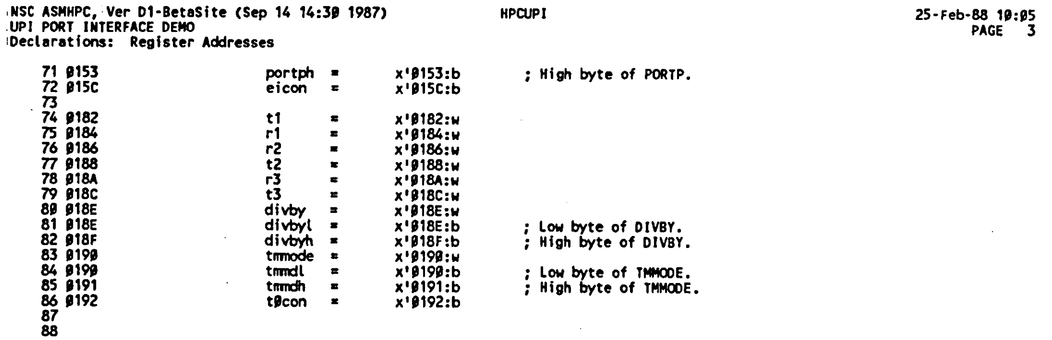
```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                              25-Feb-88 10:05
                                                                                                   PAGE    1


                              .title  HPCUPI,'UPI PORT INTERFACE DEMO'

                    ;         Demo program for HPC46083 UPI Port:
                    ;                 Demonstrates use of the HPC as an interface
                    ;                 between an NS32C016 CPU and some typical
                    ;                 front-panel types of devices:
                    ;                         LED indicators (up to 8)
                    ;                         Pushbuttons (up to 8)
                    ;                         LCD alphanumeric display controller (Hitachi HD44780)
                    ;                         Speaker for error beeps
                    ;                 Also generates Real-Time Clock interrupts at a
                    ;                     selectable rate.
                    ;
                    ;                 Generates !DIAG interrupt on errors;
                    ;                         severity code of NOTE (e.g. real-time event lost),
                    ;                                           or FATAL (e.g. bad command).
                    ;                 Recovery from fatal errors provided by RESET command.
```

TL/DD/9976-17



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:3Ø 1987)          HPCUPI                              25-Feb-88 1Ø:Ø5
UPI PORT INTERFACE DEMO                                                                           PAGE   2
Declarations:  Register Addresses

                                .form   'Declarations:  Register Addresses'

     ØØCØ           psw     =       x'CØ:w  ; PSW register
     ØØC8           al      =       x'C8:b  ; Low byte of Accumulator.
     ØØC9           ah      =       x'C9:b  ; High byte of Accumulator.
     ØØCC           bl      =       x'CC:b  ; Low byte of Register B.
     ØØCD           bh      =       x'CD:b  ; High byte of Register B.
     ØØCE           xl      =       x'CE:b  ; Low byte of Register X.
     ØØCF           xh      =       x'CF:b  ; High byte of Register X.

     ØØDØ           enir    =       x'DØ:b
     ØØD2           irpd    =       x'D2:b
     ØØD4           ircd    =       x'D4:b
     ØØD6           sio     =       x'D6:b
     ØØD8           porti   =       x'D8:b
     ØØEØ           obuf    =       x'EØ:b  ; (Low byte of PORTA.)
     ØØE1           portah  =       x'E1:b  ; High byte of PORTA.
     ØØE2           portb   =       x'E2:w
     ØØE2           portbl  =       x'E2:b  ; Low byte of PORTB.
     ØØE3           portbh  =       x'E3:b  ; High byte of PORTB.
     ØØE6           upic    =       x'E6:b
     ØØFØ           ibuf    =       x'FØ:b  ; (Low byte of DIRA.)
     ØØF1           dirah   =       x'F1:b  ; High byte of DIRA.
     ØØF2           dirb    =       x'F2:w
     ØØF2           dirbl   =       x'F2:b  ; Low byte of DIRB.
     ØØF3           dirbh   =       x'F3:b  ; High byte of DIRB.
     ØØF4           bfun    =       x'F4:w
     ØØF4           bfunl   =       x'F4:b  ; Low byte of BFUN.
     ØØF5           bfunh   =       x'F5:b  ; High byte of BFUN.

     Ø1Ø4           portd   =       x'Ø1Ø4:b
     Ø12Ø           enu     =       x'Ø12Ø:b
     Ø122           enui    =       x'Ø122:b
     Ø124           rbuf    =       x'Ø124:b
     Ø126           tbuf    =       x'Ø126:b
     Ø128           enur    =       x'Ø128:b

     Ø14Ø           t4      =       x'Ø14Ø:w
     Ø142           r4      =       x'Ø142:w
     Ø144           t5      =       x'Ø144:w
     Ø146           r5      =       x'Ø146:w
     Ø148           t6      =       x'Ø148:w
     Ø14A           r6      =       x'Ø14A:w
     Ø14C           t7      =       x'Ø14C:w
     Ø14E           r7      =       x'Ø14E:w
     Ø15Ø           pwmode  =       x'Ø15Ø:w
     Ø15Ø           pwmdl   =       x'Ø15Ø:b        ; Low byte of PWMODE.
     Ø151           pwmdh   =       x'Ø151:b        ; High byte of PWMODE.
     Ø152           portp   =       x'Ø152:w
     Ø152           portpl  =       x'Ø152:b        ; Low byte of PORTP.
```

TL/DD/9976-18



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:3Ø 1987)          HPCUPI                              25-Feb-88 1Ø:Ø5
UPI PORT INTERFACE DEMO                                                                           PAGE   3
Declarations:  Register Addresses

     Ø153           portph  =       x'Ø153:b        ; High byte of PORTP.
     Ø15C           eicon   =       x'Ø15C:b

     Ø182           t1      =       x'Ø182:w
     Ø184           r1      =       x'Ø184:w
     Ø186           r2      =       x'Ø186:w
     Ø188           t2      =       x'Ø188:w
     Ø18A           r3      =       x'Ø18A:w
     Ø18C           t3      =       x'Ø18C:w
     Ø18E           divby   =       x'Ø18E:w
     Ø18E           divbyl  =       x'Ø18E:b        ; Low byte of DIVBY.
     Ø18F           divbyh  =       x'Ø18F:b        ; High byte of DIVBY.
     Ø19Ø           tmmode  =       x'Ø19Ø:w
     Ø19Ø           tmmdl   =       x'Ø19Ø:b        ; Low byte of TMMODE.
     Ø191           tmmdh   =       x'Ø191:b        ; High byte of TMMODE.
     Ø192           tØcon   =       x'Ø192:b


```

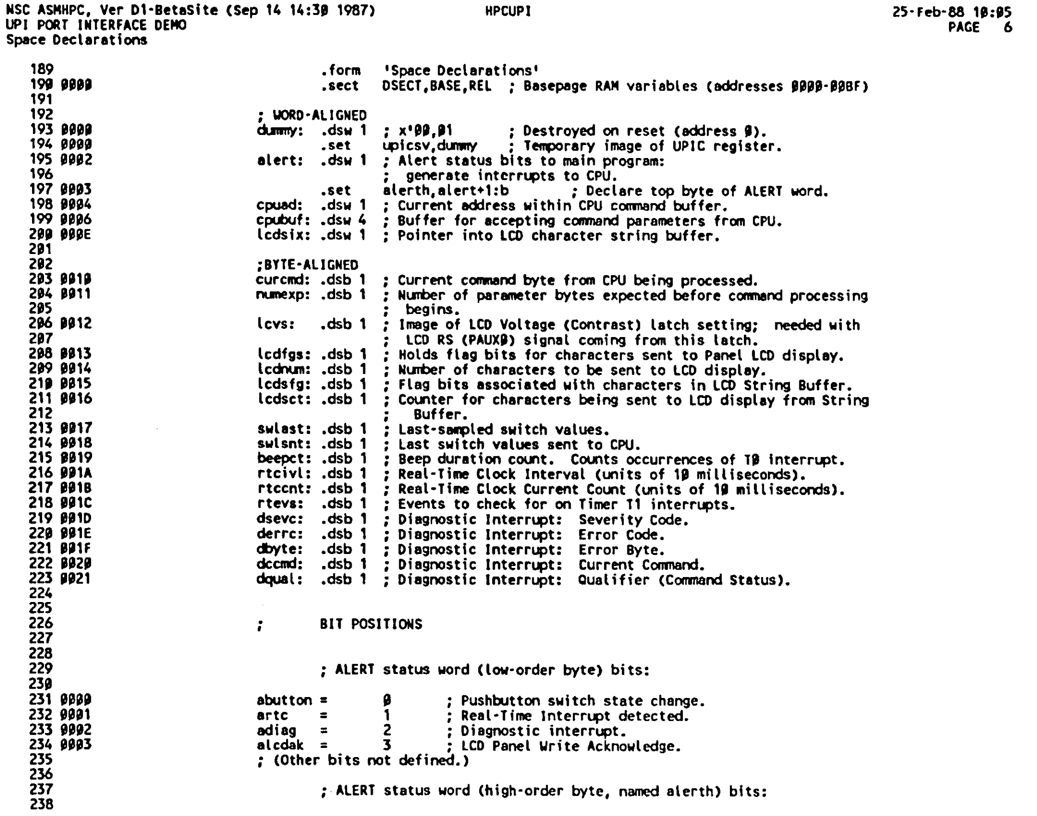
TL/DD/9976-19

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                      PAGE   4
Declarations:  Register Bit Positions

                            .form   'Declarations:  Register Bit Positions'

                     ; Name      Position      Register(s)
                     ; ----      --------      -----------
                     ;
     0000            gie    =       0       ; enir
     0002            i2     =       2       ; enir, irpd, ircd
     0003            i3     =       3       ; enir, irpd, ircd
     0004            i4     =       4       ; enir, irpd, ircd
     0005            tmrs   =       5       ; enir, irpd
     0006            uart   =       6       ; enir, irpd
     0007            ei     =       7       ; enir, irpd

     0001            uwmode =       1       ; ircd
     0000            uwdone =       0       ; irpd

     0000            tbmt   =       0       ; enu
     0001            rbfl   =       1       ; enu
     0004            b8or9  =       4       ; enu
     0005            xbit9  =       5       ; enu
     0002            wakeup =       2       ; enur
     0003            rbit9  =       3       ; enur
     0006            frmerr =       6       ; enur
     0007            doeerr =       7       ; enur
     0000            eti    =       0       ; enui
     0001            eri    =       1       ; enui
     0002            xtclk  =       2       ; enui
     0003            xrclk  =       3       ; enui
     0007            b2stp  =       7       ; enui

     0000            wrrdy  =       0       ; upic
     0001            rdrdy  =       1       ; upic
     0002            la0    =       2       ; upic
     0003            upien  =       3       ; upic
     0004            b8or16 =       4       ; upic

     0000            t0tie  =       0       ; tmmdl
     0001            t0pnd  =       1       ; tmmdl
     0003            t0ack  =       3       ; tmmdl
     0004            t1tie  =       4       ; tmmdl
     0005            t1pnd  =       5       ; tmmdl
     0006            t1stp  =       6       ; tmmdl
     0007            t1ack  =       7       ; tmmdl
     0000            t2tie  =       0       ; tmmdh
     0001            t2pnd  =       1       ; tmmdh
     0002            t2stp  =       2       ; tmmdh
     0003            t2ack  =       3       ; tmmdh
     0004            t3tie  =       4       ; tmmdh
     0005            t3pnd  =       5       ; tmmdh
     0006            t3stp  =       6       ; tmmdh
```

TL/DD/9976-20

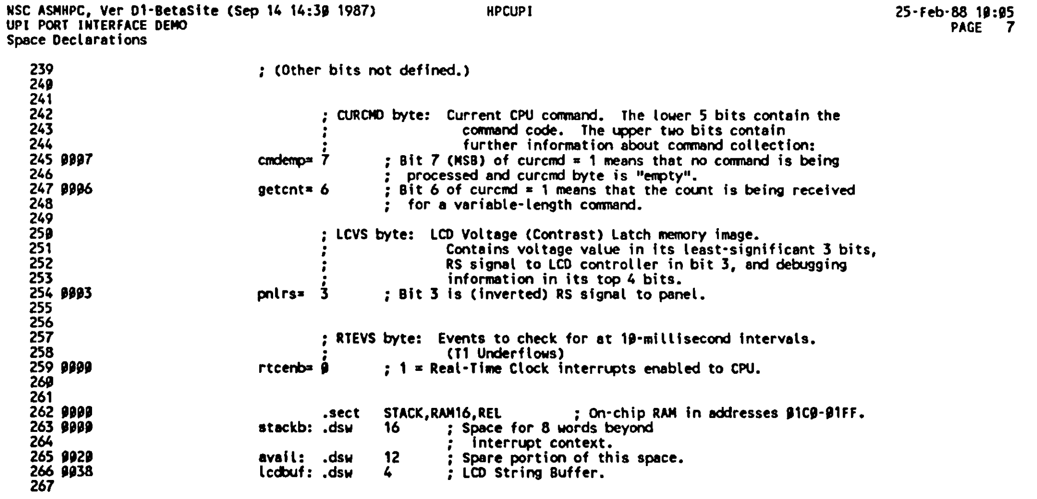
```
0007                t3ack    =      7       ; tmmdh

0000                t4tie    =      0       ; pwmdl
0001                t4pnd    =      1       ; pwmdl
0002                t4stp    =      2       ; pwmdl
0003                t4ack    =      3       ; pwmdl
0004                t5tie    =      4       ; pwmdl
0005                t5pnd    =      5       ; pwmdl
0006                t5stp    =      6       ; pwmdl
0007                t5ack    =      7       ; pwmdl
0000                t6tie    =      0       ; pwmdh
0001                t6pnd    =      1       ; pwmdh
0002                t6stp    =      2       ; pwmdh
0003                t6ack    =      3       ; pwmdh
0004                t7tie    =      4       ; pwmdh
0005                t7pnd    =      5       ; pwmdh
0006                t7stp    =      6       ; pwmdh
0007                t7ack    =      7       ; pwmdh

0000                t4out    =      0       ; portpl
0003                t4tfn    =      3       ; portpl
0004                t5out    =      4       ; portpl
0007                t5tfn    =      7       ; portpl
0000                t6out    =      0       ; portph
0003                t6tfn    =      3       ; portph
0004                t7out    =      4       ; portph
0007                t7tfn    =      7       ; portph

0000                eipol    =      0       ; eicon
0001                eimode   =      1       ; eicon
0002                eiack    =      2       ; eicon

0005                so       =      5       ; portbl, dirbl, bfunl
0006                sk       =      6       ; portbl, dirbl, bfunl
0007                pnlclk   =      7       ; portbl, dirbl

0001                lcvclk   =      1       ; portbh, dirbh
                    ; ua0 would be 2    , but requires no setup.
0003                uwrrdy   =      3       ; portbh, dirbh, bfunh
0004                cdata    =      4       ; portbh (enables non-pushbutton data to Port D).
0005                astts    =      5       ; portbh (enables pushbutton data to Port D).
0006                ledclk   =      6       ; portbh, dirbh
0007                urdrdy   =      7       ; portbh, dirbh, bfunh

                    ;       CONSTANTS
                    ;
0011                xon=    x'11    ; XON character:   Control-Q
0013                xoff=   x'13    ; XOFF character:  Control-S


```

TL/DD/9976-21



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)        HPCUPI                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                 PAGE   6
Space Declarations

                      .form   'Space Declarations'
0000                  .sect   DSECT,BASE,REL  ; Basepage RAM variables (addresses 0000-00BF)

            ; WORD-ALIGNED
0000        dummy:    .dsw 1  ; x'00,01        ; Destroyed on reset (address 0).
0000                  .set    upicsv,dummy     ; Temporary image of UPIC register.
0002        alert:    .dsw 1  ; Alert status bits to main program:
                              ;  generate interrupts to CPU.
0003                  .set    alerth,alert+1:b        ; Declare top byte of ALERT word.
0004        cpuad:    .dsw 1  ; Current address within CPU command buffer.
0006        cpubuf:   .dsw 4  ; Buffer for accepting command parameters from CPU.
000E        lcdsix:   .dsw 1  ; Pointer into LCD character string buffer.

            ;BYTE-ALIGNED
0010        curcmd:   .dsb 1  ; Current command byte from CPU being processed.
0011        numexp:   .dsb 1  ; Number of parameter bytes expected before command processing
                              ;  begins.
0012        lcvs:     .dsb 1  ; Image of LCD Voltage (Contrast) latch setting;  needed with
                              ;  LCD RS (PAUX0) signal coming from this latch.
0013        lcdfgs:   .dsb 1  ; Holds flag bits for characters sent to Panel LCD display.
0014        lcdnum:   .dsb 1  ; Number of characters to be sent to LCD display.
0015        lcdsfg:   .dsb 1  ; Flag bits associated with characters in LCD String Buffer.
0016        lcdsct:   .dsb 1  ; Counter for characters being sent to LCD display from String
                              ;   Buffer.
0017        swlast:   .dsb 1  ; Last-sampled switch values.
0018        swlsnt:   .dsb 1  ; Last switch values sent to CPU.
0019        beepct:   .dsb 1  ; Beep duration count.  Counts occurrences of T0 interrupt.
001A        rtcivl:   .dsb 1  ; Real-Time Clock Interval (units of 10 milliseconds).
001B        rtccnt:   .dsb 1  ; Real-Time Clock Current Count (units of 10 milliseconds).
001C        rtevs:    .dsb 1  ; Events to check for on Timer T1 interrupts.
001D        dsevc:    .dsb 1  ; Diagnostic Interrupt:  Severity Code.
001E        derrc:    .dsb 1  ; Diagnostic Interrupt:  Error Code.
001F        dbyte:    .dsb 1  ; Diagnostic Interrupt:  Error Byte.
0020        dccmd:    .dsb 1  ; Diagnostic Interrupt:  Current Command.
0021        dqual:    .dsb 1  ; Diagnostic Interrupt:  Qualifier (Command Status).


            ;         BIT POSITIONS


                      ; ALERT status word (low-order byte) bits:

0000        abutton =       0       ; Pushbutton switch state change.
0001        artc    =       1       ; Real-Time Interrupt detected.
0002        adiag   =       2       ; Diagnostic interrupt.
0003        alcdak  =       3       ; LCD Panel Write Acknowledge.
            ; (Other bits not defined.)

                      ; ALERT status word (high-order byte, named alerth) bits:

```

TL/DD/9976-22



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)        HPCUPI                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                 PAGE   7
Space Declarations

            ; (Other bits not defined.)


                      ; CURCMD byte:  Current CPU command.  The lower 5 bits contain the
                      ;                  command code.  The upper two bits contain
                      ;                  further information about command collection:
0007        cmdemp= 7       ; Bit 7 (MSB) of curcmd = 1 means that no command is being
                            ;  processed and curcmd byte is "empty".
0006        getcnt= 6       ; Bit 6 of curcmd = 1 means that the count is being received
                            ;  for a variable-length command.

                      ; LCVS byte:  LCD Voltage (Contrast) Latch memory image.
                      ;                Contains voltage value in its least-significant 3 bits,
                      ;                RS signal to LCD controller in bit 3, and debugging
                      ;                information in its top 4 bits.
0003        pnlrs=  3       ; Bit 3 is (inverted) RS signal to panel.


                      ; RTEVS byte:  Events to check for at 10-millisecond intervals.
                      ;                 (T1 Underflows)
0000        rtcenb= 0       ; 1 = Real-Time Clock interrupts enabled to CPU.


0000                  .sect   STACK,RAM16,REL         ; On-chip RAM in addresses 01C0-01FF.
0000        stackb: .dsw    16      ; Space for 8 words beyond
                                    ;  interrupt context.
0020        avail:  .dsw    12      ; Spare portion of this space.
0038        lcdbuf: .dsw    4       ; LCD String Buffer.

```

TL/DD/9976-23

```
                                  .form   'Code Section'
0000                              .sect   CSECT,ROM16,REL ; Code space.  (On-chip ROM)

                                  ; Declarations of subroutines called by one-byte JSRP instruction.

0000                              .spt    rdwait          ; Waits for CPU to read a value from UPI port.
0000                              .spt    wrpnl           ; Writes to LCD panel (for initialization only).

                                  ; Program starts at label "start" on reset.  This routine is the fatal
                                  ;  error handler, located here for convenience in setting breakpoint.

0000 96D018               hangup: rbit    gie,enir        ; Fatal error:  signal it and halt.
0003 96120F          R            sbit    7,lcvs          ; Signal error on most-significant bit of
                                                          ;  LCD Contrast Latch.
0006 96120B          R            sbit    pnlrs,lcvs      ; Select command mode for LCD controller.
0009 8C12E1          R            ld      portah,lcvs     ; Place error on Port A for latch.
000C 96E309                       sbit    lcvclk,portbh   ; Clock LCD Contrast Latch high,
000F 96E319                       rbit    lcvclk,portbh   ;  then low to load it.
0012 B601510A                     sbit    t6stp,pwmdh     ;
0016 B6015118                     rbit    t6tie,pwmdh     ; Set up Timer T6 for non-interrupt use.
001A 40                           nop
001B B6015119                     rbit    t6pnd,pwmdh     ; Clear Pending bit.
001F 3F00                         pop     0.w             ; Get error address from stack.
0021 B70000C4        R            ld      sp,#stackb      ; In case of stack underflow, re-initialize SP.
0025 9001                         ld      A,#x'01
0027 2E              R            jsrl    wrpnl           ; Clear LCD panel.
0028 96121B          R            rbit    pnlrs,lcvs      ; Set up panel for data.
002B 8C12E1          R            ld      portah,lcvs     ; Place error on Port A for latch.
002E 96E309                       sbit    lcvclk,portbh   ; Clock LCD Contrast Latch high,
0031 96E319                       rbit    lcvclk,portbh   ;  then low to load it.
0034 8801                         ld      A,1.b           ; Process first character of return address.
0036 3B                           swap    A
0037 990F                         and     A,#x'0F
0039 A6007AC888      R            ld      A,hextab[A].b
003E 2E              R            jsrl    wrpnl           ; Display it on LCD panel.
003F 8801                         ld      A,1.b           ; Process second character of return address.
0041 990F                         and     A,#x'0F
0043 A6007AC888      R            ld      A,hextab[A].b
0048 2E              R            jsrl    wrpnl           ; Display it on LCD panel.
0049 8800                         ld      A,0.b           ; Process third character of return address.
004B 3B                           swap    A
004C 990F                         and     A,#x'0F
004E A6007AC888      R            ld      A,hextab[A].b
0053 2E              R            jsrl    wrpnl           ; Display it on LCD panel.
0054 8800                         ld      A,0.b           ; Process last character of return address.
0056 990F                         and     A,#x'0F
0058 A6007AC888      R            ld      A,hextab[A].b
005D 2E              R            jsrl    wrpnl           ; Display it on LCD panel.

005E 96E611                hgupi: ifbit   rdrdy,upic      ; Check to see if OBUF register is full.
```

TL/DD/9976-24

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                    PAGE  9
Code Section

0061 971DE0                   ld      obuf,#vdiag     ; If not, fill it with IDIAG vector
                                                      ;  continuously.
0064 960213                   ifbit   i3,irpd         ; Check for UPI data ready.
0067 41                       jp      hgupi1
0068 6A                       jp      hgupi

0069 82A5F00C        hgupi1:  ifeq    ibuf,#x'A5      ; Check for RESET command.
006D 41                       jp      hgrst
006E 47                       jp      hgupi2
006F 96E612          hgrst:   ifbit   la0,upic
0072 43                       jp      hgupi2
0073 B4027A                   jmpl    xreset          ; If so, then go reset the HPC.


                                                      ; This is part of the outer loop, waiting for
                                                      ;  the RESET command.
0076 97F7D2          hgupi2:  ld      irpd,#x'F7      ; Clear the UWR detector,
0079 7B                       jp      hgupi           ;  and keep looking.  This is an
                                                      ;  infinite loop until RESET is seen.

007A 30              hextab:  .byte   '0','1','2','3','4','5','6','7'
007B 31
007C 32
007D 33
007E 34
007F 35
0080 36
0081 37
0082 38                       .byte   '8','9','A','B','C','D','E','F'
0083 39
0084 41
0085 42
0086 43
0087 44
0088 45
0089 46

```

TL/DD/9976-25

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                    PAGE 10
Hardware Initialization

                              .form   'Hardware Initialization'

008A 9708C0          start:   ld      psw.b,#x'08     ; Set one WAIT state.

008D                 srfsh:                   ; Start dynamic RAM refreshing,
                                              ;  as quickly as possible.
008D B6015208                 sbit    t4out,portpl    ; Trigger first refresh
                                                      ;  immediately.
0091 B601500A                 sbit    t4stp,pwmdl     ; Stop timer T4 to
                                                      ;  allow loading,
0095 83080140AB               ld      t4.w,#8         ;  then load it.
009A B601501A                 rbit    t4stp,pwmdl     ; Start timer T4.
009E B601520B                 sbit    t4tfn,portpl    ; Enable pulses out.
00A2 83080142AB               ld      r4.w,#8         ; Load R4.

00A7                 supi:                            ; Set up UPI port.
00A7 9718E6                   ld      upic,#x'18      ; 8-Bit UPI Mode
                                                      ;  enabled.

00AA 96F50B                   sbit    uwrrdy,bfunh    ; Enable UWRRDY/ out.
00AD 96F30B                   sbit    uwrrdy,dirbh
00B0 88F0                     ld      A,ibuf          ; Empty IBUF register,
                                                      ;  in case of false trigger.

00B2 96F50F                   sbit    urdrdy,bfunh    ; Enable URDRDY/ out.
00B5 96F30F                   sbit    urdrdy,dirbh

                                                      ; Set up UREAD/ interrupt.
00B8 96D40A                   sbit    i2,ircd         ; Detects rising edges.
00BB 97FBD2                   ld      irpd,#x'FB      ; Clear any false interrupt
                                                      ;  due to mode change.

                                                      ; Set up UWRITE/ interrupt.
00BE 96D40B                   sbit    i3,ircd         ; Detects rising edges.
00C1 97F7D2                   ld      irpd,#x'F7      ; Clear any false interrupt
                                                      ;  due to mode change.

00C4                 sram:                    ; Clear all RAM locations.
                                              ; Clear Basepage bank:
00C4 8D008E                   ld      BK,#x'0000,#x'00BE      ; Establish loop base and limit.
00C7 00              sraml1:  clr     A
00C8 E1                       xs      A,[B+].w
00C9 62                       jp      sraml1

                                              ; Clear Non-Basepage bank:
00CA A701C001FE               ld      BK,#x'01C0,#x'01FE      ; Establish loop base and limit.
00CF 00              sraml2:  clr     A
00D0 E1                       xs      A,[B+].w
00D1 62                       jp      sraml2

```

TL/DD/9976-26

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                                  25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                 PAGE  11
Hardware Initialization

 00D2                        sskint:                    ; Set up Stack and remove
                                                        ;  individual interrupt enables.
 00D2 B70002C4         R             ld     sp,#stackb+2    ; Move stack to high
                                                            ;  bank of on-chip RAM.
 00D6 8700000000AB     R             ld     stackb.w,#hangup  ; Safeguard against
                                                              ;  stack underflow.
 00DC 970000                         ld     enir,#x'00      ; Disable interrupts
                                                            ;  individually.

 00DF 830801928B             tminit: ld     t0con,#x'08
 00E4 874440019 0AB                  ld     tmmode,#x'4440   ; Stop timers T1, T2, T3.
 00EA 8355018EAB                     ld     divby,#x'0055    ; Timers T2 and T3 set to
                                                             ;  clock externally.
 00EF 87CCC80190AB                   ld     tmmode,#x'CCC8   ; Clear and disable timer
                                                             ;  T0-T3 interrupts.

 00F5 97FFF1                         ld     dirah,#x'FF     ; Initialize Port A upper byte for output.
 00F8 96E30D                         sbit   astts,portbh    ; Enable and de-assert ENASTTS/ signal
 00FB 96F30D                         sbit   astts,dirbh     ;  (enables pushbutton data to Port D).
 00FE 96E30C                         sbit   cdata,portbh    ; Enable and de-assert ENCDATA/ signal.
 0101 96F30C                         sbit   cdata,dirbh     ;  (enables other data to Port D).

 0104 97FFE1                 sled:   ld     portah,#x'FF    ; Set up to turn off LED's.
 0107 96E31E                         rbit   ledclk,portbh   ; Start with LEDCLK low,
 010A 96F30E                         sbit   ledclk,dirbh    ;  (enable output),
 010D 96E30E                         sbit   ledclk,portbh   ;  then high,
 0110 96E31E                         rbit   ledclk,portbh   ;  then low again.

 0113                        stmrs:                         ; Set up remaining timers.
                                                            ;  (T1-T3 already stopped
                                                            ;   and pending bits cleared
                                                            ;   at tminit above, as
                                                            ;   part of MICROWIRE init.)

 0113 872FFF0182AB                   ld     t1,#12287       ; T1 runs at 10-millisecond real-time interval.
 0119 872FFF0184AB                   ld     r1,#12287
                                                            ; Timer remains stopped, and interrupt
                                                            ;  disabled, until INITIALIZE command.

 011F 87444001500AB                  ld     pwmode,#x'4440    ; Stop timers T4-T7.
 0125 40                             nop                      ; Wait for valid PND
 0126 40                             nop                      ;  bits.
 0127 87CCC80150AB                   ld     pwmode,#x'CCC8    ; Clear and disable
                                                              ;  interrupts from all
                                                              ;  PWM timers.

 012D 87FFFF014AAB                   ld     r6,#x'FFFF      ; No modulus for LCD Display Ready timer.

 0133 83CC014CAB                     ld     t7,#204         ; Set T7 to underflow at 6 KHz rate
 0138 83CC014EAB                     ld     r7,#204         ;  (= 3 KHz at pin).
```

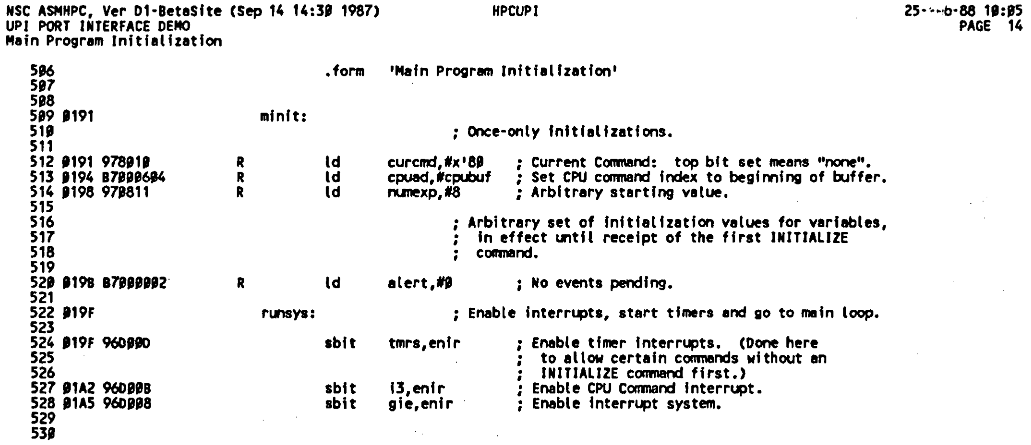
TL/DD/9976-27

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                                  25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                 PAGE  12
Hardware Initialization

0130 B601531F                  rbit    t7tfn,portph    ; Disable beep tone to panel speaker.
0141 B601511E                  rbit    t7stp,pwmdh     ; Start T7 running.


0145                    slcd:                   ; Set up LCD display.
                                                ; Requires use of timer T6, so
                                                ;  appears after timer initialization.

                                                ; First, set up LCD contrast.
0145 970A12          R         ld      lcvs,#x'0A      ; Initialize memory image of LCD Voltage
                                                       ;  latch, containing RS (PAUX0) bit also.
0148 8C12E1          R         ld      portah,lcvs     ; Arbitrary initial contrast level of 5,
                                                       ;  and RS/ (PAUX0/) is high (="command").
014B 96E319                    rbit    lcvclk,portbh   ; Start with LCVCLK low,
014E 96F309                    sbit    lcvclk,dirbh    ;  (enable output)
0151 96E309                    sbit    lcvclk,portbh   ;  then high,
0154 96E319                    rbit    lcvclk,portbh   ;  then low to get it into LCV latch.

                                                ; Initialize PNLCLK (Panel "E" signal).
0157 96E20F                    sbit    pnlclk,portbl   ; Start with PNLCLK high
015A 96F20F                    sbit    pnlclk,dirbl    ;  (enable output).

                                                ; Wait for worst-case command
                                                ;  execution time (4.9 ms, twice), in case
                                                ;  a panel command was triggered while
                                                ;  PNLCLK was floating.
015D B601510B                  sbit    t6ack,pwmdh     ; Clear T6 PND bit.
0161 8732C80148AB              ld      t6,#13000       ; Set T6 to twice 4.9 milliseconds.
0167 B601511A                  rbit    t6stp,pwmdh     ; Start timer T6.
016B B6015111           lcdlp1: ifbit  t6pnd,pwmdh     ; Wait for T6 PND bit
                                                       ;  to be set.
016F 41                        jp      lcdgo1
0170 65                        jp      lcdlp1
0171 B601510A           lcdgo1: sbit   t6stp,pwmdh     ; Stop timer T6.
0175 B601510B                  sbit    t6ack,pwmdh     ; Clear T6 PND bit.

                                                ; Reset Panel controller (per Hitachi HD44780
                                                ;   User's Manual).

                                                ; (Panel RS signal was set
                                                ;  in LCD Contrast initialization above,
                                                ;  so no change needed here to
                                                ;  flag these as commands.)

0179 9038                      ld      A,#x'38         ; Send "8-Bit Mode, 2 Lines" command:  one;
017B 2E              R         jsrl    wrpnl
017C 9038                      ld      A,#x'38         ; two;
017E 2E              R         jsrl    wrpnl
017F 9038                      ld      A,#x'38         ; three;
0181 2E              R         jsrl    wrpnl
```

TL/DD/9976-28

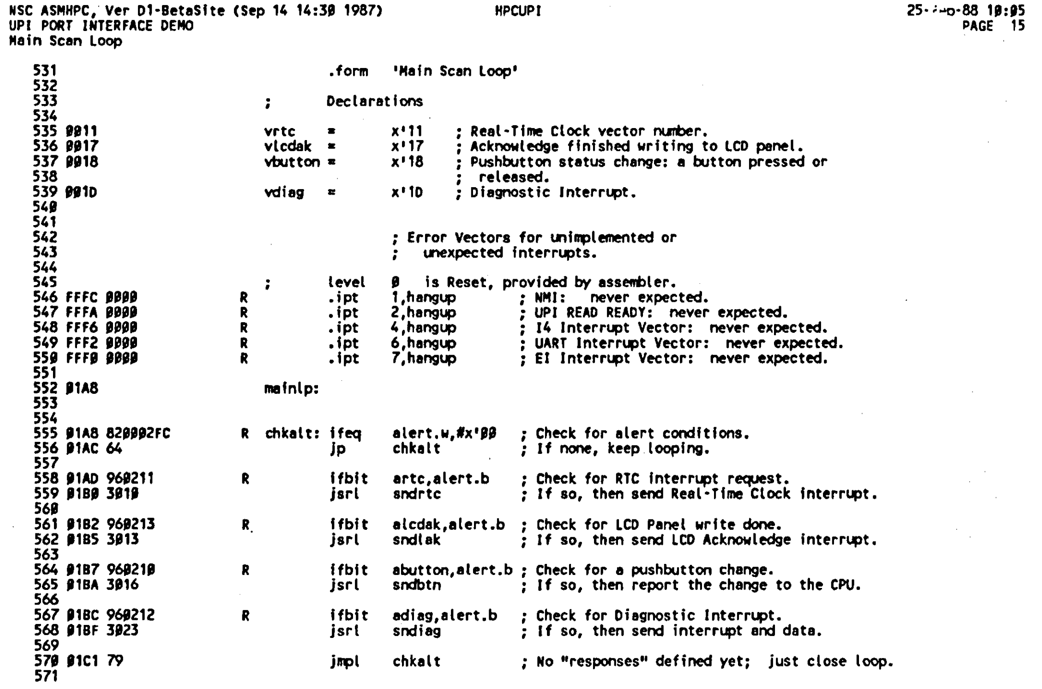
```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                                  25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                 PAGE  13
Hardware Initialization

0182 9038                      ld      A,#x'38         ; four times.
0184 2E              R         jsrl    wrpnl
0185 9008                      ld      A,#x'08         ; Disable display.
0187 2E              R         jsrl    wrpnl
0188 9001                      ld      A,#x'01         ; Clear display RAM.
018A 2E              R         jsrl    wrpnl

                                                ; Initial default mode settings.

018B 9006                      ld      A,#x'06         ; Set mode to move cursor to the right, no
018D 2E              R         jsrl    wrpnl           ;   automatic shifting of display.
018E 900E                      ld      A,#x'0E         ; Enable display:  non-blinking cursor mode.
0190 2E              R         jsrl    wrpnl



                        ;      CONTINUES TO MAIN PROGRAM INITIALIZATION
```

TL/DD/9976-29



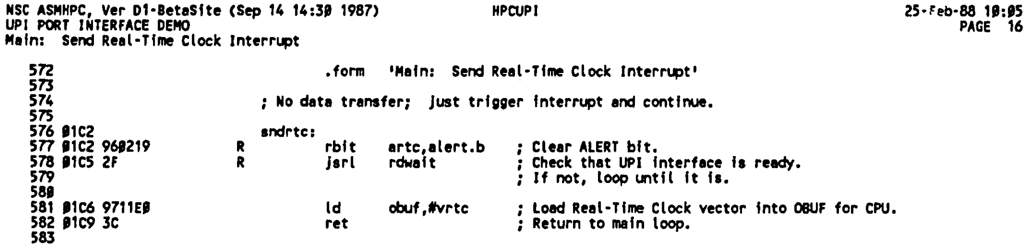
```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                                   25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                   PAGE  14
Main Program Initialization

  506                           .form   'Main Program Initialization'
  507
  508
  509 0191                minit:
  510                                             ; Once-only initializations.
  511
  512 0191 978010         R          ld      curcmd,#x'80    ; Current Command:  top bit set means "none".
  513 0194 B7000604       R          ld      cpuad,#cpubuf   ; Set CPU command index to beginning of buffer.
  514 0198 970811         R          ld      numexp,#8       ; Arbitrary starting value.
  515
  516                                             ; Arbitrary set of initialization values for variables,
  517                                             ;  in effect until receipt of the first INITIALIZE
  518                                             ;  command.
  519
  520 019B B7000002       R          ld      alert,#0        ; No events pending.
  521
  522 019F                runsys:                 ; Enable interrupts, start timers and go to main loop.
  523
  524 019F 960000                    sbit    tmrs,enir       ; Enable timer interrupts.  (Done here
  525                                                         ;  to allow certain commands without an
  526                                                         ;  INITIALIZE command first.)
  527 01A2 960008                    sbit    i3,enir         ; Enable CPU Command interrupt.
  528 01A5 960008                    sbit    gie,enir        ; Enable interrupt system.
  529
  530
```

TL/DD/9976-30



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                                   25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                   PAGE  15
Main Scan Loop

  531                           .form   'Main Scan Loop'
  532
  533                     ;     Declarations
  534
  535 0011                vrtc    =     x'11    ; Real-Time Clock vector number.
  536 0017                vlcdak  =     x'17    ; Acknowledge finished writing to LCD panel.
  537 0018                vbutton =     x'18    ; Pushbutton status change: a button pressed or
  538                                             ;  released.
  539 001D                vdiag   =     x'1D    ; Diagnostic Interrupt.
  540
  541
  542                                       ; Error Vectors for unimplemented or
  543                                       ;   unexpected interrupts.
  544
  545                     ;     level     0   is Reset, provided by assembler.
  546 FFFC 0000           R     .ipt      1,hangup        ; NMI:   never expected.
  547 FFFA 0000           R     .ipt      2,hangup        ; UPI READ READY:  never expected.
  548 FFF6 0000           R     .ipt      4,hangup        ; I4 Interrupt Vector:  never expected.
  549 FFF2 0000           R     .ipt      6,hangup        ; UART Interrupt Vector:  never expected.
  550 FFF0 0000           R     .ipt      7,hangup        ; EI Interrupt Vector:  never expected.
  551
  552 01A8                mainlp:
  553
  554
  555 01A8 820002FC       R  chkalt: ifeq   alert.w,#x'00   ; Check for alert conditions.
  556 01AC 64                        jp     chkalt          ; If none, keep looping.
  557
  558 01AD 960211         R          ifbit  artc,alert.b    ; Check for RTC interrupt request.
  559 01B0 3010                      jsrl   sndrtc          ; If so, then send Real-Time Clock interrupt.
  560
  561 01B2 960213         R          ifbit  alcdak,alert.b  ; Check for LCD Panel write done.
  562 01B5 3013                      jsrl   sndlak          ; If so, then send LCD Acknowledge interrupt.
  563
  564 01B7 960210         R          ifbit  abutton,alert.b ; Check for a pushbutton change.
  565 01BA 3016                      jsrl   sndbtn          ; If so, then report the change to the CPU.
  566
  567 01BC 960212         R          ifbit  adiag,alert.b   ; Check for Diagnostic Interrupt.
  568 01BF 3023                      jsrl   sndiag          ; If so, then send interrupt and data.
  569
  570 01C1 79                        jmpl   chkalt          ; No "responses" defined yet;  just close loop.
  571
```

TL/DD/9976-31



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                                   25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                   PAGE  16
Main:  Send Real-Time Clock Interrupt

  572                           .form   'Main:  Send Real-Time Clock Interrupt'
  573
  574                     ; No data transfer;  just trigger interrupt and continue.
  575
  576 01C2                sndrtc:
  577 01C2 960219         R          rbit    artc,alert.b    ; Clear ALERT bit.
  578 01C5 2F             R          jsrl    rdwait          ; Check that UPI interface is ready.
  579                                                        ; If not, loop until it is.
  580
  581 01C6 9711E0                    ld      obuf,#vrtc      ; Load Real-Time Clock vector into OBUF for CPU.
  582 01C9 3C                        ret                     ; Return to main loop.
  583
```

TL/DD/9976-32

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)           HPCUPI                                25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                 PAGE  17
Main:  Send LCD Write Acknowledge Interrupt

                                      .form   'Main:  Send LCD Write Acknowledge Interrupt'

                              ; No data transfer;  just trigger interrupt and continue.

     01CA                     sndlak:
     01CA 96021B        R             rbit    alcdak,alert.b  ; Clear ALERT bit.
     01CD 2F            R             jsrl    rdwait          ; Check that UPI interface is ready.
                                                              ; If not, loop until it is.

     01CE 9717E0                      ld      obuf,#vlcdak    ; Load LCD-Acknowledge vector into OBUF for CPU.
     01D1 3C                          ret                     ; Return to main loop.

```

TL/DD/9976-33

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)           HPCUPI                                25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                 PAGE  18
Main:  Send Pushbutton Status to CPU

                                      .form   'Main:  Send Pushbutton Status to CPU'

     01D2                     sndbtn:
     01D2 2F            R             jsrl    rdwait          ; Check that UPI interface is ready.
                                                              ; If not, loop until it is.

     01D3 9718E0                      ld      obuf,#vbutton   ; Load BUTTON-DATA vector into OBUF for CPU.

     01D6 2F            R             jsrl    rdwait          ; Check that UPI interface is ready.
                                                              ; If not, loop until it is.

     01D7 960018                      rbit    gie,enir        ; *** Begin Indivisible Sequence ***
     01DA 8C18E0        R             ld      obuf,swlsnt     ; Load Pushbutton Data Byte into OBUF for CPU.
     01DD 960218        R             rbit    abutton,alert.b ; Clear ALERT bit.
     01E0 960008                      sbit    gie,enir        ; *** End Indivisible Sequence ***
     01E3 3C                          ret                     ; Return to main loop.

```

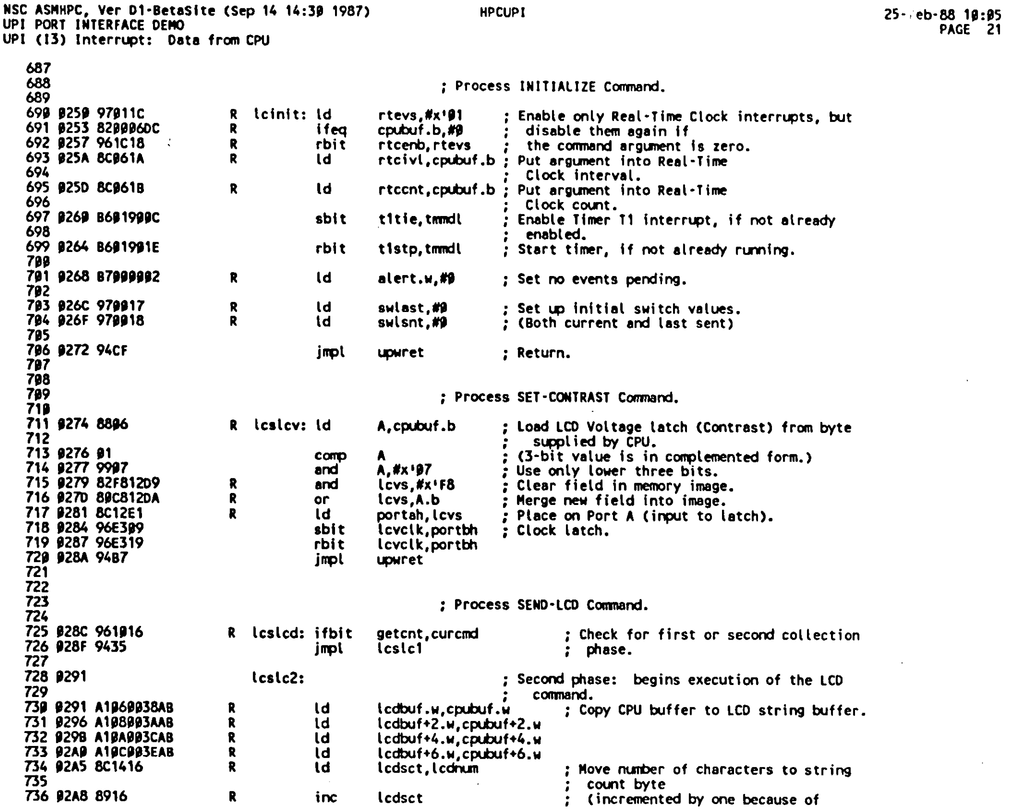
TL/DD/9976-34

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:3Ø 1987)          HPCUPI                                    25-Feb-88 1Ø:Ø5
UPI PORT INTERFACE DEMO                                                                                      PAGE  19
Main:  Send Diagnostic Interrupt to CPU

                                        .form   'Main:  Send Diagnostic Interrupt to CPU'

     Ø1E4                       sndiag:
     Ø1E4 2F            R               jsrl    rdwait           ; Wait for UPI interface ready.
     Ø1E5 971DEØ                        ld      obuf,#vdiag      ; Load vector into OBUF for CPU.
     Ø1E8 2F            R               jsrl    rdwait           ; Wait for UPI interface ready.
     Ø1E9 96DØ18                        rbit    gie,enir         ; *** Begin Indivisible Sequence ***
     Ø1EC 8C1DEØ        R               ld      obuf,dsevc       ; Transfer Severity Code.
     Ø1EF 97ØØ1D        R               ld      dsevc,#Ø         ; Clear it.
     Ø1F2 881E          R               ld      A,derrc          ; Get Error Code.
     Ø1F4 97ØØ1E        R               ld      derrc,#Ø         ; Clear it.
     Ø1F7 96Ø21A        R               rbit    adiag,alert.b    ; Clear ALERT bit.
     Ø1FA 96DØØ8                        sbit    gie,enir         ; *** End Indivisible Sequence ***
     Ø1FD 2F            R               jsrl    rdwait           ; Wait for UPI interface ready.
     Ø1FE 88EØ                          st      A,obuf           ; Transfer Error Code.
     Ø2ØØ 2F            R               jsrl    rdwait           ; Wait for UPI interface ready.
                                                         ; Remaining bytes will have meaning only for
                                                         ;   command errors.
     Ø2Ø1 8C1FEØ        R               ld      obuf,dbyte       ; Transfer Byte Received.
     Ø2Ø4 2F            R               jsrl    rdwait           ; Wait for UPI interface ready.
     Ø2Ø5 8C2ØEØ        R               ld      obuf,dccmd       ; Transfer Current Command.
     Ø2Ø8 2F            R               jsrl    rdwait           ; Wait for UPI interface ready.
     Ø2Ø9 8C21EØ        R               ld      obuf,dqual       ; Transfer Command Count.
     Ø2ØC 3C                            ret                      ; Return to main program loop.

```

TL/DD/9976-35

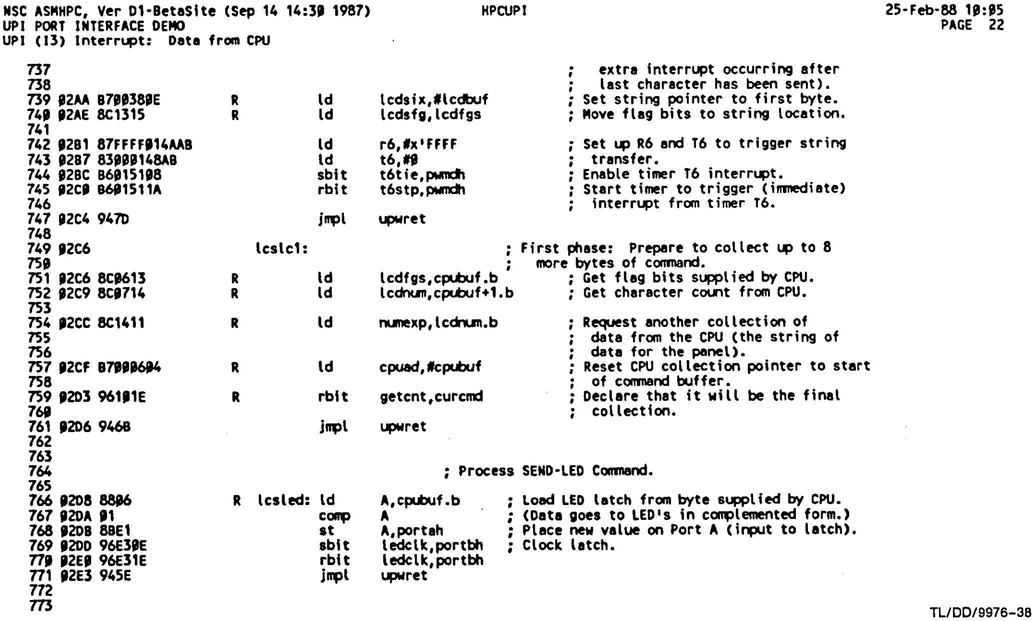
```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:3Ø 1987)          HPCUPI                                    25-Feb-88 1Ø:Ø5
UPI PORT INTERFACE DEMO                                                                                      PAGE  2Ø
UPI (I3) Interrupt:  Data from CPU

                                        .form   'UPI (I3) Interrupt:  Data from CPU'

     FFF8 ØDØ2          R               .ipt    3,upiwr          ; Declare upiwr as vector for Interrupt 3.

     Ø2ØD                       upiwr:                           ; Write Strobe received from CPU.
     Ø2ØD AFC8                          push    A                ; Save Context
     Ø2ØF AFCØ                          push    psw

     Ø211 8CE6ØØ        R               ld      upicsv.b,upic    ; Save UPIC register image for LAØ bit test.

     Ø214 961Ø17        R               ifbit   cmdemp,curcmd    ; If expecting first byte of a command,
     Ø217 94CC                          jmpl    firstc           ;  then go process it as such.

     Ø219 88FØ                          ld      A,ibuf           ; If not, input it for entry into cpubuf.

     Ø21B 9CA5                          ifeq    A,#x'A5          ; Check for RESET command.
     Ø21D 46                            jp      lcrst
     Ø21E 96ØØ12        R               ifbit   laØ,upicsv.b     ; Check for command argument written to proper
                                                                 ;  address.
     Ø221 48                            jp      lcord            ; If so, go process as a normal argument.
     Ø222 3622                          jsrl    hangup           ; If not, process as a FATAL error, generating
                                                                 ;  IDIAG interrupt.

     Ø224 96E612                lcrst:  ifbit   laØ,upic         ; Continue checking for a RESET command.
     Ø227 42                            jp      lcord
     Ø228 94C6                          jmpl    xreset           ; If so, go reset the HPC.

     Ø22A ADØ48E        R       lcord:  x       A,[cpuad].b      ; If not, place it in next available cpubuf
                                                                 ;  entry.
     Ø22D A9Ø4          R               inc     cpuad
     Ø22F 8A11          R               decsz   numexp
     Ø231 B4Ø1ØF                        jmpl    upwret           ; If not final byte of command, then return.

     Ø234 881Ø          R       lastc:  ld      A,curcmd         ; Else, process current command.
     Ø236 96C816                        ifbit   getcnt,A.b       ; Check if extended collection is being made.
     Ø239 47                            jp      lastc1           ; If not, then:
     Ø23A 961ØØF        R               sbit    cmdemp,curcmd    ;   Set command slot available again.
     Ø23D B7ØØØ6Ø4      R               ld      cpuad,#cpubuf    ;   Reset CPU buffer pointer to beginning.

     Ø241 991F                  lastc1: and     A,#x'1F          ; Mask off flag bits.
     Ø243 E7                            shl     A                ; Scale by two, and then
     Ø244 4Ø
     Ø245                               .odd
     Ø245 EC                            jidw                     ;  jump based on command value:

     Ø246 ØAØØ                  lastab: .ptw    lcinit           ; Ø = INITIALIZE command.
     Ø248 2CØØ                          .ptw    lcslcv           ; 1 = SET-CONTRAST command.
     Ø24A 42ØØ                          .ptw    lcslcd           ; 2 = SEND-LCD command.
     Ø24C 8CØØ                          .ptw    lcsled           ; 3 = SEND-LED command.
     Ø24E F3ØØ                          .ptw    illc             ; (BEEP command has only one byte.  Error.)
```

TL/DD/9976-36



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)        HPCUPI                                        25-‹eb-88 10:05
UPI PORT INTERFACE DEMO                                                                                       PAGE  21
UPI (I3) Interrupt:  Data from CPU


                                                 ; Process INITIALIZE Command.

0250 97011C          R  lcinit: ld      rtevs,#x'01      ; Enable only Real-Time Clock interrupts, but
0253 820006DC        R          ifeq    cpubuf.b,#0      ;  disable them again if
0257 961C18          R          rbit    rtcenb,rtevs     ;  the command argument is zero.
025A 8C061A          R          ld      rtcivl,cpubuf.b ; Put argument into Real-Time
                                                         ;  Clock interval.
025D 8C061B          R          ld      rtccnt,cpubuf.b ; Put argument into Real-Time
                                                         ;  Clock count.
0260 B601900C                   sbit    t1tie,tmmdl      ; Enable Timer T1 interrupt, if not already
                                                         ;  enabled.
0264 B601901E                   rbit    t1stp,tmmdl      ; Start timer, if not already running.

0268 B7000002        R          ld      alert.w,#0       ; Set no events pending.

026C 970017          R          ld      swlast,#0        ; Set up initial switch values.
026F 970018          R          ld      swlsnt,#0        ; (Both current and last sent)

0272 94CF                       jmpl    upwret           ; Return.



                                                 ; Process SET-CONTRAST Command.

0274 8806            R  lcslcv: ld      A,cpubuf.b       ; Load LCD Voltage latch (Contrast) from byte
                                                         ;   supplied by CPU.
0276 01                         comp    A                ; (3-bit value is in complemented form.)
0277 9907                       and     A,#x'07          ; Use only lower three bits.
0279 82F812D9        R          and     lcvs,#x'F8       ; Clear field in memory image.
027D 80C812DA        R          or      lcvs,A.b         ; Merge new field into image.
0281 8C12E1          R          ld      portah,lcvs      ; Place on Port A (input to latch).
0284 96E309                     sbit    lcvclk,portbh    ; Clock latch.
0287 96E319                     rbit    lcvclk,portbh
028A 94B7                       jmpl    upwret



                                                 ; Process SEND-LCD Command.

028C 961016          R  lcslcd: ifbit   getcnt,curcmd             ; Check for first or second collection
028F 9435                       jmpl    lcslc1                    ;  phase.

0291                    lcslc2:                          ; Second phase:  begins execution of the LCD
                                                         ;   command.
0291 A1060038AB      R          ld      lcdbuf.w,cpubuf.w         ; Copy CPU buffer to LCD string buffer.
0296 A108003AAB      R          ld      lcdbuf+2.w,cpubuf+2.w
029B A10A003CAB      R          ld      lcdbuf+4.w,cpubuf+4.w
02A0 A10C003EAB      R          ld      lcdbuf+6.w,cpubuf+6.w
02A5 8C1416          R          ld      lcdsct,lcdnum             ; Move number of characters to string
                                                                  ;  count byte
02A8 8916            R          inc     lcdsct                    ;  (incremented by one because of
```

TL/DD/9976-37



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)        HPCUPI                                        25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                       PAGE  22
UPI (I3) Interrupt:  Data from CPU

                                                                  ;   extra interrupt occurring after
                                                                  ;   last character has been sent).
02AA B700380E        R          ld      lcdsix,#lcdbuf            ; Set string pointer to first byte.
02AE 8C1315          R          ld      lcdsfg,lcdfgs             ; Move flag bits to string location.

02B1 87FFFF014AAB               ld      r6,#x'FFFF                ; Set up R6 and T6 to trigger string
02B7 8300014BAB                 ld      t6,#0                     ;  transfer.
02BC B6015108                   sbit    t6tie,pwmdh               ; Enable timer T6 interrupt.
02C0 B601511A                   rbit    t6stp,pwmdh               ; Start timer to trigger (immediate)
                                                                  ;  interrupt from timer T6.
02C4 947D                       jmpl    upwret

02C6                    lcslc1:                          ; First phase:  Prepare to collect up to 8
                                                         ;   more bytes of command.
02C6 8C0613          R          ld      lcdfgs,cpubuf.b           ; Get flag bits supplied by CPU.
02C9 8C0714          R          ld      lcdnum,cpubuf+1.b         ; Get character count from CPU.

02CC 8C1411          R          ld      numexp,lcdnum.b           ; Request another collection of
                                                                  ;  data from the CPU (the string of
                                                                  ;  data for the panel).
02CF B7000604        R          ld      cpuad,#cpubuf             ; Reset CPU collection pointer to start
                                                                  ;  of command buffer.
02D3 96101E          R          rbit    getcnt,curcmd             ; Declare that it will be the final
                                                                  ;  collection.
02D6 946B                       jmpl    upwret



                                                 ; Process SEND-LED Command.

02D8 8806            R  lcsled: ld      A,cpubuf.b       ; Load LED latch from byte supplied by CPU.
02DA 01                         comp    A                ; (Data goes to LED's in complemented form.)
02DB 88E1                       st      A,portah         ; Place new value on Port A (input to latch).
02DD 96E30E                     sbit    ledclk,portbh    ; Clock latch.
02E0 96E31E                     rbit    ledclk,portbh
02E3 945E                       jmpl    upwret


```

TL/DD/9976-38

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)        HPCUPI                              25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                              PAGE  23
Processing of First Byte of Command (Code)

                                     .form   'Processing of First Byte of Command (Code)'

                                             ;  One-byte commands are processed in this section.
                                             ;  Longer commands are scheduled for collection of
                                             ;    remaining bytes, and are processed in routines
                                             ;    above.

 02E5 88F0                firstc: ld     A,ibuf          ; Get command from UPI port.
 02E7 960012          R           ifbit  la0,upicsv.b    ; Check for out-of-sequence condition
                                                         ;  (argument instead of command).
 02EA 36EA                        jsrl   hangup          ; If so, process as a FATAL error (previous
                                                         ;  command was too short).

                                                   ; Processing of RESET command.

 02EC 9CA5                        ifeq   A,#x'A5         ; Check for RESET command.
 02EE 41                          jp     xreset
 02EF 59                          jp     fcord

                                                         ; This code is entered whenever a RESET
                                                         ;  command is received.
 02F0                     xreset:
 02F0 971DE0                      ld     obuf,#vdiag     ; Present dummy value for CPU,
                                                         ;  (in case a value was already in OBUF),
 02F3 2F              R           jsrl   rdwait          ;  and wait for it to be read by CPU.
 02F4 9000                        ld     A,#0            ; Initialize registers.
 02F6 8BE6                        st     A,upic
 02F8 ABF0                        st     A,ibuf.w        ; (Actually all of DIRA.)
 02FA ABF2                        st     A,dirb
 02FC ABF4                        st     A,bfun
 02FE 8BD4                        st     A,ircd
 0300 B60152AB                    st     A,portp
 0304 ABC4                        st     A,sp            ; Then, through RESET vector,
 0306 ABC0                        st     A,psw
 0308 3C                          ret                    ;  jump to start of program.

                                                   ; Here, process an ordinary command (not RESET).

 0309                     fcord:
 0309 991F                        and    A,#x'1F         ; Use only least-significant 5 bits.
 030B 9011                        ifgt   A,#x'11         ; Check for command out of range.
 030D 9432                        jmpl   illc
 030F 8B10            R           st     A,curcmd        ; Save as current command.

 0311 E7                          shl    A               ; Scale by two, and then
 0312 40
 0313                             .odd
 0313 EC                          jidw                   ;  jump based on command value:

 0314 0A00                firstab: .ptw  fcinit          ; 0 = INITIALIZE command.
```

TL/DD/9976-39

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)        HPCUPI                              25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                              PAGE  24
Processing of First Byte of Command (Code)

 0316 0D00                        .ptw   fcslcv          ; 1 = SET-CONTRAST command.
 0318 0F00                        .ptw   fcslcd          ; 2 = SEND-LCD command.
 031A 1400                        .ptw   fcsled          ; 3 = SEND-LED command.
 031C 1600                        .ptw   fcbeep          ; 4 = BEEP command.

 031E 970111          R  fcinit:  ld     numexp,#1       ; First byte of INITIALIZE command.
                                                         ;  Expects 1 more byte (RTC interval).
 0321 9420                        jmpl   upwret          ; Return.

                                                         ; First byte of SET-CONTRAST command.
 0323 970111          R  fcslcv:  ld     numexp,#1       ; Set up to expect one more byte.
 0326 5C                          jmpl   upwret

                                                         ; First byte of SEND-LCD command.
 0327 970211          R  fcslcd:  ld     numexp,#2       ; Set up to expect one more byte.
 032A 96100E          R           sbit   getcnt,curcmd   ; Note extended collection mode in Current
                                                         ;  Command byte.
 032D 55                          jmpl   upwret

                                                         ; First byte of SEND-LED command.
 032E 970111          R  fcsled:  ld     numexp,#1       ; Send to LED's: Set up to expect one more byte.
 0331 51                          jmpl   upwret

                                                         ; Process one-byte BEEP command.
 0332 96100F          R  fcbeep:  sbit   cmdemp,curcmd   ; No arguments; set CURCMD byte empty.
 0335 B601530F                    sbit   t7tfn,portph    ; Enable beep tone to panel speaker.
 0339 B6019008                    sbit   t0tie,tmmdl     ; Enable Timer T0 interrupt.
 033D 971319          R           ld     beepct,#19      ; Initialize duration count (approximately
                                                         ;  1 second, in units of Timer T0 overflows).
 0340 42                          jmpl   upwret


 0341 3741                  illc:  jsrl   hangup          ; Process illegal command codes.

                                                         ; Return from UPI Write interrupt.
 0343                       upwret:                      ; Restore Context
 0343 3FC0                         pop    psw
 0345 3FC8                         pop    A
 0347 3E                           reti

```

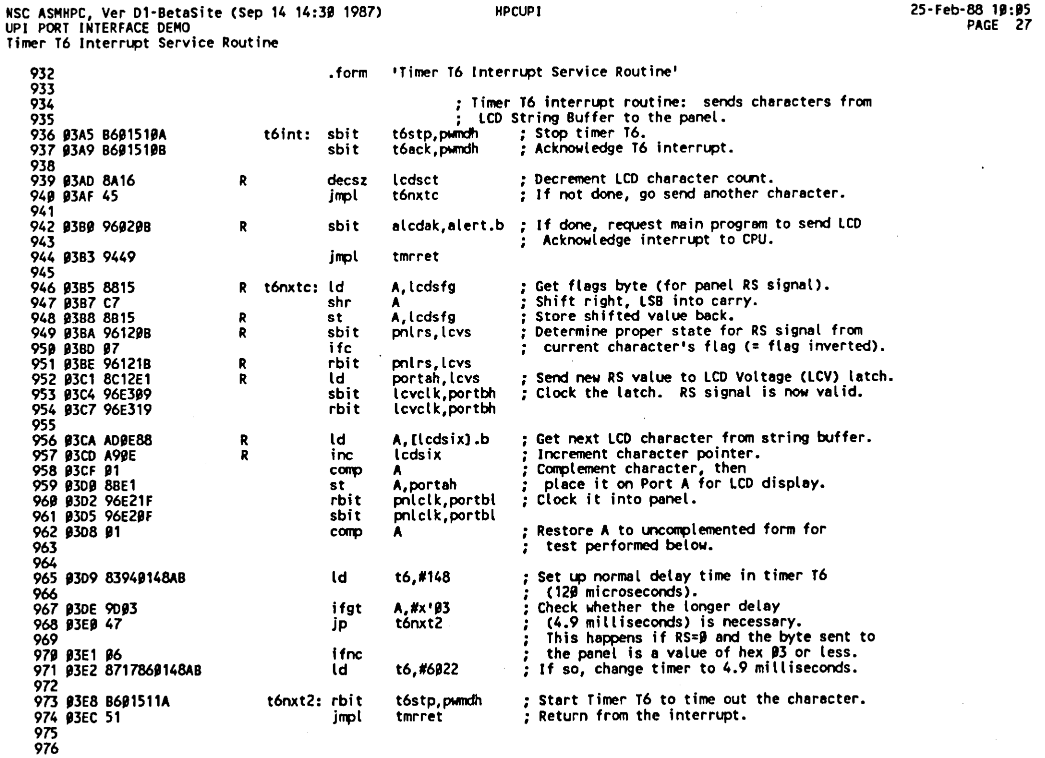
TL/DD/9976-40

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                         25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                        PAGE  25
Timer Interrupt Handler

                                      .form   'Timer Interrupt Handler'

FFF4 4803            R                .ipt    5,tmrint       ; Declare entry point for Timer Interrupt.

0348 AFC8                    tmrint:  push    A              ; Save context.
034A AFCC                             push    B              ;
034C AFC0                             push    psw            ;

034E B6019015                t1poll:  ifbit   t1pnd,tmmdl    ; Poll for Timer T1 interrupt (Real-Time Clock).
0352 54                               jmpl    t1int          ; If set, go service it.

0353 B6015111                t6poll:  ifbit   t6pnd,pwmdh    ; Poll for Timer T6 interrupt (LCD Panel Timing
0357 944C                             jmpl    t6int          ;  Interrupt).

0359 B6019011                t0poll:  ifbit   t0pnd,tmmdl    ; Poll for Timer T0 interrupt (Beep Duration).
035D 41                               jp      t0pdg          ; If set, check the Enable bit;  T0 is not
035E 46                               jp      t0notp         ;  always enabled to interrupt, but it runs
                                                             ;  continuously.

035F B6019010                t0pdg:   ifbit   t0tie,tmmdl    ; If enable is also set, then go service T0.
0363 9488                             jmpl    t0int
0365                         t0notp:                         ; (This label is deliberately here.)

0365 3765                    noint:   jsrl    hangup         ; Error:  no legal timer interrupt pending.


```

TL/DD/9976-41

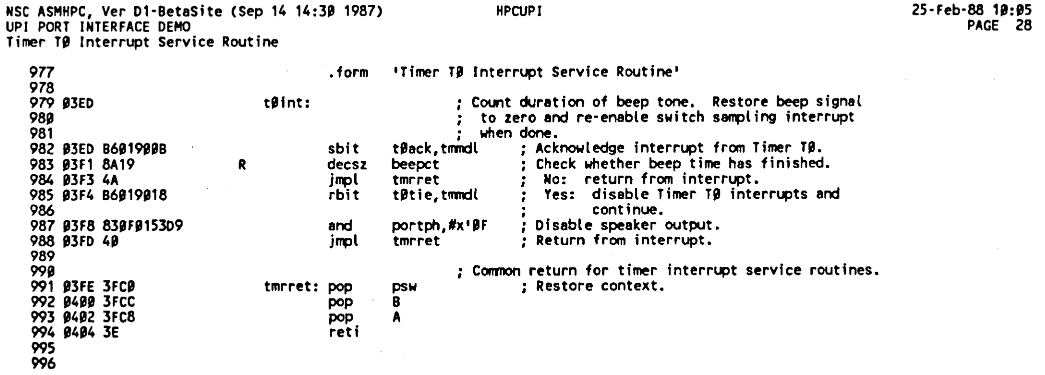
```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                         25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                        PAGE  26
Timer T1 Interrupt Service Routine

                                      .form   'Timer T1 Interrupt Service Routine'

0367 B601900F                t1int:   sbit    t1ack,tmmdl    ; Acknowledge T1 interrupt.
036B 961C10          R                ifbit   rtcenb,rtevs   ; Check if RTC interrupts are enabled.
036E 41                               jp      t1int1
036F 57                               jmpl    kbdchk         ; If not, then go check other events.
0370 8A1B            R       t1int1:  decsz   rtccnt         ; Decrement interval value.
0372 54                               jmpl    kbdchk         ; If interval has not elapsed, then go check
                                                             ;  for other events.
0373 8C1A1B          R                ld      rtccnt,rtcivl  ; Reload counter value for next interval.
0376 960211          R                ifbit   artc,alert.b   ; Check if CPU has received previous interrupt
0379 44                               jp      t1rerr         ;  request;  report error if not.
037A 960209          R                sbit    artc,alert.b   ; Set Real-Time Interrupt request to main
037D 49                               jp      kbdchk         ;  program.
037E 961D08          R       t1rerr:  sbit    0,dsevc        ; Signal NOTE severity.
0381 961E0F          R                sbit    7,derrc        ; Signal multiple-RTC error.
0384 96020A          R                sbit    adiag,alert.b  ; Request IDIAG interrupt from main program.

0387                         kbdchk:                         ; Check keyboard switches.
0387 96E31D                           rbit    astts,portbh   ; Enable pushbutton data to Port D.
038A B6010488                         ld      A,portd        ; Sample pushbutton switches.
038E 96E30D                           sbit    astts,portbh   ; Disable pushbutton data to Port D.
0391 98FF                             xor     A,#x'FF        ; Complement low-order 8 bits of A.
0393 8E17            R                x       A,swlast       ; Exchange with last sample.
0395 96170C          R                ifeq    A,swlast       ; Check if the data is stable (same as last
                                                             ;  sample).
0398 41                               jp      kbint1
0399 49                               jmpl    tmochk         ; If not, go check other events (if any).

039A 96180C          R       kbint1:  ifeq    A,swlsnt       ; Check if the data differs from the last
                                                             ;  pattern sent to the CPU.
039D 45                               jmpl    tmochk         ; If not, go check other events (if any).

039E 8818            R                st      A,swlsnt       ; Place new pattern in "last sent" location.
03A0 960208          R                sbit    abutton,alert.b ; Request "BUTTON-DATA" interrupt to CPU.


03A3                         tmochk:
                                              ; *** Insert any other RTC events here. ***

03A3 9459                             jmpl    tmrret         ; Return from Timer T1 interrupt.


```

TL/DD/9976-42



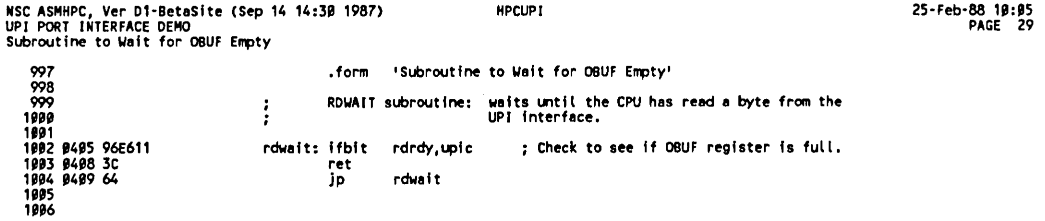
```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                         25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                          PAGE  27
Timer T6 Interrupt Service Routine

                                    .form   'Timer T6 Interrupt Service Routine'

                                                      ; Timer T6 interrupt routine:  sends characters from
                                                      ;  LCD String Buffer to the panel.
    03A5 B6015100A           t6int:  sbit    t6stp,pwmdh      ; Stop timer T6.
    03A9 B601510B                    sbit    t6ack,pwmdh      ; Acknowledge T6 interrupt.

    03AD 8A16              R         decsz   lcdsct           ; Decrement LCD character count.
    03AF 45                          jmpl    t6nxtc           ; If not done, go send another character.

    03B0 96020B            R         sbit    alcdak,alert.b   ; If done, request main program to send LCD
                                                              ;  Acknowledge interrupt to CPU.
    03B3 9449                        jmpl    tmrret

    03B5 8815              R t6nxtc: ld      A,lcdsfg         ; Get flags byte (for panel RS signal).
    03B7 C7                          shr     A                ; Shift right, LSB into carry.
    03B8 8B15              R         st      A,lcdsfg         ; Store shifted value back.
    03BA 96120B            R         sbit    pnlrs,lcvs       ; Determine proper state for RS signal from
    03BD 07                          ifc                      ;  current character's flag (= flag inverted).
    03BE 96121B            R         rbit    pnlrs,lcvs
    03C1 8C12E1            R         ld      portah,lcvs      ; Send new RS value to LCD Voltage (LCV) latch.
    03C4 96E309                      sbit    lcvclk,portbh    ; Clock the latch.  RS signal is now valid.
    03C7 96E319                      rbit    lcvclk,portbh

    03CA AD0E88            R         ld      A,[lcdsix].b     ; Get next LCD character from string buffer.
    03CD A90E              R         inc     lcdsix           ; Increment character pointer.
    03CF 01                          comp    A                ; Complement character, then
    03D0 8BE1                        st      A,portah         ;  place it on Port A for LCD display.
    03D2 96E21F                      rbit    pnlclk,portbl    ; Clock it into panel.
    03D5 96E20F                      sbit    pnlclk,portbl
    03D8 01                          comp    A                ; Restore A to uncomplemented form for
                                                              ;  test performed below.

    03D9 8394014800AB                ld      t6,#148          ; Set up normal delay time in timer T6
                                                              ;  (120 microseconds).
    03DE 9D03                        ifgt    A,#x'03          ; Check whether the longer delay
    03E0 47                          jp      t6nxt2           ;  (4.9 milliseconds) is necessary.
                                                              ;  This happens if RS=0 and the byte sent to
    03E1 06                          ifnc                     ;  the panel is a value of hex 03 or less.
    03E2 8717860148AB                ld      t6,#6022         ; If so, change timer to 4.9 milliseconds.

    03E8 B601511A            t6nxt2: rbit    t6stp,pwmdh      ; Start Timer T6 to time out the character.
    03EC 51                          jmpl    tmrret           ; Return from the interrupt.


```

TL/DD/9976-43



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                         25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                          PAGE  28
Timer T0 Interrupt Service Routine

                                    .form   'Timer T0 Interrupt Service Routine'

    03ED                     t0int:                   ; Count duration of beep tone.  Restore beep signal
                                                      ;  to zero and re-enable switch sampling interrupt
                                                      ;  when done.
    03ED B601900B                    sbit    t0ack,tmmdl      ; Acknowledge interrupt from Timer T0.
    03F1 8A19              R         decsz   beepct           ; Check whether beep time has finished.
    03F3 4A                          jmpl    tmrret           ;  No:  return from interrupt.
    03F4 B6019018                    rbit    t0tie,tmmdl      ;  Yes:  disable Timer T0 interrupts and
                                                              ;         continue.
    03F8 830F0153D9                  and     portph,#x'0F     ; Disable speaker output.
    03FD 40                          jmpl    tmrret           ; Return from interrupt.

                                                      ; Common return for timer interrupt service routines.
    03FE 3FC0                tmrret: pop     psw              ; Restore context.
    0400 3FCC                        pop     B
    0402 3FC8                        pop     A
    0404 3E                          reti


```

TL/DD/9976-44



```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)            HPCUPI                         25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                          PAGE  29
Subroutine to Wait for OBUF Empty

                                    .form   'Subroutine to Wait for OBUF Empty'

                             ;       RDWAIT subroutine:  waits until the CPU has read a byte from the
                             ;                           UPI interface.

    0405 96E611              rdwait: ifbit   rdrdy,upic       ; Check to see if OBUF register is full.
    0408 3C                          ret
    0409 64                          jp      rdwait


```

TL/DD/9976-45

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)             HPCUPI                                     25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                      PAGE  30
Write to Panel Subroutine

 1007                          .form   'Write to Panel Subroutine'
 1008
 1009                                          ; Write Panel subroutine.
 1010                                          ; Used only at initialization or to report a
 1011                                          ; fatal protocol error, since it performs
 1012                                          ; the timing delay using timer T6 without interrupts.
 1013                                          ;   (Panel RS signal must be set up previously in the
 1014                                          ;    LCV latch by the calling routine.)
 1015
 1016 040A 01                wrpnl:  comp    A                ; Complement value for bus.
 1017 040B 8BE1                      st      A,portah         ; Put value on panel bus.
 1018 040D 96E21F                    rbit    pnlclk,portbl    ; Set Panel Clock low,
 1019 0410 96E20F                    sbit    pnlclk,portbl    ;  then high again;
 1020                                                         ;  pulse width approx.
 1021                                                         ;  1.2 microsec.
 1022
 1023                                          ; Wait for another
 1024                                          ;  4.9 milliseconds (twice).
 1025 0413 8732C80148AB              ld      t6,#13000        ; Twice 4.9 milliseconds.
 1026 0419 B601511A                  rbit    t6stp,pwmdh      ; Start timer T6.
 1027 041D B6015111          wrplp:  ifbit   t6pnd,pwmdh      ; Wait for PND to be set.
 1028 0421 41                        jp      wrpgo
 1029 0422 65                        jp      wrplp
 1030 0423 B601510A          wrpgo:  sbit    t6stp,pwmdh      ; Stop timer T6.
 1031 0427 B601510B                  sbit    t6ack,pwmdh      ; Clear T6 PND bit.
 1032 042B 3C                        ret                      ; Return from subroutine.
 1033
 1034                        ; END OF PROGRAM:  RESET VECTOR SET TO LABEL "start".
 1035
 1036 042C                           .end    start
```

TL/DD/9976-46

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)             HPCUPI                                     25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                                      PAGE  31
Write to Panel Subroutine

 abutton  0000   Abs  Null
 adiag    0002   Abs  Null
 ah       00C9   Abs  Byte
 al       00C8   Abs  Byte
 alcdak   0003   Abs  Null
 alert    0002   Rel  Word  BASE
 alerth   0003   Rel  Byte  BASE
 artc     0001   Abs  Null
 astts    0005   Abs  Null
 avail    0020   Rel  Word  RAM16
 b2stp    0007   Abs  Null
 b8or16   0004   Abs  Null
 b8or9    0004   Abs  Null
 beepct   0019   Rel  Byte  BASE
 bfun     00F4   Abs  Word
 bfunh    00F5   Abs  Byte
 bfunl    00F4   Abs  Byte
 bh       00CD   Abs  Byte
 bl       00CC   Abs  Byte
 cdata    0004   Abs  Null
 chkalt   01A8   Rel  Null  ROM16
 cmdemp   0007   Abs  Null
 cpuad    0004   Rel  Word  BASE
 cpubuf   0006   Rel  Word  BASE
 curcmd   0010   Rel  Byte  BASE
 dbyte    001F   Rel  Byte  BASE
 dccmd    0020   Rel  Byte  BASE
 derrc    001E   Rel  Byte  BASE
 dirah    00F1   Abs  Byte
 dirb     00F2   Abs  Word
 dirbh    00F3   Abs  Byte
 dirbl    00F2   Abs  Byte
 divby    018E   Abs  Word
 divbyh   018F   Abs  Byte
 divbyl   018E   Abs  Byte
 doeerr   0007   Abs  Null
 dqual    0021   Rel  Byte  BASE
 dsevc    001D   Rel  Byte  BASE
 dummy    0000   Rel  Word  BASE
 ei       0007   Abs  Null
 eiack    0002   Abs  Null
 eicon    015C   Abs  Byte
 eimode   0001   Abs  Null
 eipol    0000   Abs  Null
 enir     00D0   Abs  Byte
 enu      0120   Abs  Byte
 enui     0122   Abs  Byte
 enur     0128   Abs  Byte
 eri      0001   Abs  Null
 eti      0000   Abs  Null
```

TL/DD/9976-47

```
 fcbeep   0332   Rel  Null  ROM16
 fcinit   031E   Rel  Null  ROM16
 fcord    0309   Rel  Null  ROM16
 fcslcd   0327   Rel  Null  ROM16
 fcslcv   0323   Rel  Null  ROM16
 fcsled   032E   Rel  Null  ROM16
 firstab  0314   Rel  Null  ROM16
 firstc   02E5   Rel  Null  ROM16
 frmerr   0006   Abs  Null
 getcnt   0006   Abs  Null
 gie      0000   Abs  Null
 hangup   0000   Rel  Null  ROM16
 hextab   007A   Rel  Byte  ROM16
 hgrst    006F   Rel  Null  ROM16
 hgupi    005E   Rel  Null  ROM16
 hgupi1   0069   Rel  Null  ROM16
 hgupi2   0076   Rel  Null  ROM16
 i2       0002   Abs  Null
 i3       0003   Abs  Null
 i4       0004   Abs  Null
 ibuf     00F0   Abs  Byte
 illc     0341   Rel  Null  ROM16
 ircd     00D4   Abs  Byte
 irpd     00D2   Abs  Byte
 kbdchk   0387   Rel  Null  ROM16
 kbint1   039A   Rel  Null  ROM16
 la0      0002   Abs  Null
 lastab   0246   Rel  Null  ROM16
 lastc    0234   Rel  Null  ROM16
 lastc1   0241   Rel  Null  ROM16
 lcdbuf   0038   Rel  Word  RAM16
 lcdfgs   0013   Rel  Byte  BASE
 lcdgo1   0171   Rel  Null  ROM16
 lcdlp1   0168   Rel  Null  ROM16
 lcdnum   0014   Rel  Byte  BASE
 lcdsct   0016   Rel  Byte  BASE
 lcdsfg   0015   Rel  Byte  BASE
 lcdsix   000E   Rel  Word  BASE
 lcinit   0250   Rel  Null  ROM16
 lcord    022A   Rel  Null  ROM16
 lcrst    0224   Rel  Null  ROM16
 lcslc1   02C6   Rel  Null  ROM16
 lcslc2   0291   Rel  Null  ROM16
 lcslcd   028C   Rel  Null  ROM16
 lcslcv   0274   Rel  Null  ROM16
 lcsled   02D8   Rel  Null  ROM16
 lcvclk   0001   Abs  Null
 lcvs     0012   Rel  Byte  BASE
 ledclk   0006   Abs  Null
 mainlp   01A8   Rel  Null  ROM16
```

TL/DD/9976-48

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)           HPCUPI                                25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                              PAGE  33
Write to Panel Subroutine

 minit     0191   Rel   Null   ROM16
 noint     0365   Rel   Null   ROM16
 numexp    0011   Rel   Byte   BASE
 obuf      00E0   Abs   Byte
 pnlclk    0007   Abs   Null
 pnlrs     0003   Abs   Null
 portah    00E1   Abs   Byte
 portb     00E2   Abs   Word
 portbh    00E3   Abs   Byte
 portbl    00E2   Abs   Byte
 portd     0104   Abs   Byte
 porti     00D8   Abs   Byte
 portp     0152   Abs   Word
 portph    0153   Abs   Byte
 portpl    0152   Abs   Byte
 psw       00C0   Abs   Word
 pwmdh     0151   Abs   Byte
 pwmdl     0150   Abs   Byte
 pwmode    0150   Abs   Word
 r1        0184   Abs   Word
 r2        0186   Abs   Word
 r3        018A   Abs   Word
 r4        0142   Abs   Word
 r5        0146   Abs   Word
 r6        014A   Abs   Word
 r7        014E   Abs   Word
 rbfl      0001   Abs   Null
 rbit9     0003   Abs   Null
 rbuf      0124   Abs   Byte
 rdrdy     0001   Abs   Null
 rdwait    0405   Rel   Null   ROM16
 rtccnt    001B   Rel   Byte   BASE
 rtcenb    0000   Abs   Null
 rtcivl    001A   Rel   Byte   BASE
 rtevs     001C   Rel   Byte   BASE
 runsys    019F   Rel   Null   ROM16
 sio       00D6   Abs   Byte
 sk        0006   Abs   Null
 slcd      0145   Rel   Null   ROM16
 sled      0104   Rel   Null   ROM16
 sndbtn    01D2   Rel   Null   ROM16
 sndiag    01E4   Rel   Null   ROM16
 sndlak    01CA   Rel   Null   ROM16
 sndrtc    01C2   Rel   Null   ROM16
 so        0005   Abs   Null
 sram      00C4   Rel   Null   ROM16
 sraml1    00C7   Rel   Null   ROM16
 sraml2    00CF   Rel   Null   ROM16
 srfsh     008D   Rel   Null   ROM16
 sskint    00D2   Rel   Null   ROM16
```

TL/DD/9976-49

```
ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                                   25-Feb-88 10:05
PORT INTERFACE DEMO                                                                              PAGE  34
e to Panel Subroutine

ckb   0000   Rel   Word  RAM16
rt    008A   Rel   Null  ROM16
rs    0113   Rel   Null  ROM16
i     00A7   Rel   Null  ROM16
ast   0017   Rel   Byte  BASE
snt   0018   Rel   Byte  BASE
ck    0003   Abs   Null
bn    0192   Abs   Byte
ht    03ED   Rel   Null  ROM16
btp   0365   Rel   Null  ROM16
dg    035F   Rel   Null  ROM16
d     0001   Abs   Null
ll    0359   Rel   Null  ROM16
e     0000   Abs   Null
      0182   Abs   Word
k     0007   Abs   Null
t     0367   Rel   Null  ROM16
t1    0370   Rel   Null  ROM16
d     0005   Abs   Null
ll    034E   Rel   Null  ROM16
rr    037E   Rel   Null  ROM16
p     0006   Abs   Null
e     0004   Abs   Null
      0188   Abs   Word
k     0003   Abs   Null
d     0001   Abs   Null
b     0002   Abs   Null
e     0000   Abs   Null
t3    018C   Abs   Word
t3k   0007   Abs   Null
t3d   0005   Abs   Null
t3b   0006   Abs   Null
t3e   0004   Abs   Null
t4    0140   Abs   Word
t4    0003   Abs   Null
t4    0000   Abs   Null
t4    0001   Abs   Null
t4    0002   Abs   Null
t4    0003   Abs   Null
t4    0000   Abs   Null
t5    0144   Abs   Word
t5a   0007   Abs   Null
t5o   0004   Abs   Null
t5p   0005   Abs   Null
t5s   0006   Abs   Null
t5t   0007   Abs   Null
t5t   0004   Abs   Null
t6    0148   Abs   Word
t6a   0003   Abs   Null
t6i   03A5   Rel   Null  ROM16
```

TL/DD/9976-50

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                   PAGE  35
Write to Panel Subroutine

 t6nxt2   03E8   Rel  Null  ROM16
 t6nxtc   03B5   Rel  Null  ROM16
 t6out    0000   Abs  Null
 t6pnd    0001   Abs  Null
 t6poll   0353   Rel  Null  ROM16
 t6stp    0002   Abs  Null
 t6tfn    0003   Abs  Null
 t6tie    0000   Abs  Null
 t7       014C   Abs  Word
 t7ack    0007   Abs  Null
 t7out    0004   Abs  Null
 t7pnd    0005   Abs  Null
 t7stp    0006   Abs  Null
 t7tfn    0007   Abs  Null
 t7tie    0004   Abs  Null
 tbmt     0000   Abs  Null
 tbuf     0126   Abs  Byte
 tminit   00DF   Rel  Null  ROM16
 tmmdh    0191   Abs  Byte
 tmmdl    0190   Abs  Byte
 tmmode   0190   Abs  Word
 tmochk   03A3   Rel  Null  ROM16
 tmrint   0348   Rel  Null  ROM16
 tmrret   03FE   Rel  Null  ROM16
 tmrs     0005   Abs  Null
 uart     0006   Abs  Null
 upic     00E6   Abs  Byte
 upicsv   0000   Rel  Word  BASE
 upien    0003   Abs  Null
 upiwr    020D   Rel  Null  ROM16
 upwret   0343   Rel  Null  ROM16
 urdrdy   0007   Abs  Null
 uwdone   0000   Abs  Null
 uwmode   0001   Abs  Null
 uwrrdy   0003   Abs  Null
 vbutton  0018   Abs  Null
 vdiag    001D   Abs  Null
 vlcdak   0017   Abs  Null
 vrtc     0011   Abs  Null
 wakeup   0002   Abs  Null
 wrpgo    0423   Rel  Null  ROM16
 wrplp    041D   Rel  Null  ROM16
 wrpnl    040A   Rel  Null  ROM16
 wrrdy    0000   Abs  Null
 xbit9    0005   Abs  Null
 xh       00CF   Abs  Byte
 xl       00CE   Abs  Byte
 xoff     0013   Abs  Null
 xon      0011   Abs  Null
 xrclk    0003   Abs  Null
```

TL/DD/9976-51

```
NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)          HPCUPI                    25-Feb-88 10:05
UPI PORT INTERFACE DEMO                                                                   PAGE  36
Write to Panel Subroutine

 xreset   02F0   Rel  Null  ROM16
 xtclk    0002   Abs  Null


 **** Errors:     0, Warnings:    0
```

TL/DD/9976-52

4.3 Two Demo Programs (NS32CG16 Source Code)

The following two programs run on the NS32CG16 CPU, and exercise the functions implemented in the HPC firmware.

One thing to note in this software is that the interrupt service routines are not written as such; they are simple subroutines called by the actual service routines, which are contained within a modified version of the MON16 monitor program. The reasons for modifying MON16 were two-fold:

1. There is no RAM in the application system within the first 64k of the addressing space. The presence of RAM there is necessary for MON16 to support custom interrupt handlers without internal modification.
2. The HPC requires use of the "RETT 0" instruction, rather than "RETI", to return from maskable as well as non-maskable interrupts.

Given these two constraints, it was considered most useful to modify MON16 to contain a set of interrupt service routines, which would then use a set of addresses in RAM (a table at address "vex") to call custom interrupt servers as standard subroutines. An interrupt service routine calls its custom subroutine after saving the dedicated registers and the general registers, R0, R1 and R2 on the stack.

The symbol "vex" is defined externally, and must be declared to match the address used by the modified MON16.

Details of the modified MON16 are available from National Semiconductor Corporation, Microprocessor Applications Group or the Microcontroller Applications Group, phone (408) 721-5000. These modifications are also a standard part of the MONCG monitor program for the NS32CG016 microprocessor.

4.3.1 Panel Exerciser Program

This program for the NS32CG16 CPU exercises several functions of a panel consisting of the following:

- A two-line (8 chars. per line) LCD panel, arranged horizontally into a single 16-line display.
- A speaker, activated by the BEEP command.
- Six pushbuttons, which are presented by the !BUTTON-DATA interrupt to the CPU as follows:

Keyboard Status Byte

0	PB6	PB5	PB4	0	PB2	PB1	PB0

- Five LED's, activated in the SEND-LED command by the following bits:

LED Control Byte

—	—	LD5	LD4	LD3	LD2	LD1	—

The intended layout for the front panel is as shown below. (Please pardon the apparently haphazard assignment of the pushbuttons and LED's; this was dictated by the nature of the module we used for developing this application.)

Front Panel Layout

Cursor Addr. →	00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47
LCD's:	*			*			*			*			*			*
LED's:	LD2			LD3			LD1			LD5			LD4			(Beep)
PB's:	PB5			PB1			PB0			PB4			PB6			PB2

The locations shown with asterisks on the LCD panel above will display an asterisk character while the corresponding pushbutton below it is depressed. (The number above each LCD location indicates its cursor address in hexadecimal.)

Each time a pushbutton (except PB2) is pressed, the corresponding LED indicator above it is toggled. Rather than toggling an LED, PB2 causes a BEEP command to be issued.

The program starts up the panel with the LCD display blank, and LED's LD1 and LD2 on.

```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88        Page: 1

   1
   2                                        # Front Panel Exerciser Program.
   3
   4                                        # "vex" contains absolute address of NMI service routine entry point.
   5                                        # "vex"+4 starts list of maskable interrupt routine entry points;
   6                                        #         first is interrupt 0x10.
   7
   8                                        # Note:  This code assumes that it is running in Supervisor Mode.
   9                                        #        Before running, make sure to set PSR to 0200 hex.
  10                                        #        Also, all unused interrupts automatically branch to label
  11                                        #        "badint";  a breakpoint should be set there.
  12
  13
  14                                        .globl  start,main
  15                                        .globl  rtcint
  16                                        .globl  lcdint
  17                                        .globl  swint
  18                                        .globl  badint
  19
  20                                        .set    hpcctrl,0xFFFC00        # HPC Control/Status I/O location.
  21                                        .set    hpcdata,0xFFFE00        # HPC Data I/O location.
  22                                        .set    hpcpoll,0xFD0000        # HPC Poll address (UPIC).
  23
  24                                        .set    INIT,0x0
  25                                        .set    SET_CONT,0x1
  26                                        .set    SEND_LCD,0x2
  27                                        .set    SEND_LED,0x3
  28                                        .set    BEEP,0x4
  29                                        .set    RESET_HPC,0xA5
  30
  31                                start:
  32                                                                # Fill interrupt vector locations.
  33
  34  T00000000  67ddc000                   addr    badint,vex              # Interrupt NMI.    (Unimplemented)
                 025a0000
                 0000
  35  T0000000a  67ddc000                   addr    badint,vex+4            # Interrupt 0x10.   (Unimplemented)
                 02500000
                 0004
  36  T00000014  67ddc000                   addr    rtcint,vex+8            # Interrupt 0x11.   Real-Time Clock.
                 021c0000
                 0008
  37  T0000001e  67ddc000                   addr    badint,vex+12           # Interrupt 0x12.   (Unimplemented)
                 023c0000
                 000c
  38  T00000028  67ddc000                   addr    badint,vex+16           # Interrupt 0x13.   (Unimplemented)
                 02320000
                 0010
  39  T00000032  67ddc000                   addr    badint,vex+20           # Interrupt 0x14.   (Unimplemented)
                 02280000
                 0014
  40  T0000003c  67ddc000                   addr    badint,vex+24           # Interrupt 0x15.   (Unimplemented)
                 021e0000
                 0018
  41  T00000046  67ddc000                   addr    badint,vex+28           # Interrupt 0x16.   (Unimplemented)
```

TL/DD/9976-53

```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 2

                    02140000
                    001c
 42  T00000050  67ddc000          addr    lcdint,vex+32        # Interrupt 0x17.  LCD data written.
                    01e80000
                    0020
 43  T0000005a  67ddc000          addr    swint,vex+36         # Interrupt 0x18.  Pushbutton event.
                    01ea0000
                    0024
 44  T00000064  67ddc000          addr    badint,vex+40        # Interrupt 0x19.  (Unimplemented)
                    01f60000
                    0028
 45  T0000006e  67ddc000          addr    badint,vex+44        # Interrupt 0x1A.  (Unimplemented)
                    01ec0000
                    002c
 46  T00000078  67ddc000          addr    badint,vex+48        # Interrupt 0x1B.  (Unimplemented)
                    01e20000
                    0030
 47  T00000082  67ddc000          addr    badint,vex+52        # Interrupt 0x1C.  (Unimplemented)
                    01d80000
                    0034
 48  T0000008c  67ddc000          addr    badint,vex+56        # Interrupt 0x1D.  Diagnostic:  stop.
                    01ce0000
                    0038
 49  T00000096  67ddc000          addr    badint,vex+60        # Interrupt 0x1E.  (Unimplemented)
                    01c40000
                    003c
 50  T000000a0  67ddc000          addr    badint,vex+64        # Interrupt 0x1F.  (Unimplemented)
                    01ba0000
                    0040
 51  T000000aa  67ddc000          addr    badint,vex+68        # Interrupt 0x20.  (Unimplemented)
                    01b00000
                    0044
 52  T000000b4  67ddc000          addr    badint,vex+72        # Interrupt 0x21.  (Unimplemented)
                    01a60000
                    0048
 53
 54  T000000be  54a500c0          movb    $INIT,hpcctrl    # INITIALIZE command.
                    fffc00
 55  T000000c5  54a500c0          movb    $0,hpcdata       # RTC value:  feature disabled.
                    fffe00
 56
 57  T000000cc  54a503c0          movb    $SEND_LED,hpcctrl      # Initialize LEDs to normal state.
                    fffc00
 58  T000000d3  54a506c0          movb    $0x06,hpcdata
                    fffe00
 59  T000000da  d4a606c0          movb    $0x06,leds       # Save in memory image.
                    000145
 60
 61                         run:
 62  T000000e1  7da30800          bispsrw $0x800         # Enable interrupts from HPC.
 63
 64
 65                         main:                   # Main program starts here.
 66
 67  T000000e5  5cd8c000          movqb   $0,lcdflg       # Set waiting for LCD.
```

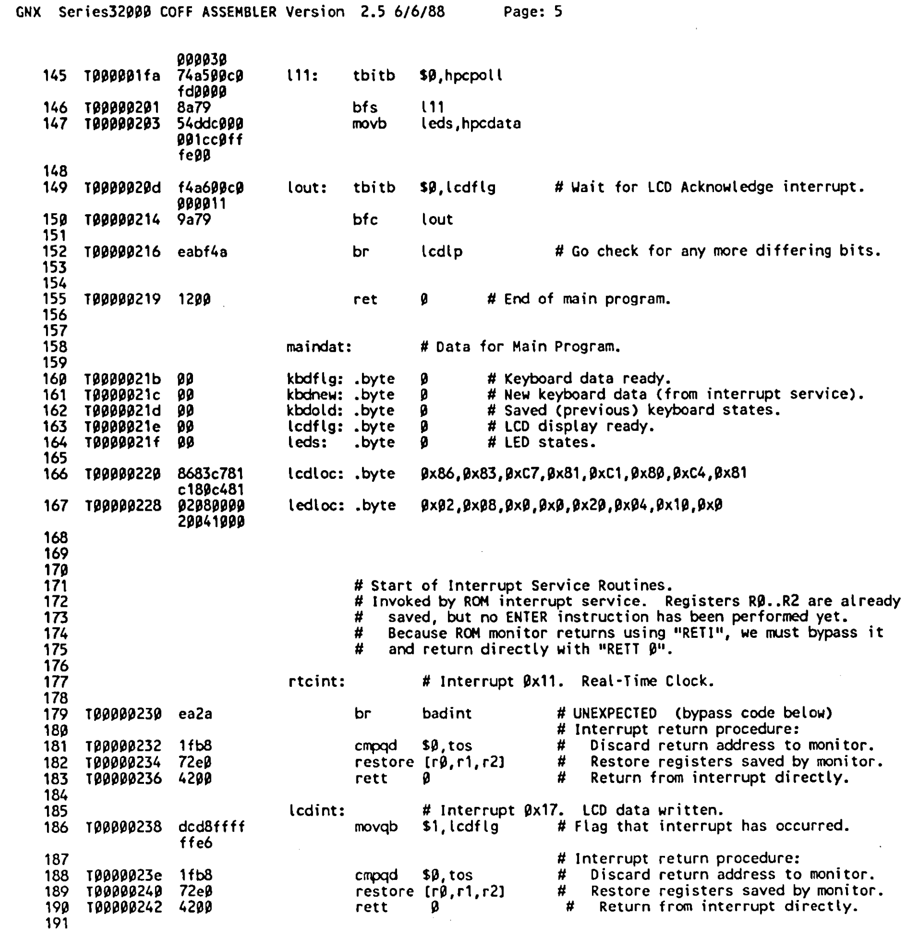
TL/DD/9976-54

```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 3

                   0139
 68
 69  T000000eb  54a502c0           movb   $SEND_LCD,hpcctrl     # Turn off LCD cursor and clear panel.
                fffc00
 70  T000000f2  54a500c0           movb   $0,hpcdata
                fffe00
 71  T000000f9  54a502c0           movb   $2,hpcdata
                fffe00
 72  T00000100  54a50cc0           movb   $0x0C,hpcdata
                fffe00
 73  T00000107  54a501c0           movb   $1,hpcdata
                fffe00
 74
 75  T0000010e  f4a600c0    l1:    tbitb  $0,lcdflg        # Wait for panel available.
                000110
 76  T00000115  9a79               bfc    l1
 77
 78  T00000117  5cd8c000           movqb  $0,kbdflg
                0104
 79
 80
 81                         kbdlp:
 82
 83  T0000011d  f4a600c0    l2:    tbitb  $0,kbdflg        # Wait for keyboard data.
                0000fe
 84  T00000124  9a79               bfc    l2
 85
 86  T00000126  7da10800           bicpsrw $0x800          # Sample, and update semaphores.
 87  T0000012a  14d8c000           movb   kbdnew,r0
                00f2
 88  T00000130  54d8c000           movb   kbdold,r1
                00ed
 89  T00000136  d4dec000           movb   kbdnew,kbdold
                00e6c000
                00e7
 90  T00000140  5cd8c000           movqb  $0,kbdflg
                00db
 91  T00000146  7da30800           bispsrw $0x800
 92
 93  T0000014a  5f10               movqd  $0,r2            # Initialize offset pointer in r2.
 94
 95  T0000014c  7800               xorb   r0,r1            # Generate map of differing bits.
 96  T0000014e  1c08               cmpqb  $0,r1            # Check that a change actually occurred.
 97  T00000150  1a10               bne    lcdlp
 98
 99  T00000152  54a503c0           movb   $SEND_LED,hpcctrl # If not, error is shown by turning on
                fffc00
100  T00000159  54a520c0           movb   $0x20,hpcdata    #   ALARM LED.
                fffe00
101
102
103                         lcdlp:
104  T00000160  6e8408             ffsb   r1,r2            # Find first differing bit.
105  T00000163  8abfba             bfs    kbdlp            # If none, go wait for another keyboard event.
106  T00000166  4e4810             cbitb  r2,r1            # Clear difference flag.
```

TL/DD/9976-55

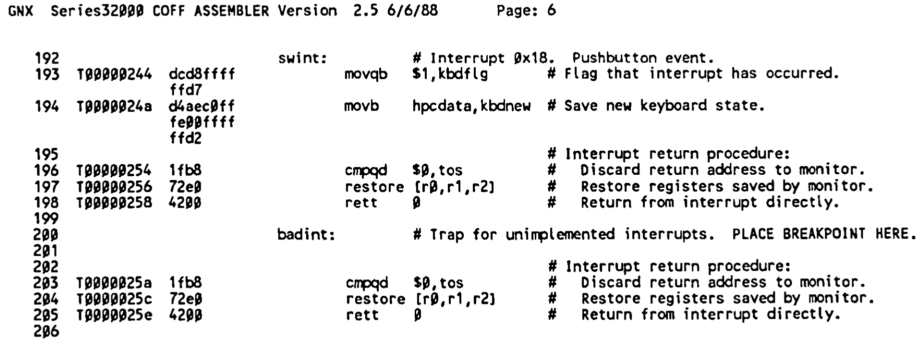
```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 4

107
108  T00000169  5cd8c000          movqb   $0,lcdflg       # Do LCD command:  first clear Acknowledge flag.
                00b5
109
110  T0000016f  74a500c0   l3:    tbitb   $0,hpcpoll
                fd0000
111  T00000176  8a79              bfs     l3
112  T00000178  54a502c0          movb    $SEND_LCD,hpcctrl # Start command to display new bit state.
                fffc00
113
114  T0000017f  74a500c0   l4:    tbitb   $0,hpcpoll
                fd0000
115  T00000186  8a79              bfs     l4
116  T00000188  54a502c0          movb    $2,hpcdata      # Flags:  One command followed by one data.
                fffe00
117
118  T0000018f  74a500c0   l5:    tbitb   $0,hpcpoll
                fd0000
119  T00000196  8a79              bfs     l5
120  T00000198  54a502c0          movb    $2,hpcdata      # Two data bytes follow.
                fffe00
121
122  T0000019f  74a500c0   l6:    tbitb   $0,hpcpoll
                fd0000
123  T000001a6  8a79              bfs     l6
124  T000001a8  54e5dac0          movb    lcdloc[r2:b],hpcdata    # Send cursor position byte.
                000078c0
                fffe00
125
126  T000001b3  74a500c0   l7:    tbitb   $0,hpcpoll
                fd0000
127  T000001ba  8a79              bfs     l7
128  T000001bc  3410              tbitb   r2,r0
129  T000001be  8a0c              bfs     l8
130  T000001c0  54a520c0          movb    $0x20,hpcdata   # If new bit is zero, send blank.
                fffe00
131  T000001c7  ea8046            br      lout
132
133  T000001ca  54a52ac0   l8:    movb    $0x2A,hpcdata   # If bit is one, send asterisk instead,
                fffe00
134
135  T000001d1  74a500c0   l9:    tbitb   $0,hpcpoll
                fd0000
136  T000001d8  8a79              bfs     l9
137  T000001da  34a002            tbitb   $2,r0           #  and if the key is MENU,
138  T000001dd  9a0b              bfc     l10
139
140  T000001df  54a504c0          movb    $BEEP,hpcctrl   #  then beep,
                fffc00
141  T000001e6  ea27              br      lout
142
143  T000001e8  54a503c0   l10:   movb    $SEND_LED,hpcctrl       #  else toggle appropriate LED.
                fffc00
144  T000001ef  f8e6dac0          xorb    ledloc[r2:b],leds
                000039c0
```

TL/DD/9976-56



```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 5

                   000030
   145  T000001fa  74a500c0    l11:    tbitb   $0,hpcpoll
                   fd0000
   146  T00000201  8a79                bfs     l11
   147  T00000203  54ddc000            movb    leds,hpcdata
                   001cc0ff
                   fe00
   148
   149  T0000020d  f4a600c0    lout:   tbitb   $0,lcdflg       # Wait for LCD Acknowledge interrupt.
                   000011
   150  T00000214  9a79                bfc     lout
   151
   152  T00000216  eabf4a              br      lcdlp           # Go check for any more differing bits.
   153
   154
   155  T00000219  1200                ret     0       # End of main program.
   156
   157
   158                          maindat:        # Data for Main Program.
   159
   160  T0000021b  00          kbdflg: .byte   0       # Keyboard data ready.
   161  T0000021c  00          kbdnew: .byte   0       # New keyboard data (from interrupt service).
   162  T0000021d  00          kbdold: .byte   0       # Saved (previous) keyboard states.
   163  T0000021e  00          lcdflg: .byte   0       # LCD display ready.
   164  T0000021f  00          leds:   .byte   0       # LED states.
   165
   166  T00000220  8683c781    lcdloc: .byte   0x86,0x83,0xC7,0x81,0xC1,0x80,0xC4,0x81
                   c180c481
   167  T00000228  02080000    ledloc: .byte   0x02,0x08,0x0,0x0,0x20,0x04,0x10,0x0
                   20041000
   168
   169
   170
   171                                 # Start of Interrupt Service Routines.
   172                                 # Invoked by ROM interrupt service.  Registers R0..R2 are already
   173                                 #   saved, but no ENTER instruction has been performed yet.
   174                                 #   Because ROM monitor returns using "RETI", we must bypass it
   175                                 #   and return directly with "RETT 0".
   176
   177                         rtcint:         # Interrupt 0x11.  Real-Time Clock.
   178
   179  T00000230  ea2a                br      badint          # UNEXPECTED  (bypass code below)
   180                                                         # Interrupt return procedure:
   181  T00000232  1fb8                cmpqd   $0,tos          #   Discard return address to monitor.
   182  T00000234  72e0                restore [r0,r1,r2]      #   Restore registers saved by monitor.
   183  T00000236  4200                rett    0               #   Return from interrupt directly.
   184
   185                         lcdint:         # Interrupt 0x17.  LCD data written.
   186  T00000238  dcd8ffff            movqb   $1,lcdflg       # Flag that interrupt has occurred.
                   ffe6
   187                                                         # Interrupt return procedure:
   188  T0000023e  1fb8                cmpqd   $0,tos          #   Discard return address to monitor.
   189  T00000240  72e0                restore [r0,r1,r2]      #   Restore registers saved by monitor.
   190  T00000242  4200                rett    0               #   Return from interrupt directly.
   191
```

TL/DD/9976–57



```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 6

   192                         swint:          # Interrupt 0x18.  Pushbutton event.
   193  T00000244  dcd8ffff            movqb   $1,kbdflg       # Flag that interrupt has occurred.
                   ffd7
   194  T0000024a  d4aec0ff            movb    hpcdata,kbdnew  # Save new keyboard state.
                   fe00ffff
                   ffd2
   195                                                         # Interrupt return procedure:
   196  T00000254  1fb8                cmpqd   $0,tos          #   Discard return address to monitor.
   197  T00000256  72e0                restore [r0,r1,r2]      #   Restore registers saved by monitor.
   198  T00000258  4200                rett    0               #   Return from interrupt directly.
   199
   200                         badint:         # Trap for unimplemented interrupts.  PLACE BREAKPOINT HERE.
   201
   202                                                         # Interrupt return procedure:
   203  T0000025a  1fb8                cmpqd   $0,tos          #   Discard return address to monitor.
   204  T0000025c  72e0                restore [r0,r1,r2]      #   Restore registers saved by monitor.
   205  T0000025e  4200                rett    0               #   Return from interrupt directly.
   206
```

TL/DD/9976–58

4.3.2 Real-Time Clock Display Program

This program (rtc.s) enables the Real-Time Clock interrupts from the HPC, and counts them to generate a display of elapsed time on the LCD panel.

```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 1


                                # Real-Time Clock Exerciser:  Places elapsed time in seconds onto
                                #                               LCD Panel.

                                # "vex"   contains absolute address of NMI service routine entry point.
                                # "vex"+4 starts list of maskable interrupt routine entry points;
                                #         first is interrupt 0x10.

                                # Note:  This code assumes that it is running in Supervisor Mode.
                                #        Before running, make sure to set PSR to 0200 hex.
                                #        Also, all unused interrupts automatically branch to label
                                #        "badint";  a breakpoint should be set there.


                                .globl  start,main
                                .globl  rtcint
                                .globl  lcdint
                                .globl  swint
                                .globl  badint

                                .set    hpcctrl,0xFFFC00       # HPC Control/Status I/O location.
                                .set    hpcdata,0xFFFE00       # HPC Data I/O location.
                                .set    hpcpoll,0xFD0000       # HPC Poll address (UPIC).
                                .set    INIT,0x0
                                .set    SET_CONT,0x1
                                .set    SEND_LCD,0x2
                                .set    SEND_LED,0x3
                                .set    BEEP,0x4
                                .set    RESET_HPC,0xA5

                         start:
                                                     # Fill interrupt vector locations.

T00000000 67ddc000              addr    badint,vex             # Interrupt NMI.   (Unimplemented)
          024e0000
          0000
T0000000a 67ddc000              addr    badint,vex+4           # Interrupt 0x10.  (Unimplemented)
          02440000
          0004
T00000014 67ddc000              addr    rtcint,vex+8           # Interrupt 0x11.  Real-Time Clock.
          02040000
          0008
T0000001e 67ddc000              addr    badint,vex+12          # Interrupt 0x12.  (Unimplemented)
          02300000
          000c
T00000028 67ddc000              addr    badint,vex+16          # Interrupt 0x13.  (Unimplemented)
          02260000
          0010
T00000032 67ddc000              addr    badint,vex+20          # Interrupt 0x14.  (Unimplemented)
          021c0000
          0014
T0000003c 67ddc000              addr    badint,vex+24          # Interrupt 0x15.  (Unimplemented)
          02120000
          0018
T00000046 67ddc000              addr    badint,vex+28          # Interrupt 0x16.  (Unimplemented)
```

TL/DD/9976-59

```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88       Page: 2

                 02080000
                 001c
 42  T00000050   67ddc000            addr    lcdint,vex+32          # Interrupt 0x17.  LCD data written.
                 01e40000
                 0020
 43  T0000005a   67ddc000            addr    swint,vex+36           # Interrupt 0x18.  Pushbutton event.
                 01e80000
                 0024
 44  T00000064   67ddc000            addr    badint,vex+40          # Interrupt 0x19.  (Unimplemented)
                 01ea0000
                 0028
 45  T0000006e   67ddc000            addr    badint,vex+44          # Interrupt 0x1A.  (Unimplemented)
                 01e00000
                 002c
 46  T00000078   67ddc000            addr    badint,vex+48          # Interrupt 0x1B.  (Unimplemented)
                 01d60000
                 0030
 47  T00000082   67ddc000            addr    badint,vex+52          # Interrupt 0x1C.  (Unimplemented)
                 01cc0000
                 0034
 48  T0000008c   67ddc000            addr    badint,vex+56          # Interrupt 0x1D.  Diagnostic:  stop.
                 01c20000
                 0038
 49  T00000096   67ddc000            addr    badint,vex+60          # Interrupt 0x1E.  (Unimplemented)
                 01b80000
                 003c
 50  T000000a0   67ddc000            addr    badint,vex+64          # Interrupt 0x1F.  (Unimplemented)
                 01ae0000
                 0040
 51  T000000aa   67ddc000            addr    badint,vex+68          # Interrupt 0x20.  (Unimplemented)
                 01a40000
                 0044
 52  T000000b4   67ddc000            addr    badint,vex+72          # Interrupt 0x21.  (Unimplemented)
                 019a0000
                 0048
 53
 54  T000000be   54a500c0            movb    $INIT,hpcctrl    # INITIALIZE command.
                 fffc00
 55  T000000c5   54a505c0            movb    $5,hpcdata       # RTC value:  interval of 50 milliseconds.
                 fffe00
 56
 57  T000000cc   5cd8c000            movqb   $0,flags       . # Clear interrupt flags.
                 013e
 58                                  .set    rtcflg,0         # Bit 0 means RTC interrupt detected.
 59                                  .set    lcdflg,1         # Bit 1 means LCD interrupt detected.
 60  T000000d2   d4a614c0            movb    $20,rtcctr       # Clear RTC modulus counter (div by 20).
                 000139
 61  T000000d9   5fd8c000            movqd   $0,timcnt        # Clear seconds counter.
                 0133
 62
 63                          run:
 64  T000000df   7da30800            bispsrw $0x800           # Enable interrupts from HPC.
 65
 66                                                   # Neither communication port is selected yet.
 67
```

TL/DD/9976-60

```
GNX  Series32000 COFF ASSEMBLER Version  2.5 6/6/88        Page: 3

 68                               main:                  # Put main program here.
 69
 70  T000000e3  4ec8a601                cbitb  $lcdflg,flags   # Place cursor at first character of panel.
                c0000127
 71  T000000eb  54a502c0                movb   $SEND_LCD,hpcctrl
                fffc00
 72  T000000f2  54a500c0                movb   $0,hpcdata
                fffe00
 73  T000000f9  54a501c0                movb   $1,hpcdata
                fffe00
 74  T00000100  54a580c0                movb   $0x80,hpcdata
                fffe00
 75  T00000107  f4a601c0      l1:       tbitb  $lcdflg,flags
                000103
 76  T0000010e  9a79                    bfc    l1
 77
 78  T00000110  4ec8a601                cbitb  $lcdflg,flags   # Write initial value of zeroes.
                c00000fa
 79  T00000118  54a502c0                movb   $SEND_LCD,hpcctrl
                fffc00
 80  T0000011f  54a5ffc0                movb   $0xFF,hpcdata
                fffe00
 81  T00000126  54a508c0                movb   $8,hpcdata
                fffe00
 82  T0000012d  54a530c0                movb   $0x30,hpcdata
                fffe00
 83  T00000134  54a530c0                movb   $0x30,hpcdata
                fffe00
 84  T0000013b  54a530c0                movb   $0x30,hpcdata
                fffe00
 85  T00000142  54a530c0                movb   $0x30,hpcdata
                fffe00
 86  T00000149  54a530c0                movb   $0x30,hpcdata
                fffe00
 87  T00000150  54a530c0                movb   $0x30,hpcdata
                fffe00
 88  T00000157  54a530c0                movb   $0x30,hpcdata
                fffe00
 89  T0000015e  54a530c0                movb   $0x30,hpcdata
                fffe00
 90  T00000165  f4a601c0      l2:       tbitb  $lcdflg,flags
                0000a5
 91  T0000016c  9a79                    bfc    l2
 92
 93  T0000016e  f4a600c0      mainlp:   tbitb  $rtcflg,flags
                00009c
 94  T00000175  9a79                    bfc    mainlp
 95  T00000177  4ec8a600                cbitb  $rtcflg,flags
                c0000093
 96
 97  T0000017f  7ca101                  bicpsrb $0x01          # Clear carry.
 98  T00000182  4effa600                addpd  $0x01,timcnt    # Increment BCD elapsed time.
                000001c0
                00008a
 99
```

TL/DD/9976-61

The HPC as a Front-End Processor

National Semiconductor
Application Note 551
Brian Marley

ABSTRACT

This application note covers the use of the National Semiconductor HPC46083 High-Performance microController as a front-end processor to collect and block data from RS-232 (serial) and Centronics (parallel) ports for a Host CPU (a typical application being an intelligent graphics-oriented printer). This application note builds on Application Note AN-550 (UPI Port); the result being a program that implements a versatile front-end processor for a National NS32CG16 CPU.

1.0 INTRODUCTION

In Application Note AN-550, "A Software Driver for the HPC Universal Peripheral Interface Port", we saw how a National Semiconductor HPC46083 microcontroller can be connected and programmed to perform intelligent peripheral functions for a host CPU; our example being an application connecting an NS32CG16 CPU through the HPC to a typical front panel.

In this application note, we will expand on the hardware and the driver software presented there in order to implement a very useful function for a high-performance microcontroller: that of a front-end processor for data collection. To demonstrate a real-world application for this kind of function, we implement here an intelligent interface to a Centronics-style parallel input port and an RS-232 serial port, typical of a graphics-oriented printer.

2.0 THE FRONT-END PROCESSOR FUNCTION

As systems start to support higher data rates, one of the ever-present challenges is to minimize the interrupt processing load on the CPU, which can become intolerable if the CPU must process each character received in a separate interrupt. Since the character transfer task is typically so simple (reading a character from an input port and placing it into a memory buffer), it is often the case that the unavoidable context switch time associated with the interrupt outweighs the time spent processing the input character. In addition, the communication task may not be the CPU's highest priority: for example, in band-style laser printers the CPU must keep up with the paper movement; it can neither rerun an image nor stop the paper. The communication rate therefore suffers; even printers running from a Centronics-style parallel port are typically unable to accept data faster than 4k characters per second.

The traditional technique for overcoming this obstacle is to implement Direct Memory Access (DMA) for the communication ports. This is, however, quite a large investment in hardware, requiring an external DMA controller chip and more sophisticated bus structures to support it. In other words, it may be acceptable for a computer system, but it is overly expensive for an embedded controller application. Also, the response time required of the CPU can still be stringent, especially in implementing flow control to pace the character rate from the external system presenting the data.

The HPC46083 microcontroller, however, allows a much more cost-effective approach to the problem. As a peripheral, it interfaces to the CPU much as any peripheral controller would. In the application documented here, it buffers up to 128 characters before interrupting the CPU, thus dropping the CPU input interrupt processing frequency by over two orders of magnitude, while allowing a character input rate of over 20 kb/sec.

2.1 Data Transfer Technique

The benefit provided by a front-end processor is derived from the efficiency it adds to the process of getting data into the CPU's data buffer; that is, how much of the CPU's processing time gets dedicated to this task.

The efficiency is provided by two means:

1. Reduction of interrupt overhead. By interrupting the CPU only once every 100 characters, the overhead per character becomes virtually negligible.
2. Elimination of error testing overhead. If the CPU were communicating with a UART directly, it would have to poll for error conditions on each character. In our implementation, there are two interrupt vectors for data transfer: one for good data (which transfers a block of data), and one for bad data (which transfers one character and its error flags). The good data interrupt routine, then, which is invoked almost exclusively, contains a very simple inner loop. After reading the character count from the HPC, all that the CPU needs to do is:

— Move a character from the HPC's OBUF register to the current destination address. No time is wasted polling the HPC status; the hardware synchronization technique described in Application Note AN-550 handles this.

— Increment the destination address. (Checking against buffer limits could be done here, but is more efficiently handled outside the inner loop).

— Decrement the character count and test it; loop if nonzero.

The HPC firmware also supports this technique by guaranteeing that the reporting of character errors (and BREAK conditions) is synchronized with good data, so that the CPU can tell exactly where in the data stream the error occurred.

2.2 Logic Replacement

Front-end processing tasks by no means use up the HPC's capabilities in a system. In our application, the HPC also serves as the CPU's only interrupt controller, allowing a large number of vectors with no additional hardware. It performs additional control tasks such as dynamic RAM refresh request timing, front panel control and real-time clock functions given in Application Note AN-550 with inexpensive interfacing. In a single 4 kbyte program developed in our group, we were also able to add an interface to an inexpensive serial EEPROM device (connected directly to the MICROWIRE/PLUS™ port of the HPC) and to a laser-printer engine for non-imaging control functions, and we also implemented a higher-resolution event timing feature. (These are topics for future application notes, however, and are not dealt with here.)

To summarize, then, the HPC not only can provide front-end processing functions, but can pay for itself by replacing other logic in the system.

3.0 HARDWARE

The following sections refer to the schematic pages included. We will discuss here only the portions involving the Centronics Parallel and RS-232 Serial ports. See Application Note AN-550 for details of the other connections shown (the UPI port and front-panel functions).

3.1 The Centronics Parallel Port

The Centronics port was implemented on the connector designated J5. Most of the interface is diagrammed on Sheet 4 of the schematic.

3.1.1 Control Inputs

Pin 1 of the J5 connector receives the Data Strobe ($\overline{\text{STROBE}}$) input, which signals the presence of valid data from the external system. On Sheet 4, in area C5, this signal appears from the connector. It is filtered using a Schmitt trigger (a spare 1488 RS-232 receiver chip), and is then presented to the HPC (Sheet 3) as interrupt signal I4.

Pin 31 is the Input Prime signal ($\overline{\text{PRIME}}$), which is asserted low by the external system in order to reset the interface. It appears on Sheet 4 in area D5, and is filtered in a similar manner. It is then gated with the signal ENPRIME from the Centronics Control Latch, and the resulting signal is presented to the HPC on pin *EXUI, which is the External UART Interrupt input. The gating is used to prevent confusion between UART and $\overline{\text{PRIME}}$ interrupts: while the Centronics port is selected, only $\overline{\text{PRIME}}$ causes interrupts, and while the RS-232 port is selected, this gating keeps $\overline{\text{PRIME}}$ interrupts from being asserted.

3.1.2 Data Inputs

Eight data bits, from J5 pins 2 through 9, appear in areas B8 and C8 of Sheet 4. They are latched into a 74LS374 latch on the leading edge of the $\overline{\text{STROBE}}$ signal (note the inversion through the Schmitt receiver on $\overline{\text{STROBE}}$). The latch is enabled to present data to the HPC's Port D pins by the signal $\overline{\text{ENCDATA}}$, which comes from HPC pin B12. Note that Port D is also used for inputting pushbutton switch data from a front panel.

3.1.3 Control Outputs

The Centronics control and handshake signals are presented by loading the Centronics Control Latch (Sheet 4, area B4) from the HPC's pins A8 through A15 (Port A Upper) using as a strobe the signal CCTLCLK from HPC pin P2.

Pin 10 of connector J5 is the Centronics Acknowledge ($\overline{\text{CACK}}$) pulse, which is used to signal the external system that the HPC is ready for the next byte of data. This is one of the two handshake signals used to pace data flow. It is initialized high by the HPC, and is pulsed low when required.

Pin 11 is the Centronics Busy (CBUSY) signal, which is generated by the flip-flop on Sheet 4, area C3. It is set directly by a $\overline{\text{STROBE}}$ pulse, and is also loaded from the Centronics Control Latch whenever the HPC finishes reading a byte of data (rising edge of $\overline{\text{ENCDATA}}$). This will clear CBUSY under normal conditions, allowing the external system to send another byte of data.

Five additional signals, whose functions vary significantly from printer to printer, are presented on connector J5 from the Centronics Control Latch. These are:

Pin 13, which generally indicates that the printer is selected.

Pin 12, which indicates that the printer needs attention (for example, that it is out of paper).

Pin 32, which indicates a more permanent or unusual problem (lamp check or paper jam).

Pins 33 and 35, which vary more widely in use.

These five pins are manipulated by commands from the CPU; the HPC simply presents them as commanded.

3.1.4 Other Signals

Pin 18 of the Centronics port connector receives a permanent +5V signal (area B2 of Sheet 4), and a set of other pins (middle of Sheet 2) are connected permanently to ground.

3.2 The RS-232 Serial Port

The serial port (on connector J6) makes use of the HPC's on-chip UART and baud rate generator; very little off-chip hardware is required. The entire RS-232 circuit appears on Sheet 3 of the schematic.

This port is implemented in a way typical of printers, and so there are no sophisticated handshaking connections. The interface looks like an RS-232 DTE device: Connector J6 pin 2 is transmitted data (out) and pin 3 is received data (in).

The RS-232 data input appears in area B8 of Sheet 3, as signal RXD. After the RS-232 receiver, it is presented on the HPC's UART input pin (I6). Note that this pin can be monitored directly as a port bit; this enables the HPC to check periodically for the end of a BREAK condition without being subjected to a constant stream of interrupts for null characters.

The Data Set Ready signal (DSR) is received from pin 6 of J6, and presented on HPC pin I7, where it can be monitored by the HPC firmware.

The Request to Send signal (RTS) is a constant high level placed on J6 pin 4.

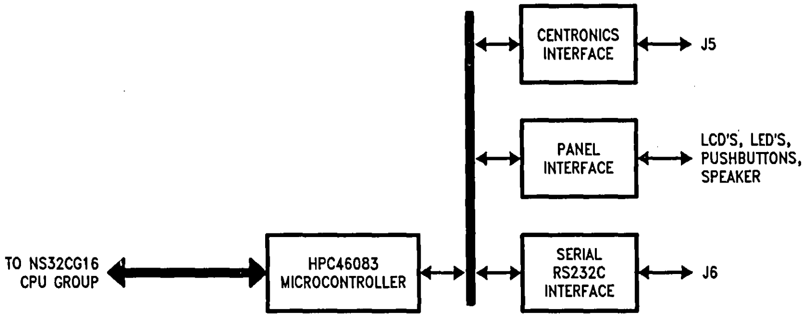
Transmitted data (TXD) is presented from the HPC's UART output pin (B0), through a buffering gate, to an RS-232 driver, and then out on J6 pin 3. The buffering gate would be unnecessary if the CMOS 14C88 driver were being used, but the gate was a spare and allowed cost savings using the less expensive TTL 1488 chip.

Data Terminal Ready (DTR) is simply presented from a programmable port pin of the HPC (pin B1). It is buffered through a spare inverter, and then presented to RS-232 connector J6 pin 20 through an RS-232 driver. As with the UART output, the buffering would be unnecessary with the 14C88 type of RS-232 driver; however, note that the HPC firmware would have to be modified slightly due to the resulting polarity difference on the pin.

J6 pins 1 (Frame Ground) and 7 (Signal Ground) are, of course, grounded, as shown in this sheet also.

3.3 Schematic Sheets

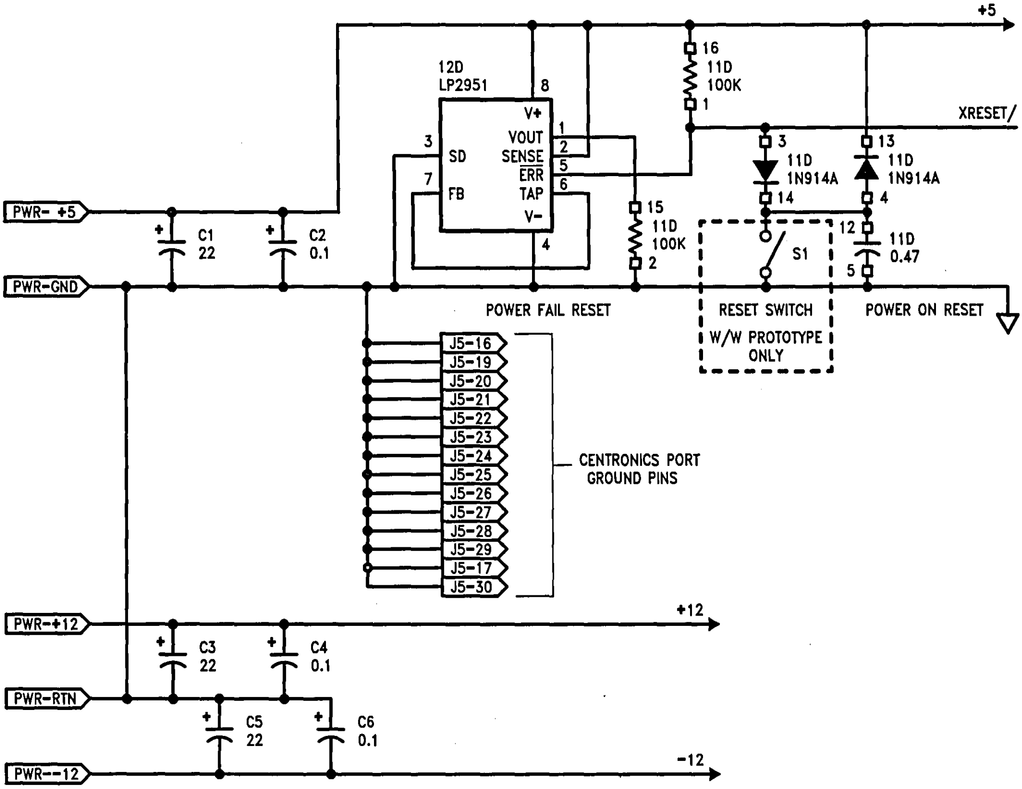
Sheet 1



TL/DD/9977-1

Power and Ground Distribution

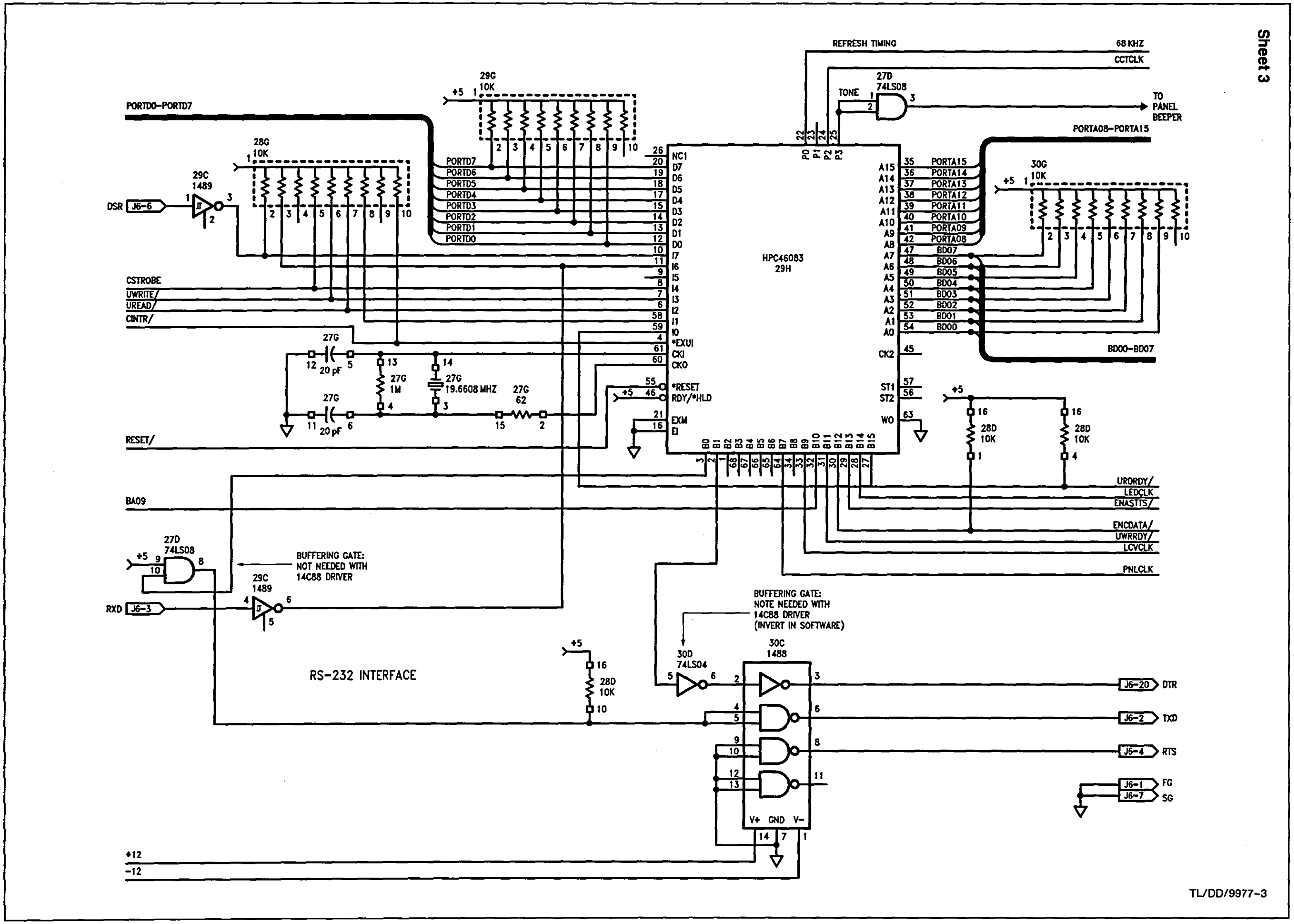
Sheet 2



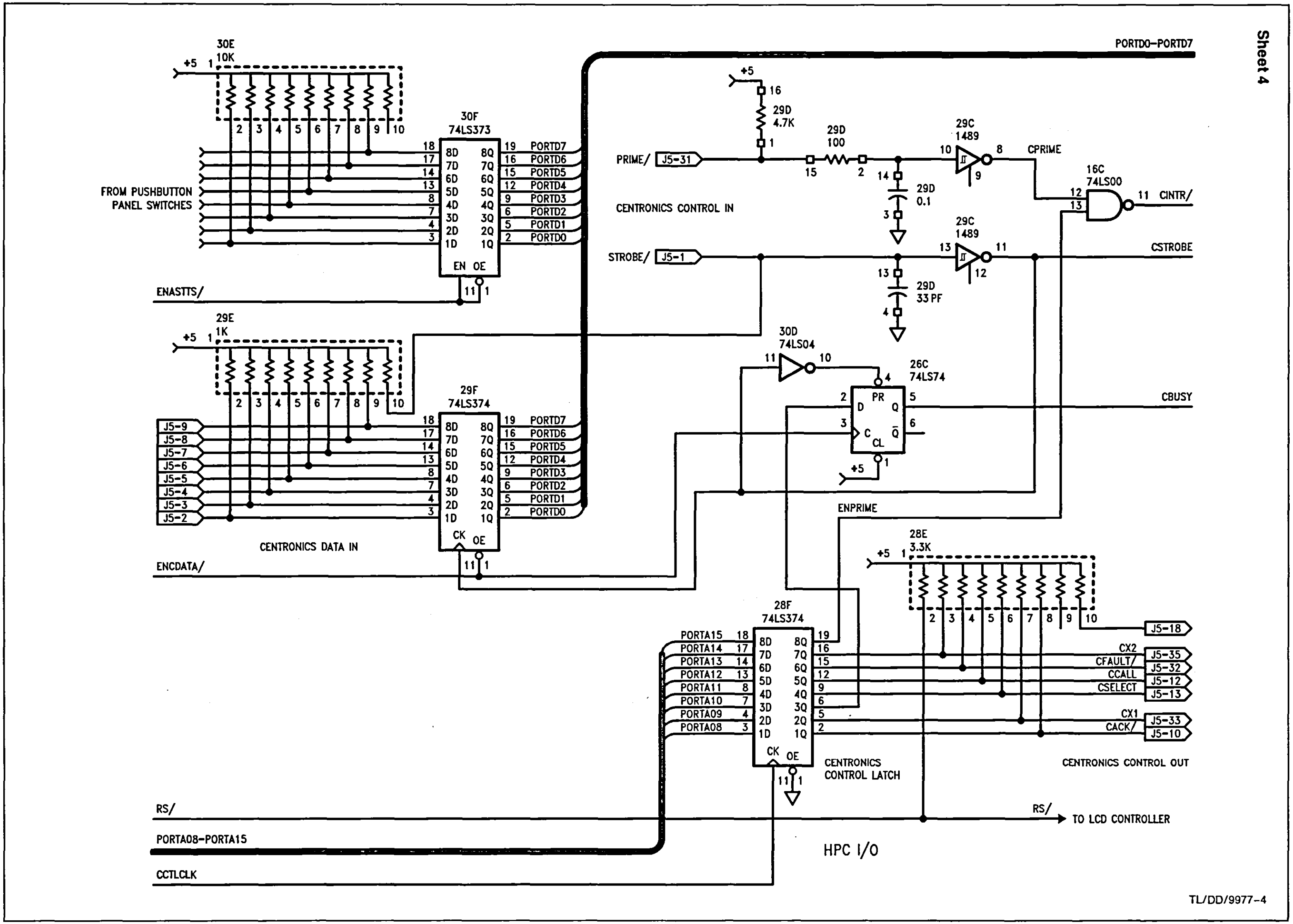
TL/DD/9977-2

Notes: (Unless otherwise specified)

1. All capacitance values in microfards, 50V.
2. All resistor values in Ohms, ¼W, 5%.


Sheet 3
REFRESH TIMING
68 KHZ
CCTCLK
27D
74LS08
TONE
TO
PANEL
BEEPER
PORTA08–PORTA15
PORTD0–PORTD7
29G
10K
+5
28G
10K
29C
1489
DSR
J6–6
PORTD7
PORTD6
PORTD5
PORTD4
PORTD3
PORTD2
PORTD1
PORTD0
NC1
D7
D6
D5
D4
D3
D2
D1
D0
I7
I6
I5
I4
I3
I2
I1
I0
*EXUI
CKI
CKO
*RESET
RDY/*HLD
EXM
EI
P0
P1
P2
P3
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
CK2
ST1
ST2
WO
B0
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
B13
B14
B15
HPC46083
29H
PORTA15
PORTA14
PORTA13
PORTA12
PORTA11
PORTA10
PORTA09
PORTA08
BD07
BD06
BD05
BD04
BD03
BD02
BD01
BD00
30G
10K
BD00–BD07
CSTROBE
UWRITE/
UREAD/
CINTR/
27G
20 pF
1M
19.6608 MHZ
62
RESET/
28D
10K
URDRDY/
LEDCLK
ENASTTS/
ENCDATA/
UWRRDY/
LCVCLK
PNLCLK
BA09
27D
74LS08
BUFFERING GATE:
NOT NEEDED WITH
14C88 DRIVER
29C
1489
RXD
J6–3
RS-232 INTERFACE
BUFFERING GATE:
NOTE NEEDED WITH
14C88 DRIVER
(INVERT IN SOFTWARE)
30D
74LS04
30C
1488
V+
GND
V−
J6–20
DTR
J6–2
TXD
J6–4
RTS
J6–1
FG
J6–7
SG
+12
−12

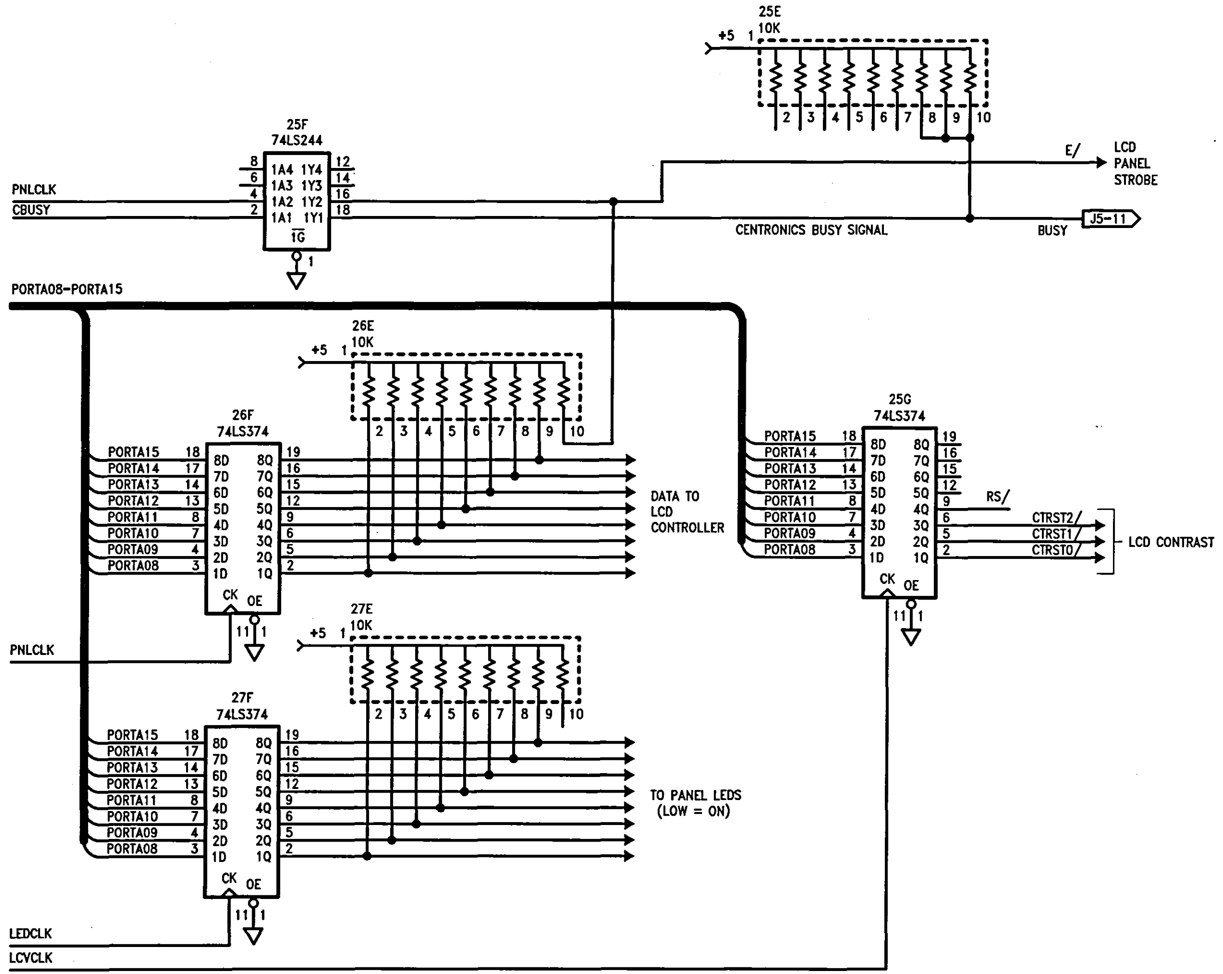
TL/DD/9977-3



HPC I/O

TL/DD/9977-4

Sheet 5



TL/DD/9977-5

4.0 PROTOCOL

The command and interrupt protocol is a superset of that implemented for Application Note AN-550. The two commands SELECT-CENT and SELECT-UART are added to select and initialize each of the communication ports (Centronics or RS-232). The CPU can exercise control over data buffering by the commands FLUSH-BUF, CPU-BUSY, CPU-NOT-BUSY and SET-IFC-BUSY. It can set Centronics port error flags and status using SET-CENT-STS, and it can test for RS-232 status using the TEST-UART command. The HPC also allows the CPU to send characters out on the RS-232 port using the SEND-UART command.

New interrupts presented by the HPC are !DATA, which transfers up to 128 bytes of buffered data to the CPU, !PRIME and !UART-STATUS, which inform the CPU of port status changes, and !DATA-ERR, which reports in detail any error ocurring in characters received. The interrupt !ACK-UART is presented to the CPU to acknowledge that the SEND-UART command has been completed.

Note that the command codes for the front panel functions have been changed. Their formats, however, have not changed, nor have their functions, except that the INITIALIZE command now performs a disconnection function on the RS-232 and Centronics ports.

4.1 Commands

The first byte (command code) is sent to address FFFC00, and any argument bytes are then written to address FFFE00. The CPU may poll the UPIC register at address FD0000 to determine when the HPC can receive the next byte, or it can simply attempt to write, in which case it will be held in Wait states until the HPC can receive it. Except where noted, the CPU may send commands continuously without waiting for acknowledgement interrupts from previous commands.

00 INITIALIZE — This command has two functions. The first INITIALIZE command after a hardware reset (or RESET-HPC command) enables the !RTC and !BUTTON-DATA interrupts. Both data communcation ports are set to their "Busy" states until a "SELECT" command is sent. The INITIALIZE command may be re-issued by the CPU to de-select both communication ports, and to either start or stop the !RTC interrupts. There is one argument:

RTC-Interval: One-byte value. If zero, !RTC interrupts are disabled. Otherwise, the !RTC interrupts occur at the interval specified (in units of 10 ms per count).

01 SELECT-CENT — Select the Centronics port and set it ready, using the timing sequence specified by the supplied ACK-Mode argument. Data from the port is enabled, and the !PRIME interrupt is also enabled. Arguments:

ACK-Mode: one byte in the format:

x	x	x	x	x	L	Timing

where the Timing field is encoded as:

00 = BUSY falling edge occurs after $\overline{\text{ACK}}$ pulse.

01 = BUSY falling edge occurs during $\overline{\text{ACK}}$ pulse.

10 = BUSY falling edge occurs before $\overline{\text{ACK}}$ pulse.

and the L bit, when set, requests Line Mode. It suppresses the removal of BUSY and the occurrence of the $\overline{\text{ACK}}$ pulse when the buffer is passed to the CPU. To fully implement Line Mode, this mode should be used with Pass-Count = 1 and Stop-Count = 1, and the CPU must use the SET-CENT-STS command to acknowledge each character itself.

Pass-Count: Number of characters in buffer before the HPC passes them automatically to CPU. One byte.

Stop-Count: Number of characters in buffer before HPC tells the external system to stop. One byte.

Note that the buffer is a maximum of 128 bytes in length, in this implementation.

Requires INITIALIZE command first.

02 SELECT-UART — Select Serial port and set it ready, according to supplied arguments. Requires INITIALIZE command first. Arguments are:

Baud: Baud rate selection. One Byte containing.

0 = 300 baud

1 = 600 baud

2 = 1200 baud

3 = 2400 baud

4 = 4800 baud

5 = 9600 baud

6 = 19200 baud

7 = 38400 baud

8 = 76800 baud

Frame: One byte, selecting character length, parity and number of stop bits.

Value	Data Bits	Parity	Stop Bits
0	8	Odd	1
1	8	Even	1
2	8	None	1
3	8	None	2
4	7	Odd	1
5	7	Even	1
6	7	Odd	2
7	7	Even	2

Flow: One byte, bit-encoded for handshaking and flow control modes:

0	0	0	0	XON	DTR		DSR
7	6	5	4	3	2	1	0

DSR: 1 = the HPC disables the UART receiver while the DSR input is inactive.

DTR: Polarity of DTR output, and whether it is used as a flow-control handshake.

00 = Permanently low (negative voltage).

01 = Permanently high (positive voltage).

10 = Handshaking: low means ready.

11 = Handshaking: high means ready.

XON: 1 = the HPC performs XON/XOFF flow control.

Pass-Count: Number of characters in buffer before the HPC passes them automatically to CPU. One byte.

Stop-Count: Number of characters in buffer before HPC tells the external system to stop. One byte.

Note that the buffer is a maximum of 128 bytes in length, in this implementation.

Requires INITIALIZE command first.

03 (reserved)

04 FLUSH-BUF No arguments. Flush HPC data communication buffer to CPU. Any data in the buffer is immediately sent to the CPU (using the !DATA interrupt). This command triggers the !DATA interrupt only if the buffer contains at least one byte. Requires INITIALIZE command and SELECT command first.

05 CPU-BUSY No arguments. Indicates that the CPU cannot accept any more data (the CPU's data buffer is full). This suppresses the !DATA and !DATA-ERR interrupts. Requires INITIALIZE command and SELECT command first.

06 CPU-NOT-BUSY No arguments. This undoes a previous CPU-BUSY command, and indicates that the CPU can now accept more data from the HPC. Requires INITIALIZE command and SELECT command first.

07 SET-IFC-BUSY "Set Interface Busy". No arguments. Commands the HPC to immediately signal the external system to stop sending characters. This status is removed only by performing a SELECT command. Requires INITIALIZE command and SELECT command first.

08 SET-CENT-STS "Set Centronics Port Status". Loads Centronics latch from the supplied argument byte. Argument is eight bits, which must be encoded as follows:

ENPRIME	CX2	$\overline{\text{FAULT}}$	CALL	SELECT	BUSY	CX1	$\overline{\text{ACK}}$

The $\overline{\text{ACK}}$ bit should always be a "1". The CPU must use the BUSY bit to generate an $\overline{\text{ACK}}$ pulse: if the BUSY bit is zero, the $\overline{\text{ACK}}$ signal will be automatically pulsed low, then high, (regardless of the previous states of BUSY and $\overline{\text{ACK}}$).

Requires INITIALIZE command and SELECT-CENT command first.

09 SET-CONTRAST The single argument is a 3-bit number specifying a contrast level for the LCD panel (0 is least contrast, 7 is highest contrast). There is no response interrupt. Does not require INITIALIZE command first.

0A SEND-LCD This writes a string of up to 8 bytes to the LCD panel. Arguments are:

flags: A single byte, containing the RS bit associated with each byte of data. The first byte's RS value is in the least-significant bit of the FLAGS byte.

#bytes: The number of bytes to be written to the LCD display.

byte[1]-byte[#bytes]: The data bytes themselves.

The HPC determines the proper delay timing required for command bytes (RS = 0) from their encodings. This is either 4.9 ms or 120 μs.

The response from the HPC is the !ACK-SEND-LCD interrupt, and this command must not be repeated until the interrupt is received. This command does not require an INITIALIZE command first.

0B SEND-LED The singe argument is a byte containing a "1" in each position for which an LED should be lit.

There is no response interrupt, and this command does not require the INITIALIZE command first.

0C BEEP No arguments. This beeps the panel for approximately one second. No response interrupt. If a new BEEP command is issued during the beep, no error occurs (the buzzer tone is extended to one second beyond the most recent command). Does not require INITIALIZE command first.

0D SEND-UART The single one-byte argument is sent on the UART port. An acknowledgement interrupt !ACK-UART occurs on completion. This command must not be repeated until the interrupt is received. Requires INITIALIZE and SELECT-UART commands first.

0E TEST-UART Triggers a !UART-STATUS interrupt. This command must not be repeated until the interrupt is received. No arguments. Requires INITIALIZE and SELECT-UART commands first.

A5 RESET-HPC Resets the HPC if it is written to address FFFC00. It may be written at any time that the UPI port is ready for input; it will automatically cancel any partially-entered command. The CPU's Maskable Interrupt must be disabled before issuing this command.

After issuing this command, the CPU should first poll the UPIC register at address FD0000 to see that the HPC has input the command (the least-significant bit [Write Ready] is zero). It must then wait for at least 25 μs, then read a byte from address FFFE00. The HPC now begins its internal re-initialization. The CPU must wait for at least 80 μs to allow the HPC to re-initialize the UPI port. Since part of the RESET procedure causes Ports A and B to float briefly (this includes the CPU's Maskable Interrupt input pin), the CPU should keep its maskable interrupt disabled during this time. It also must not enter a command byte during this time because the byte may be lost.

4.2 Interrupts

The HPC interrupts the CPU, and provides the following values as the interrupt vectors for the CPU hardware. The CPU then reads data from the HPC at address FFFE00. All data provided by the HPC must be read by the CPU before returning from the interrupt service routine, otherwise the HPC would either hang or generate a false interrupt. The CPU may poll the UPIC register at address FD0000 to determine when each data byte is ready, or it may simply attempt to read from address FFFE00, and it will be held in Wait states until the data is provided by the HPC.

Note: All CPU interrupt service routines, including the NMI interrupt routines, must return using the "RETT 0" instruction. Do NOT use "RETI".

Vector

00–0F (none) (Reserved for CPU internal traps and the NMI interrupt.)

10 !DATA Buffer data is being transferred to CPU. This will happen either automatically, at a point defined by the most recent SELECT command, or as the result of a FLUSH-BUF command. It is followed by a one-byte Length (number of characters: current HPC firmware has a range of 1–128), then that number of characters. Enabled by SELECT command after at least one INITIALIZE command.

11 !RTC Real-Time Clock Interrupt. No data returned. Enabled by INITIALIZE command if interval value supplied is non-zero.

Note: This version of HPC firmware issues a non-fatal !DIAG interrupt if the CPU fails to service each !RTC interrupt before the next one becomes pending.

12 (reserved)

13 !PRIME Centronics INPUT PRIME signal has become active. No data returned. Enabled by SELECT-CENT command after at least one INITIALIZE command.

14 (reserved)

15 (reserved)

16 (reserved)

17 !ACK-SEND-LCD This is the response to the SEND-LCD command, to acknowledge that data has all been written to Panel LCD display. No other data is provided with this interrupt. Always enabled, but occurs only in response to a SEND-LCD command.

18 !BUTTON-DATA Pushbutton status has changed: one or more buttons have been either pressed or released. The new status of the switches is reported in a data byte, encoded as follows:

Any pushbutton that is depressed is presented as a "1". All other bit positions, including unused positions, are zeroes. The pushbuttons are debounced before being reported to the CPU. This interrupt is enabled by the first INITIALIZE command after a reset.

19 !UART-STATUS

UART status has changed. This interrupt occurs only while the UART is selected. A data byte shows the UART's new state:

Bit	**Condition**
0 (LSB)	New state of DSR signal. This causes an interrupt only if DSR monitoring was requested in the last SELECT-UART command. The UART receiver is automatically enabled and disabled by the HPC, so no CPU action is required on receiving this interrupt. If a SELECT-UART command is entered, requesting DSR monitoring, and DSR is inactive, a !UART-STATUS interrupt occurs immediately.
1	This bit is set if a UART BREAK has just ended.
2–7	(unused)

Note 1: If the CPU has issued a CPU-NOT-READY command, this BREAK interrupt may be seen before the !DATA-ERR interrupt that announces the start of the BREAK (and its position in the data stream).

Note 2: The DSR and UART input (BREAK) signals are sampled every 10 ms.

1A !DATA-ERR

An error has been encountered in data coming from the currently-selected communication port. It is enabled by the first SELECT command after the first INITIALIZE command. Two data bytes are returned:

errchr: One byte containing the character on which the error was seen (this character is NOT placed in the data buffer).

errfgs: Error flags, detailing the error seen:

Bit	**Error Seen**
0 (LSB)	(unassigned)
1	(unassigned)
2	UART BREAK condition detected. This may be preceded by one or two framing errors.
3	**Error Overflow:** More errors occurred than HPC could report (the HPC has no FIFO for error reporting).
4	**Buffer Overflow:** Flow control failed to stop the external system, and the buffer overflowed.
5	**Parity Error:** Serial Port only.
6	**Framing Error:** Serial Port only.
7 (MSB)	**Data Overrun:** Serial Port only.

If bit 2, 3 or 4 is set, the communication port has been automatically shut down by the HPC. The CPU must issue a new SELECT command to re-enable the port.

When a character is received with an error, all characters appearing before it in the buffer are automatically flushed before this interrupt occurs. This is done to preserve the error character's position in the data stream. If the CPU decides to ignore the presence of an error, the character may be simply appended by the CPU to the data already in its data buffer. Please note: If the CPU has issued a CPU-NOT-READY command, the flush cannot occur, and this interrupt will not be issued until the flush has occurred.

1B !ACK-UART

A CPU character has been sent on the UART, and the UART is ready for another. No data is returned with this interrupt. It is always enabled, but occurs only in response to the SEND-UART command.

1C (reserved)

1D !DIAG

Diagnostic Interrupt. This interrupt is used to report failure conditions and CPU command errors. There are five data bytes passed by this interrupt:

Severity
Error Code
Data in Error (passed, but contents not defined)
Current Command (passed, but contents not defined)
Command Status (passed, but contents not defined)

The Severity byte contains one bit for each severity level, as follows:

X	X	X	F	X	X	C	N

N (Note): least severe. The CPU missed an event; currently only the !RTC interrupt will cause this.

C (Command): medium severity. Not currently implemented. Any command error is now treated as a FATAL error (below).

F (Fatal): highest severity. The HPC has recognized a non-recoverable error. It must be reset before the CPU may re-enable its Maskable Interrupt. In this case, the remaining data bytes may be read by the CPU, but they will all contain the value 1D (hexadecimal). The CPU must issue a RESET command, or wait for a hardware reset. See below for the procedure for FATAL error recovery.

The Error Code byte contains, for non-FATAL errors, a more specific indication of the error condition:

RTC	(Reserved for COMMAND)

RTC = Real-Time Clock overrun: CPU did not acknowledge the RTC interrupt before two had occurred.

The other bits are reserved for details of Command errors, and are not implemented at this time.

The remaining 3 bytes are not yet defined, but are intended to provide details of the HPC's status when an illegal command is received.

Note: Except in the FATAL case, all 5 bytes provided by the HPC *must* be read by the CPU, regardless of the specific cause of the error.

Fatal Error Recovery:

When the HPC signals a !DIAG error with FATAL severity, the CPU may use the following procedure to recover:

1. Write the RESET command (A5 hex) to the HPC at address FFFC00.
2. By inspecting the UPIC register at address FD0000, wait for the HPC to read the command (the $\overline{\text{WRRDY}}$ bit will go low).
3. Wait an additional 25 μs.
4. Read from address FFFE00. This will clear the OBUF register and reset the Read Ready status of the UPI port. The HPC will guarantee that a byte of data is present; it is not necessary to poll the UPIC register. This step is necessary because only a hardware reset will clear the Read Ready indication otherwise (HPC firmware cannot clear it).
5. Wait at least 80 μs. This gives the HPC enough time to re-initialize the UPI port.
6. After Step 5 has been completed, the CPU may re-enable the Maskable Interrupt and start issuing commands. Since the HPC is still performing initialization, however, the first command may sit in the UPI IBUF register or a few milliseconds before the HPC starts to process it.

5.0 SOURCE LISTINGS AND COMMENTARY

5.1 HPC Firmware Guide

This section is intended to provide help in following the flow of the HPC firmware. Discussion of features already documented in Application Note AN-550 are abbreviated here; see that application note for details.

The firmware for the HPC is almost completely interrupt-driven. The main program's role is to poll mailboxes that are maintained by the interrupt service routines, and to send an interrupt to the CPU whenever a HPC interrupt routine requests one in its mailbox.

On reset, the HPC firmware begins at the label "start". However, the first routine appearing in ROM is the Fatal Error routine. This is done for ease of breakpointing, to keep this routine at a constant address as changes are made elsewhere in the firmware.

5.1.1 Fatal Error Routine

At the beginning of the ROM is a routine (label "hangup") that is called when a fatal error is detected by the HPC. This routine is identical to that documented in Application Note AN-550.

5.1.2 Initialization

At label "start", entered on a Reset signal or by the RESET-HPC command from the CPU, the HPC begins its internal initialization. It loads the PSW register (to select 1 Wait state), and then (at label "srfsh"), it starts the Refresh clock pulses running for the dynamic RAM by initializing Timer T4 and starting it.

At "supi", the UPI port is initialized for transfers between the HPC and the CPU.

At label "sram", all RAM within the HPC is initialized to zero.

At "sskint", the stack pointer is initialized to point to the upper bank of on-chip RAM (at address 01C0). The address of the fatal error routine "hangup" is then pushed, so that it will be called if the stack underflows.

At "tminit", the timers T1–T3 are stopped and any interrupts pending from timers T0–T3 are cleared. This step arbitrarily initializes the UART baud rate to 9600, but this selection has no effect.

At "scent", the Centronics port is initialized and set up to appear busy to the external system.

At "suart", the HPC UART is initialized for serial data from the external system. The RS-232 DTR signal is arbitrarily set low, which generally means that the printer is not ready. The state of DTR is not actually valid until the first SELECT-UART command is received, which selects the handshaking mode.

At "sled", the LED control signals are initialized,and all LED indicators are turned off.

At "stmrs", all timers are loaded with their initial values, and timers T5–T7 are stopped and any interrupts pending from them are cleared.

At "slcd", the LCD display is initialized to a default contrast level of 5, then commands are sent to initialize it to 8-bit, 2-line mode, with the cursor visible and moving to the right by default. This section calls a subroutine "wrpnl" for each character; the subroutine simply writes the character in the accumulator out to the LCD display and waits for approximately 10 ms.

The program then continues to label "minit", which initializes the variables in the HPC's on-chip RAM to their proper contents.

At label "runsys", the necessary interrupts are enabled (from the timers, and from pin I3, which is the UPI port interrupt from the CPU), and the program exits to the Main Program at label "mainlp". Interrupts from the Centronics and UART ports are not enabled until the appropriate SELECT command is received.

5.1.3 Main Program (UPI Port Output to CPU)

The Main Program is the portion of the HPC firmware that runs with interrupts enabled. It consists of a scanning loop at label "mainlp" and a set of subroutines (explained below). It is responsible for interrupting the CPU and passing data to it; the HPC is allowed to write data to the CPU only after interrupting it. Unlike the simpler UPI/Front Panel interface described in Application Note AN-550, this main loop scans two separate variables in on-chip RAM that are set up by interrupt service routines: a word called "alert", and a byte called "bstat" (for "Buffer Status"). Both variables are used to determine whether any conditions exist that should cause an interrupt to the CPU.

The "alert" word contains one bit for each interrupt that the HPC can generate. If a bit is set (by an interrupt service routine), the Main Program jumps to an appropriate subroutine to notify the CPU. The subroutine checks whether the UPI interface's OBUF register is empty, and if not, it waits (by calling the subroutine "rdwait"). It then writes the vector number to the OBUF register. This has the effect of interrupting the CPU (because the pin $\overline{\text{URDRDY}}$ goes low), and the CPU hardware reads the vector from the OBUF register.

If there is more information to give to the CPU, the HPC places it, one byte at a time, into the OBUF register, waiting each time for OBUF to be emptied by the CPU. This technique assumes that the CPU remains in the interrupt service routine until all data has been transferred: if the CPU were to return from interrupt service too early, the next byte of data given to it would cause another interrupt, with an incorrect vector.

(Note, however, that the CPU may be interrupted with a Non-Maskable interrupt from a separate source. This simply inserts a pause into the process of reading data from the HPC. Since the HPC is running its main program at this point, with interrupts still enabled, it will not lose data from its communication port under these circumstances.)

The "bstat" byte represents a special case involving the interrupt !DATA to the CPU. This byte shows the main program whether the data communication buffer (which holds data from the external system) is full enough to send its contents to the CPU. If so, the main program calls the subroutine "snddta", which interrupts the CPU, then sends one data byte containing the number of characters to be transferred (currently as many as 128 are possible), and then the characters themselves.

The CPU may, at any time, demand that the HPC transfer all characters that are within its communication buffer. (This is called a "flush" command, which sets one of the bits of the "alert" word, described above.) The HPC, in response, will empty the buffer to the CPU with a !DATA interrupt, even if only one character is left. If the buffer is completely empty, however, the flush command is ignored.

Subroutines called from the Main Program loop are:

sndrtc: sends a Real-Time Clock interrupt to the CPU. No data is transferred; only the interrupt vector.

sndlak: interrupts the CPU to acknowledge that a string of data (from a SEND-LCD command) has been written to the LCD display. No data is transferred for this interrupt.

sndbtn: interrupts the CPU to inform it that a pushbutton has been pressed or released. A data byte is transferred from variable "swlsnt", which shows the new states of all the pushbuttons.

sndfsh: performs a Flush operation. If there is data, it jumps to the "snddta" routine to send the contents of the buffer to the CPU. If there is no data, however, this subroutine simply returns without generating an interrupt.

snddta: sends data from the communication buffer to the CPU. It may be entered for one of three reasons:

1. the communication buffer is full enough that it must be sent automatically to the CPU.
2. a Flush command has been received from the CPU. (The bit "aflush" in the ALERT word is set.)
3. an error has been detected on a character received from the external system. This causes an internal Flush request, so that all good characters are sent to the CPU before the bad character is reported. This case is also different because it does not flush the entire buffer, but only up to the point of the error. The limit is held in the variable "fshlim".

The subroutine sends a "length" byte (from variable "numout", sampled from "numchr", which is maintained by the communication interrupt routines). This indicates how many characters will be transferred. The subroutine next sends the characters themselves. It then updates the buffer status variables in on-chip RAM, to indicate how many characters were removed.

Depending on other status of the selected communication port, this subroutine may re-enable communication on the port if it was stopped (for example, if the buffer was too full to accept more data until the "snddta" routine emptied it). This is done at label "sdstp".

sndprm: interrupts the CPU because the INPUT PRIME signal on the Centronics parallel port was activated by the external system. No data is transferred by this interrupt.

sndust: interrupts the CPU to report a change in UART status. This interrupt may also be triggered by the CPU using the TEST-UART command.

snderr: interrupts the CPU to inform it that a character with an error was received. The character and a byte containing error flags are transferred to the CPU.

snduak: interrupts the CPU in response to a SEND-UART command, to acknowledge that the requested character has been sent on the UART transmitter, and that it is ready to transmit another character.

sndiag: interrupts the CPU to inform it of a !DIAG interrupt condition, when it is of NOTE severity. (Other !DIAG conditions are handled at label "hangup".)

5.1.4 UPI Port Input from CPU (Interrupt I3)

This interrupt service routine, at label "upiwr", accepts commands from the CPU. Apart from the existence of additional commands, the structure of this routine is identical to that of Application Note AN-550. We document here the labels and functions involved in this larger application.

Command Processing Routines

INITIALIZE I3 interrupt labels: State 1 = fcinit State 3 = lcinit

SELECT-CENT I3 interrupt labels: State 1 = fcselc State 3 = lcselc

SELECT-UART I3 interrupt labels: State 1 = fcselu State 3 = lcselu

FLUSH-BUF I3 interrupt labels: State 1 = fcflsh State 3 = (none)

At label "fcflsh", the "alert" word bit "aflush" is set, which requests that the main program flush the communication buffer.

CPU-BUSY I3 interrupt labels: State 1 = fccbsy State 3 = (none)

At label "fccbsy", the buffer status byte "bstat" is set to indicate that the CPU is busy and cannot accept more data from the HPC. This disables the !DATA interrupt.

CPU-NOT-BUSY I3 interrupt labels: State 1 = fccnby State 3 = (none)

At label "fccnby", the buffer status byte "bstat" is set to indicate that the CPU is ready to accept more data from the HPC. The !DATA interrupt is re-enabled.

SET-IFC-BUSY I3 interrupt labels: State 1 = fcifby State 3 = (none)

At label "fcifby", the currently selected interface is set busy, in order to present an error indication.

SET-CENT-STS I3 interrupt labels: State 1 = fcscst State 3 = lcscst

At label "lcscst", the Centronics Port status byte "cps" is loaded from the value supplied by the CPU, and the Centronics port control signals are updated to reflect these new settings. The subroutine "setcen" is used to set up the control signals, and it also pulses the Centronics $\overline{\text{ACK}}$ signal if appropriate.

SET-CONTRAST I3 interrupt labels: State 1 = fcslcv State 3 = lcslcv

At label "lcslcv" (Set LCD Voltage), the LCD Contrast latch is loaded from the value supplied by the CPU.

SEND-LCD I3 interrupt labels: State 1 = fcslcd State 3 = lcslcd

This command sends a string of up to eight bytes to the LCD display. Application Note AN-550 describes the implementation of this command in detail.

SEND-LED I3 interrupt labels: State 1 = fcsled State 3 = lcsled

At label "lcslcd", the byte provided by the CPU is written to the LED latch.

BEEP I3 interrupt labels: State 1 = fcbeep State 3 = (none)

This command sends a one-second beep tone to a speaker.

SEND-UART I3 interrupt labels: State 1 = fcsndu State 3 = lcsndu

At label "lcsndu", the single argument (the character to be sent) is placed in variable "uschr", and the bit "schr" is set in variable "ups" (UART Port Status). By doing this, the character has been queued for transmission. The transmission is performed by the subroutine at label "setuar", which is also responsible for performing the XON/XOFF flow control protocol. If a character is already being sent (the transmitter interrupt is enabled), then this is the only action required, since the transmitter interrupt automatically invokes the "setuar" subroutine. However, if the transmitter is idle, this routine must itself call "setuar" to transmit the character.

The subroutine "setuar" itself calls another subroutine at label "uecsnd", which formats the character to be transmitted into the frame selected by the current UART framing mode. It then sends the character. Note that the UART framing mode applies to output as well as input characters.

TEST-UART I3 interrupt labels: State 1 = fcusts State 3 = (none)

At label "fcusts", the HPC sets the "austat" bit of the ALERT word, requesting the Main Program to send a !UART-STATUS interrupt to the CPU.

5.1.5 Centronics Commmunication

This task is triggered by each edge of the Centronics port $\overline{\text{STROBE}}$ signal. This signal is detected by the HPC on the I4 interrupt line. On the leading edge of $\overline{\text{STROBE}}$, the character is input to the data communication buffer. This edge also sets the BUSY signal, by hardware action. On the trailing edge, the BUSY flag is affected by the HPC firmware. If the HPC is ready to receive more characters, the BUSY signal is cleared and the $\overline{\text{ACK}}$ signal is pulsed. If the HPC is not ready to receive more data, it leaves the BUSY signal high, which prevents the external system from sending more characters.

The Centronics port $\overline{\text{STROBE}}$ handler is at label "cenint". It first determines whether a falling or rising edge was detected on the $\overline{\text{STROBE}}$ signal. If the leading (falling) edge was detected, then it jumps to label "cstrbl"; otherwise it jumps to label "cstrbt" to process a trailing edge.

At label "cstrbl", the character is placed in the next available position of the communication buffer, if the buffer is not already full. (If it is already full, then it is processed as an error, as discussed below.) Then some tests are performed:

If the buffer is not full enough to pass data to the CPU, then the routine exits by jumping to label "cenlex", where it prepares to detect the trailing edge of $\overline{\text{STROBE}}$. Otherwise, it sets the "pass" bit in the variable "bstat", which requests the main program to send data to the CPU, and then it continues.

If the buffer is not full enough to tell the external system to stop sending characters, then the routine exits by jumping to "cenlex". Otherwise, it sets the "stop" bit in variable "bstat", indicating that the external system has been stopped, and it also sets the "cbusy" flag in variable "cps", which will prevent the Centronics BUSY and $\overline{\text{ACK}}$ signals from being changed when the $\overline{\text{STROBE}}$ pulse ends. The routine continues.

If the buffer has become completely full, then the "full" bit in "bstat" is set, indicating that any more characters received will trigger an error. Character processing then continues at label "cenlex".

At "cenlex", the Centronics Control Latch is set (temporarily) to force the BUSY signal high, because it should not become low until the $\overline{\text{STROBE}}$ pulse ends. The I4 pin, which detects the STROBE signal, is then re-programmed to detect the trailing edge (rising edge at the Centronics connector, but falling edge at pin I4 due to an inverting buffer). If the trailing edge already has occurred, then this reprogramming will set another interrupt pending immediately. There is, however, a possibility that the strobe edge could occur simultaneously with the reprogramming, with unknown results. For this reason, the STROBE signal is sampled by the firmware, and if the pulse has already completed, then instead of returning from the interrupt it jumps immediately to interrupt routine "cstrbt", which processes the trailing edge.

The code at label "cstrbt" is entered whenever either a trailing edge interrupt is detected on pin I4 (STROBE), or the leading edge interrupt routine jumps to it. It reprograms the I4 pin to detect a leading edge again, clears the I4 interrupt (which is automatically cleared only on interrupt service), then jumps to the "setcen" subroutine, which manipulates the BUSY and $\overline{\text{ACK}}$ signals appropriately, according to the contents of the "cps" variable and the selected $\overline{\text{ACK}}$ timing mode in variable "ackmd".

5.1.5.1 Centronics Error Handling

A buffer overrun error is processed at label "cenerr". This is the only kind of character error that can happen on a Centronics interface, and it would be due to an incorrect connection or a software error.

For internal firmware debugging purposes, the "cps" variable bit "cbusy" is again set to ensure that the Centronics interface will keep the BUSY signal set.

If an error is already waiting to be reported (bit "aerr" of variable "alert" is already set), then this is a "multiple error" condition, and cannot be fully reported. Instead, at label "cenmer", the bit "errovf" in variable "errfgs" is set. This variable is sent to the CPU when the error is reported. Also, the I4 interrupt is disabled, to prevent any further STROBE interrupts until a new SELECT-CENT command is received from the CPU.

If no error is waiting to be reported, then bit "aerr" of variable "alert" is set, requesting the main program to generate an !ERROR interrupt to the CPU. Further data is provided to be passed to the CPU:

variable "errfgs" is initialized to indicate only a buffer overrun error.

variable "errchr" is loaded with the character that was received and could not fit in the buffer.

Because the received character is reported with the error interrupt, and because no data is lost yet, the Centronics port is not disabled by this condition.

5.1.6 UART Communication

UART communication is performed by the UART interrupt routine at label "uarint". After pushing the required registers onto the stack, the routine determines which interface is selected. If it is the Centronics port, the only cause of the interrupt is the INPUT PRIME signal, and the HPC jumps to label "uarprm" (see Background Processing/Monitoring Tasks, below). If the UART port is selected, then it is due to either a receiver or a transmitter interrupt (and the INPUT PRIME is gated so that it cannot be presented).

5.1.6.1 UART Output

At label "uarout", a transmitter interrupt has been received.

If the bit "icpu" in variable "ups" is set, this means that the character just transmitted was a character sent by a CPU SEND-UART command, and the CPU is notified by requesting the !ACK-UART interrupt from the Main Program.

The subroutine "setuar" is now called, to determine whether any more characters need to be sent, either for XON/XOFF handshaking or because the CPU has requested the HPC to send another character. If so, another character is sent by "setuar", and the UART transmitter interrupt remains enabled. If not, the "setuar" routine disables the transmitter interrupt.

5.1.6.2 UART Input

At label "uartin", an interrupt has been generated by the UART receiver. This means that a character is available to be placed into the Communication Buffer.

The first action taken by the HPC is to read the receiver status register ENUR (which contains the 9th data bit and the Data Overrun and Framing Error error flags), then it reads the character itself from the RBUF register. The ENUR register is saved temporarily in variable "enrimg" for future processing, but is also held in the Accumulator, which is used here to "accumulate" error flags. The HPC then prepares to check for a parity error.

Parity checking is not a hardware feature of the HPC's UART, so a bit-table lookup is performed using the "X,[B].b" addressing mode of the IFBIT instruction. This addressing mode is similar to NS32000 bit addressing, in that it allows one to address up to 64 kbits (addressed from the contents of the X register) from a base address given in the B register. By placing the character to be checked into the X register, and pointing the B register at a properly constructed table (labels "evntbl" and "oddtbl"), a parity error can be detected in a single IFBIT instruction (see for example label "u8dopr").

After loading the X and B registers, a multi-way branch is performed (jid), which branches to one of 8 labels depending on the character framing mode variable "uframe" (which is loaded by the SELECT-UART command). Each mode handles parity differently: labels "uiod8" and "uiev8" check for odd or even parity, respectively, including 9 character bits (8 data plus 1 parity) to make the test. Labels "uiod7" and "uiev7" include only 8 bits (7 data plus 1 parity). Label "nopar" handles the cases where no parity is included in the character frame. Also within these routines, a decision is made whether a Framing Error seen in the character is also a Break condition: if two consecutive characters are seen with framing errors with all zeroes in their parity and data fields, then the second character is reported as a Break character as well as having a framing error. If, at label "uinpok", no errors have been flagged in the Accumulator, the routine branches to label "uingd" to place the character into the Data Communication Buffer for the CPU. If errors have been discovered, then the character is instead reported to the CPU using the !DATA-ERR at label "uinerc".

The "uingd" portion of this routine is very similar to the portion of the Centronics input routine that places characters into the buffer for the CPU. A different mechanism is used for flow control, of course, to stop the external system if the buffer becomes full.

At label "uinerc", a check is made to determine whether the CPU has received the last character error reported. If not, this is a "multiple error" condition, handled at label "uinmce". If so, then this is reported as a new error at label "uin1ce". The error character and its error flags are provided to the Main Program in the mailboxes "errchr" and "errfgs", and the bit "aerr" in variable "alert" is set to request that a !DATA-ERR interrupt be sent to the CPU.

On a multiple-error condition, the new error flags are ORed with the old ones, handshaking is used to stop the external host system from sending more characters, and the UART receiver is automatically disabled. The CPU must issue a new SELECT-UART command to re-enable it.

Another pair of routines report an error if the buffer overflows. This error is reported at label "uin1ef" if no other error report is pending, or at label "uinmef" if this is a multiple error condition. On a multiple error, an attempt is made to stop the external host system from sending characters, and the UART receiver is disabled until the CPU issues a SELECT-UART command. (A single error does not disable the receiver, because no data has been lost yet: the !DATA-ERR interrupt reports the character with the error report.)

5.1.7 Buffer Status Reporting

For internal debugging purposes, four unassigned signals from the LCD Contrast Latch are updated to show the status of the buffer. While the buffer is full enough to pass to the CPU, one bit of the latch (IC 25G, pin 12) is high. While the buffer is full enough that the external system should stop, pin 15 is high. While the CPU is not ready to receive data from the CPU, pin 16 is high. If a buffer overrun condition occurs, and data is lost, or if any fatal error occurs (with a hexadecimal code appearing on the LCD display), then pin 19 goes high. The code that handles these bits is flagged with the word "DEBUG" in the comment field.

5.1.8 Background Processing/Monitoring Tasks

These are tasks that are not triggered directly by CPU commands.

Real-Time Clock (T1) Timer T1 is loaded with a constant interval value which is used to interrupt the HPC at 10 ms intervals. When the Timer T1 interrupt occurs (labels "tmrint", "t1poll", "t1int"), and the real-time interrupt is enabled, the variable "rtccnt" is decremented to determine whether a !RTC interrupt should be issued to the CPU. If so, the bit "artc" in the "alert" word is set, requesting the main program to send a !RTC interrupt to the CPU. The main program, at label "sndrtc", interrupts the CPU. No other data is passed to the CPU.

At label "kbdchk" the panel pushbutton switches are also sampled. This process is described fully in Application Note AN-550.

At label "dsrchk", the state of the UART DSR flag is checked if the UART is selected and DSR monitoring mode has been requested by the CPU. If it has changed, this routine requests the Main Program to issue a !UART-STATUS interrupt to the CPU. The UART receiver is also enabled and disabled by the state of this signal if DSR monitoring has been requested. (The CPU does not have to react to the interrupt for normal operation, but might wish to record its occurrence.)

At label "brkchk", if the UART is selected, and a BREAK has been detected, the UART data input pin is polled to determine whether the BREAK condition has ended. If a BREAK has ended, then this routine requests the Main Program to issue a !UART-STATUS interrupt to the CPU.

Centronics INPUT PRIME When the $\overline{\text{EXUI}}$ pin on the HPC is activated, and the Centronics port is selected rather than the UART, the UART service routine (at label "uarprm") sets bit "aprime" in the "alert" variable, requesting the main program to send a !PRIME interrupt to the CPU. The Centronics port is internally flagged (in the "cps" variable) as being "busy", and the Centronics Control Latch is updated to set the BUSY signal high. The UART interrupt is then disabled until a SELECT-CENT command is received from the CPU. In the main program, the !PRIME interrupt is sent to the CPU at label "sndprm". No other data is sent.

5.2 HPC Firmware Listing

```
        # Centronics Port input / checksum calculation / LCD output.
        #       Accepts up to 1024 characters on Centronics port,
        #       accumulates 8-bit checksum, and on receiving Ctrl-D,
        #       displays checksum on LCD display.

        .globl  start,main
        .globl  dataint,rtcint,primeint
        .globl  lcdint
        .globl  swint,usttsint,errint,uwrint
        .globl  diagint,badint

        .set    hpcctrl,0xFFFC00        # HPC Control/Status I/O location.
        .set    hpcdata,0xFFFE00        # HPC Data I/O location.
        .set    hpcpoll,0xFD0000        # HPC Poll address (UPIC).

        .set    cr,0xD
        .set    lf,0xA
        .set    ctrlD,'D'-0x40

start:
                                # Fill interrupt vector locations.

        addr    badint,vex              # Interrupt NMI.   (Unimplemented)
        addr    dataint,vex+4           # Interrupt 0x10.  Comm Buffer data.
        addr    rtcint,vex+8            # Interrupt 0x11.  Real-Time Clock.
        addr    badint,vex+12           # Interrupt 0x12.  (Unimplemented)
        addr    primeint,vex+16         # Interrupt 0x13.  Centronics PRIME.
        addr    badint,vex+20           # Interrupt 0x14.  (Unimplemented)
        addr    badint,vex+24           # Interrupt 0x15.  (Unimplemented)
        addr    badint,vex+28           # Interrupt 0x16.  (Unimplemented)
        addr    lcdint,vex+32           # Interrupt 0x17.  LCD data written.
        addr    swint,vex+36            # Interrupt 0x18.  Pushbutton event.
        addr    usttsint,vex+40         # Interrupt 0x19.  UART Status change.
        addr    errint,vex+44           # Interrupt 0x1A.  Error detected.
        addr    uwrint,vex+48           # Interrupt 0x1B.  UART Write ack.
        addr    badint,vex+52           # Interrupt 0x1C.  (Unimplemented)
        addr    diagint,vex+56          # Interrupt 0x1D.  Diagnostic.
        addr    badint,vex+60           # Interrupt 0x1E.  (Unimplemented)
        addr    badint,vex+64           # Interrupt 0x1F.  (Unimplemented)
        addr    badint,vex+68           # Interrupt 0x20.  (Unimplemented)
        addr    badint,vex+72           # Interrupt 0x21.  (Unimplemented)

        movb    $0,hpcctrl      # INITIALIZE command.
        movb    $100,hpcdata    # RTC value:  once per second.

        movb    $0x0B,hpcctrl   # Turn on two LED's to show we're alive.
        movb    $0x06,hpcdata

        movb    $1,hpcctrl      # Select Centronics port.
        movb    $1,hpcdata      #       BUSY drops during ACK/ pulse.
        movb    $100,hpcdata    #       Accept 100 characters before passing
                                #       buffer to CPU;
        movb    $120,hpcdata    #       Apply flow control if buffer has 120
                                #       characters.
```

TL/DD/9977-6

```
run:
       bispsrw $0x800          # Enable interrupts from HPC.

main:                   # Main program starts here.

       movd    datoptr,rl      # Register Rl contains buffer out pointer.

mwait: cmpd    datiptr,rl    # Wait here for a block to come in.
       bls     mwait

       movb    0(rl),r0        # Here, process character.
       cmpb    r0,$ctrlD       # if End of File, go type checksum.
       beq     typout

       addb    r0,ckdata
       addqd   $1,rl
       br      mwait

typout:                        # Send checksum out on LCDs.
       bicpsrw $0x800          # Disable interrupts.

       cbitb   $0,poutflg      # Clear LCD output acknowledge flag.

       movb    $0xA,hpcctrl    # Send-LCD command.
       movb    $0x6,hpcdata
       movb    $3,hpcdata

       movb    $0x1,hpcdata    # Clear panel LCD's.

       movzbd  ckdata,r0       # Send first hex character.
       lshd    $-4,r0
       movb    asctab[r0:b],r0
       movb    r0,hpcdata

       movzbd  ckdata,r0       # Send second hex character.
       andb    $0xF,r0
       movb    asctab[r0:b],r0
       movb    r0,hpcdata

       bispsrw $0x800          # Re-enable interrupts.

pnlout:        tbitb   $0,poutflg
       bfc     pnlout

       movqb   0,ckdata
       movd    $databuf,datiptr
       movd    datoptr,rl

       br      mwait   # Close loop:  infinite.

       ret     0       # End of main program.


maindat:       # Data for Main Program.

datiptr: .double databuf  # Pointer to Data Buffer area.
datoptr: .double databuf  # Pointer to Data Buffer area.
poutflg: .byte  1         # UART Output Ready.
ckdata:  .byte 0          # Accum. checksum.
asctab:         .byte  '0','1','2','3','4','5','6','7'
```

TL/DD/9977-7

```
         .byte  '8','9','a','b','c','d','e','f'
databuf: .blkb  1024   # Data buffer area.



        # Start of Interrupt Service Routines.
        # Invoked by ROM interrupt service.  Registers R0..R2 are already
        #   saved, but no ENTER instruction has been performed yet.

dataint:        # Interrupt 0x10.  Comm Buffer ready.

        movzbd  hpcdata,r0      # Get character count from HPC.
        movd    datiptr,r1

datalp:         movb    hpcdata,0(r1)   # Loop:  get character from HPC,
        addqd   1,r1            #        increment buffer address,
        acbd    -1,r0,datalp    #        decrement count and loop.

        movd    r1,datiptr
        ret     0

rtcint:                 # Interrupt 0x11.  Real-Time Clock.

        movb    $4,hpcctrl      # Send Flush-Buf command to HPC.
        ret     0

primeint:       # Interrupt 0x13.  Centronics PRIME.

        movb    $1,hpcctrl
        movb    $1,hpcdata
        movb    $100,hpcdata
        movb    $120,hpcdata
        ret     0

lcdint:                 # Interrupt 0x17.  LCD data written.

        sbitb   $0,poutflg
        ret     0

swint:          # Interrupt 0x18.  Pushbutton event.

        br      badint
        ret     0

usttsint:       # Interrupt 0x19.  UART Status change.

        br      badint
        ret     0

errint:                 # Interrupt 0x1A.  Error detected.

        br      badint
        ret     0

uwrint:                 # Interrupt 0x1B.  UART Write ack.

        br      badint
        ret     0

diagint:        # Interrupt 0x1D.  Diagnostic.
```

TL/DD/9977-8

```
        movb    hpcdata,r0
        movb    hpcdata,r0
        movb    hpcdata,r0
        movb    hpcdata,r0
        movb    hpcdata,r0
        ret     0

badint:                  # Trap for unimplemented interrupts.

        ret     0
```

TL/DD/9977-9

```
        # UART Port input / checksum calculation / UART output.
        #       Accepts up to 1024 characters on UART port,
        #       accumulates 8-bit checksum, and on receiving Ctrl-D,
        #       displays checksum by sending out on RS-232 port.

        .globl  start,main
        .globl  dataint,rtcint,primeint
        .globl  lcdint
        .globl  swint,usttsint,errint,uwrint
        .globl  diagint,badint

        .set    hpcctrl,0xFFFC00        # HPC Control/Status I/O location.
        .set    hpcdata,0xFFFE00        # HPC Data I/O location.
        .set    hpcpoll,0xFD0000        # HPC Poll address (UPIC).

        .set    cr,0xD
        .set    lf,0xA
        .set    ctrlD,'D'-0x40

start:
                                # Fill interrupt vector locations.

        addr    badint,vex              # Interrupt NMI.   (Unimplemented)
        addr    dataint,vex+4           # Interrupt 0x10.  Comm Buffer data.
        addr    rtcint,vex+8            # Interrupt 0x11.  Real-Time Clock.
        addr    badint,vex+12           # Interrupt 0x12.
        addr    primeint,vex+16         # Interrupt 0x13.  Centronics PRIME.
        addr    badint,vex+20           # Interrupt 0x14.
        addr    badint,vex+24           # Interrupt 0x15.
        addr    badint,vex+28           # Interrupt 0x16.
        addr    lcdint,vex+32           # Interrupt 0x17.  LCD data written.
        addr    swint,vex+36            # Interrupt 0x18.  Pushbutton event.
        addr    usttsint,vex+40         # Interrupt 0x19.  UART Status change.
        addr    errint,vex+44           # Interrupt 0x1A.  Error detected.
        addr    uwrint,vex+48           # Interrupt 0x1B.  UART Write ack.
        addr    badint,vex+52           # Interrupt 0x1C.  (Unimplemented)
        addr    diagint,vex+56          # Interrupt 0x1D.  Diagnostic.
        addr    badint,vex+60           # Interrupt 0x1E.  (Unimplemented)
        addr    badint,vex+64           # Interrupt 0x1F.  (Unimplemented)
        addr    badint,vex+68           # Interrupt 0x20.  (Unimplemented)
        addr    badint,vex+72           # Interrupt 0x21.  (Unimplemented)

        movb    $0,hpcctrl      # INITIALIZE command.
        movb    $100,hpcdata    # RTC value:  once per second.

        movb    $0x0B,hpcctrl   # Turn on two LED's to show we're alive.
        movb    $0x06,hpcdata

        movb    $2,hpcctrl      # Select UART and set up parameters.
        movb    $5,hpcdata      #       9600 baud,
        movb    $2,hpcdata      #       8 bits, no parity,
        movb    $0xA,hpcdata    #       XON/XOFF protocol, DTR always on.
        movb    $100,hpcdata    #       Accept 100 characters before passing
                                #       buffer to CPU;
        movb    $120,hpcdata    #       Apply flow control if buffer has 120
```

TL/DD/9977-10

```
                                # characters.
run:
        bispsrw $0x800          # Enable interrupts from HPC.

main:                   # Main program starts here.

        movd    datoptr,r1      # Register R1 contains buffer out pointer.

mwait:  cmpd    datiptr,r1    # Wait here for a block to come in.
        bls     mwait

        movb    0(r1),r0        # Here, process character.
        cmpb    r0,$ctrlD       # if End of File, go type checksum.
        beq     typout

        addb    r0,ckdata
        addqd   $1,r1
        br      mwait

typout:                         # Send checksum out on RS-232 port.
        movb    $cr,r0
        bsr     serout

        movb    $lf,r0
        bsr     serout

        movzbd  ckdata,r0
        lshd    $-4,r0
        movb    asctab[r0:b],r0
        bsr     serout

        movzbd  ckdata,r0
        andb    $0xF,r0
        movb    asctab[r0:b],r0
        bsr     serout

        movb    $cr,r0
        bsr     serout

        movb    $lf,r0
        bsr     serout

        movqb   0,ckdata
        movd    $databuf,datiptr
        movd    datoptr,r1

        br      mwait   # Close loop:  infinite.

        ret     0       # End of main program.

serout:         tbitb   $0,uoutflg
        bfc     serout

        cbitb   $0,uoutflg      # Indicate UART not ready.

        bicpsrw $0x800          # Critical Sequence:
        movb    $0xD,hpcctrl    #  Give Send-UART command to HPC.
        movb    r0,hpcdata      #  Give character to HPC.
        bispsrw $0x800          # End critical sequence.
```

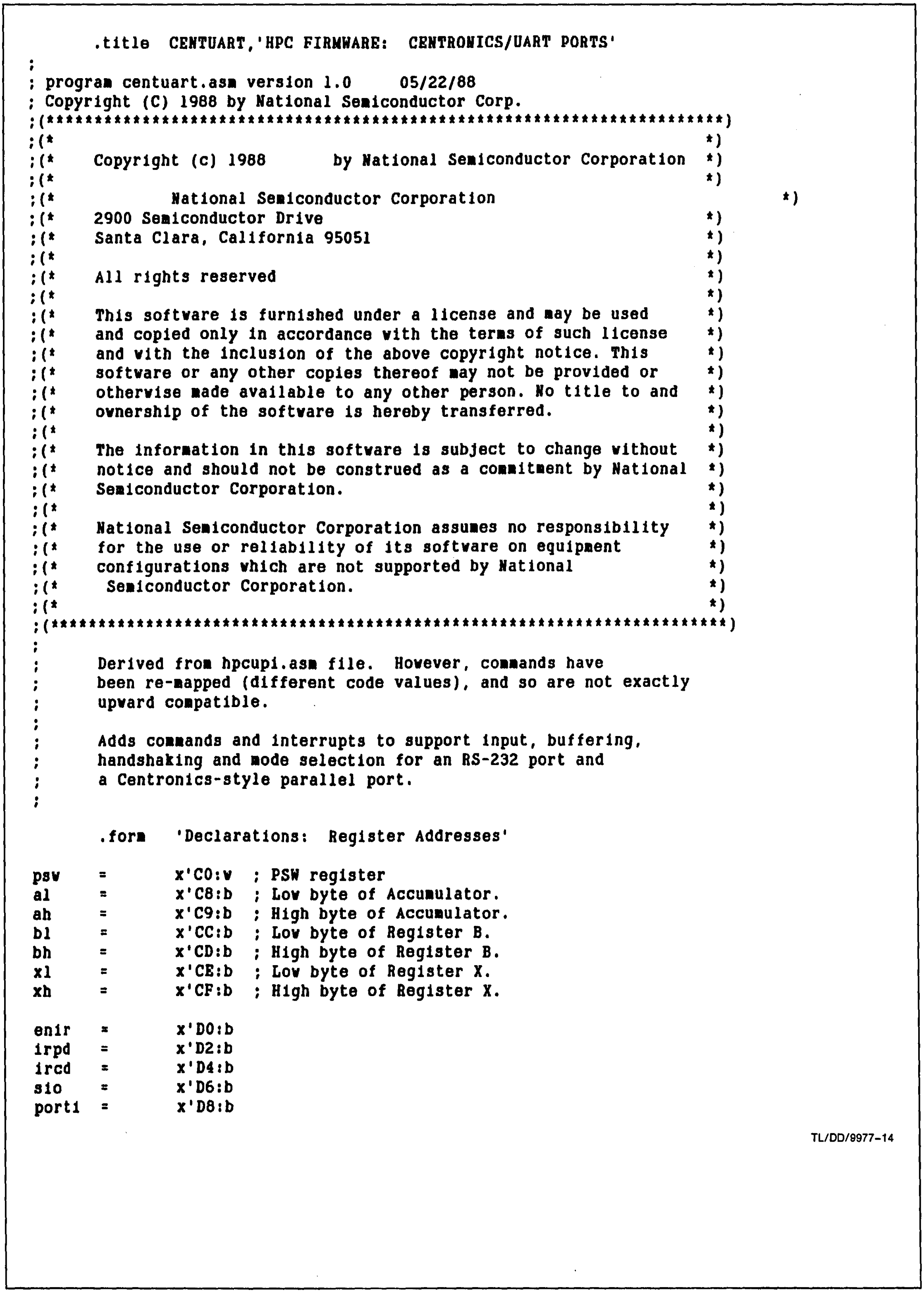
TL/DD/9977-11

```
        ret     0


maindat:        # Data for Main Program.

datiptr: .double databuf  # Pointer to Data Buffer area.
datoptr: .double databuf  # Pointer to Data Buffer area.
uoutflg: .byte  1       # UART Output Ready.
ckdata:  .byte 0        # Accum. checksum.
asctab:         .byte  '0','1','2','3','4','5','6','7'
        .byte  '8','9','a','b','c','d','e','f'
databuf: .blkb  1024    # Data buffer area.



        # Start of Interrupt Service Routines.
        # Invoked by ROM interrupt service.  Registers R0..R2 are already
        #   saved, but no ENTER instruction has been performed yet.

dataint:        # Interrupt 0x10.  Comm Buffer ready.

        movzbd  hpcdata,r0      # Get character count from HPC.
        movd    datiptr,r1

datalp:         movb    hpcdata,0(r1)   # Loop:  get character from HPC,
        addqd   1,r1            #        increment buffer address,
        acbd    -1,r0,datalp    #        decrement count and loop.

        movd    r1,datiptr
        ret     0

rtcint:                 # Interrupt 0x11.  Real-Time Clock.

        movb    $4,hpcctrl      # Send Flush-Buf command to HPC.
        ret     0

primeint:       # Interrupt 0x13.  Centronics PRIME.

        br      badint
        ret     0

lcdint:                 # Interrupt 0x17.  LCD data written.

        br      badint
        ret     0

swint:          # Interrupt 0x18.  Pushbutton event.

        br      badint
        ret     0

usttsint:       # Interrupt 0x19.  UART Status change.

        br      badint
        ret     0

errint:                 # Interrupt 0x1A.  Error detected.

        br      badint
```

TL/DD/9977-12

```
        ret     0

uwrint:                 # Interrupt 0x1B.  UART Write ack.

        sbitb   $0,uoutflg
        ret     0

diagint:        # Interrupt 0x1D.  Diagnostic.
        movb    hpcdata,r0
        movb    hpcdata,r0
        movb    hpcdata,r0
        movb    hpcdata,r0
        movb    hpcdata,r0
        ret     0

badint:                 # Trap for unimplemented interrupts.

        ret     0
```

TL/DD/9977-13



```
        .title  CENTUART,'HPC FIRMWARE:  CENTRONICS/UART PORTS'
;
; program centuart.asm version 1.0     05/22/88
; Copyright (C) 1988 by National Semiconductor Corp.
;(**********************************************************************)
;(*                                                                   *)
;(*    Copyright (c) 1988       by National Semiconductor Corporation *)
;(*                                                                   *)
;(*            National Semiconductor Corporation                                *)
;(*    2900 Semiconductor Drive                                       *)
;(*    Santa Clara, California 95051                                  *)
;(*                                                                   *)
;(*    All rights reserved                                            *)
;(*                                                                   *)
;(*    This software is furnished under a license and may be used     *)
;(*    and copied only in accordance with the terms of such license   *)
;(*    and with the inclusion of the above copyright notice. This     *)
;(*    software or any other copies thereof may not be provided or    *)
;(*    otherwise made available to any other person. No title to and  *)
;(*    ownership of the software is hereby transferred.               *)
;(*                                                                   *)
;(*    The information in this software is subject to change without  *)
;(*    notice and should not be construed as a commitment by National *)
;(*    Semiconductor Corporation.                                     *)
;(*                                                                   *)
;(*    National Semiconductor Corporation assumes no responsibility   *)
;(*    for the use or reliability of its software on equipment        *)
;(*    configurations which are not supported by National             *)
;(*     Semiconductor Corporation.                                    *)
;(*                                                                   *)
;(**********************************************************************)
;
;       Derived from hpcupi.asm file.  However, commands have
;       been re-mapped (different code values), and so are not exactly
;       upward compatible.
;
;       Adds commands and interrupts to support input, buffering,
;       handshaking and mode selection for an RS-232 port and
;       a Centronics-style parallel port.
;

        .form   'Declarations:  Register Addresses'

psw     =       x'C0:w  ; PSW register
al      =       x'C8:b  ; Low byte of Accumulator.
ah      =       x'C9:b  ; High byte of Accumulator.
bl      =       x'CC:b  ; Low byte of Register B.
bh      =       x'CD:b  ; High byte of Register B.
xl      =       x'CE:b  ; Low byte of Register X.
xh      =       x'CF:b  ; High byte of Register X.

enir    =       x'D0:b
irpd    =       x'D2:b
ircd    =       x'D4:b
sio     =       x'D6:b
porti   =       x'D8:b
```

TL/DD/9977–14

```
obuf    =       x'E0:b   ; (Low byte of PORTA.)
portah  =       x'E1:b   ; High byte of PORTA.
portb   =       x'E2:w
portbl  =       x'E2:b   ; Low byte of PORTB.
portbh  =       x'E3:b   ; High byte of PORTB.
upic    =       x'E6:b
ibuf    =       x'F0:b   ; (Low byte of DIRA.)
dirah   =       x'F1:b   ; High byte of DIRA.
dirb    =       x'F2:w
dirbl   =       x'F2:b   ; Low byte of DIRB.
dirbh   =       x'F3:b   ; High byte of DIRB.
bfun    =       x'F4:w
bfunl   =       x'F4:b   ; Low byte of BFUN.
bfunh   =       x'F5:b   ; High byte of BFUN.

portd   =       x'0104:b
enu     =       x'0120:b
enui    =       x'0122:b
rbuf    =       x'0124:b
tbuf    =       x'0126:b
enur    =       x'0128:b

t4      =       x'0140:w
r4      =       x'0142:w
t5      =       x'0144:w
r5      =       x'0146:w
t6      =       x'0148:w
r6      =       x'014A:w
t7      =       x'014C:w
r7      =       x'014E:w
pwmode  =       x'0150:w
pwmdl   =       x'0150:b ; Low byte of PWMODE.
pwmdh   =       x'0151:b ; High byte of PWMODE.
portp   =       x'0152:w
portpl  =       x'0152:b ; Low byte of PORTP.
portph  =       x'0153:b ; High byte of PORTP.
eicon   =       x'015C:b

t1      =       x'0182:w
r1      =       x'0184:w
r2      =       x'0186:w
t2      =       x'0188:w
r3      =       x'018A:w
t3      =       x'018C:w
divby   =       x'018E:w
divbyl  =       x'018E:b ; Low byte of DIVBY.
divbyh  =       x'018F:b ; High byte of DIVBY.
tmmode  =       x'0190:w
tmmdl   =       x'0190:b ; Low byte of TMMODE.
tmmdh   =       x'0191:b ; High byte of TMMODE.
t0con   =       x'0192:b


        .form   'Declarations:  Register Bit Positions

; Name      Position      Register(s)
; ----       --------      -----------

gie     =       0        ; enir
i2      =       2        ; enir, irpd, ircd
```

TL/DD/9977-15

```
i3      =       3       ; enir, irpd, ircd
i4      =       4       ; enir, irpd, ircd
tmrs    =       5       ; enir, irpd
uart    =       6       ; enir, irpd
ei      =       7       ; enir, irpd

dsr     =       7       ; porti only:  poll UART DSR.

uwmode =        1       ; ircd
uwdone =        0       ; irpd

tbmt    =       0       ; enu
rbfl    =       1       ; enu
b8or9   =       4       ; enu
xbit9   =       5       ; enu
wakeup =        2       ; enur
rbit9   =       3       ; enur
frmerr =        6       ; enur
doeerr =        7       ; enur
eti     =       0       ; enui
eri     =       1       ; enui
xtclk   =       2       ; enui
xrclk   =       3       ; enui
b2stp   =       7       ; enui

wrrdy   =       0       ; upic
rdrdy   =       1       ; upic
la0     =       2       ; upic
upien   =       3       ; upic
b8or16 =        4       ; upic

t0tie   =       0       ; tmmdl
t0pnd   =       1       ; tmmdl
t0ack   =       3       ; tmmdl
t1tie   =       4       ; tmmdl
t1pnd   =       5       ; tmmdl
t1stp   =       6       ; tmmdl
t1ack   =       7       ; tmmdl
t2tie   =       0       ; tmmdh
t2pnd   =       1       ; tmmdh
t2stp   =       2       ; tmmdh
t2ack   =       3       ; tmmdh
t3tie   =       4       ; tmmdh
t3pnd   =       5       ; tmmdh
t3stp   =       6       ; tmmdh
t3ack   =       7       ; tmmdh

t4tie   =       0       ; pwmdl
t4pnd   =       1       ; pwmdl
t4stp   =       2       ; pwmdl
t4ack   =       3       ; pwmdl
t5tie   =       4       ; pwmdl
t5pnd   =       5       ; pwmdl
t5stp   =       6       ; pwmdl
t5ack   =       7       ; pwmdl
t6tie   =       0       ; pwmdh
t6pnd   =       1       ; pwmdh
t6stp   =       2       ; pwmdh
t6ack   =       3       ; pwmdh
t7tie   =       4       ; pwmdh
```

TL/DD/9977-16

```
t7pnd  =        5       ; pwmdh
t7stp  =        6       ; pwmdh
t7ack  =        7       ; pwmdh

t4out  =        0       ; portpl
t4tfn  =        3       ; portpl
t5out  =        4       ; portpl
t5tfn  =        7       ; portpl
t6out  =        0       ; portph
t6tfn  =        3       ; portph
t7out  =        4       ; portph
t7tfn  =        7       ; portph

cenclk =        0       ; portph (CCTLCLK signal).


txd    =        0       ; portbl, dirbl, bfunl
dtr    =        1       ; portbl, dirbl
pnlclk =        7       ; portbl, dirbl

lcvclk =        1       ; portbh, dirbh
; ua0 would be  2     , but requires no setup.
uwrrdy =        3       ; portbh, dirbh, bfunh
cdata  =        4       ; portbh (enables Centronics data to Port D).
astts  =        5       ; portbh (enables panel switches to Port D).
ledclk =        6       ; portbh, dirbh
urdrdy =        7       ; portbh, dirbh, bfunh

;       CONSTANTS
;
xon=    x'11    ; XON character:   Control-Q
xoff=   x'13    ; XOFF character:  Control-S


        .form   'Space Declarations'

botad= x'40     ; First address in buffer.
topad= x'BF     ; Last address in buffer.
bufsiz=         topad-botad+1   ; Length of buffer.
;
        .sect   BUFFER,BASE,ABS=botad   ; Data Communication Buffer.

        .dsb    bufsiz

        .endsect


        .sect   DSECT,BASE,REL  ; Basepage RAM variables (addresses 0000-00BF)

; WORD-ALIGNED
dummy: .dsw 1   ; x'00,01       ; Destroyed on reset (address 0).
        .set    upicsv,dummy    ; Temporary image of UPIC register.
alert: .dsw 1   ; Alert status bits to main program:
                ;  generate interrupts to CPU.
        .set    alerth,alert+1  ; Declare top byte of ALERT word.
cpuad: .dsw 1   ; Current address within CPU command buffer.
cpubuf:         .dsw 4  ; Buffer for accepting command parameters from CPU.
lcdsix:         .dsw 1  ; Pointer into LCD character string buffer.

;BYTE-ALIGNED
```

TL/DD/9977-17

```
numchr:         .dsb 1  ; Number of characters currently in data buffer.
cadin: .dsb 1   ; Current input byte address in data buffer
                ;  (first empty loc.).
cadout:         .dsb 1  ; Current output byte address in data buffer.
pascnt:         .dsb 1  ; Number of characters before data buffer full enough to
                ;  transmit to CPU.
stpcnt:         .dsb 1  ; Number of characters before host system is told to stop
                ;  transmitting.
numout:         .dsb 1  ; Number of data characters (total) being sent to CPU in
                ;  current transfer.
cntout: .dsb 1  ; Number of data characters remaining to be sent to CPU in
                ;  current transfer.
bstat: .dsb 1   ; Buffer Status byte.
cps:   .dsb 1   ; Centronics Port Status byte
                ;  (image of control signals).
ackmd: .dsb 1   ; Acknowledge Timing Mode:  Position of ACK/ pulse edges
                ;  on Centronics port relative to BUSY falling edge.
curcmd:         .dsb 1  ; Current command byte from CPU being processed.
numexp:         .dsb 1  ; Number of parameter bytes expected before command processing
                ;  begins.
lcvs:  .dsb 1   ; Image of LCD Voltage (Contrast) latch setting;  needed with
                ;  LCD RS (PAUXO) signal coming from this latch.
fshlim:         .dsb 1  ; Flush limit count:  used to limit number of characters passed
                ;  to CPU when an error report is pending.
errchr:         .dsb 1  ; Holds character on which an error was detected.
errfgs:         .dsb 1  ; Holds error flags associated with error character.
lcdfgs:         .dsb 1  ; Holds flag bits for characters sent to Panel LCD display.
lcdnum:         .dsb 1  ; Number of characters to be sent to LCD display.
lcdsfg:         .dsb 1  ; Flag bits associated with characters in LCD String Buffer.
lcdsct:         .dsb 1  ; Counter for characters being sent to LCD display from String
                ;   Buffer.
swlast:         .dsb 1  ; Last-sampled switch values.
swlsnt:         .dsb 1  ; Last switch values sent to CPU.
beepct:         .dsb 1  ; Beep duration count.  Counts occurrences of T0 interrupt.
uframe:         .dsb 1  ; Frame mode for UART.
uflow: .dsb 1   ; Flow control mode for UART.
ups:   .dsb 1   ; UART Status byte.
uschr: .dsb 1   ; UART Send Character:  from CPU.
uinchr:         .dsb 1  ; UART Input Character:  character last received from UART.
enrimg:         .dsb 1  ; UART ENUR register image in memory.
rtcivl:         .dsb 1  ; Real-Time Clock Interval (units of 10 milliseconds).
rtccnt:         .dsb 1  ; Real-Time Clock Current Count (units of 10 milliseconds).
rtevs: .dsb 1   ; Events to check for on Timer T1 interrupts.
ustat: .dsb 1   ; UART status for CPU.
dsevc: .dsb 1   ; Diagnostic Interrupt:  Severity Code.
derrc: .dsb 1   ; Diagnostic Interrupt:  Error Code.
dbyte: .dsb 1   ; Diagnostic Interrupt:  Error Byte.
dccmd: .dsb 1   ; Diagnostic Interrupt:  Current Command.
dqual: .dsb 1   ; Diagnostic Interrupt:  Qualifier (Command Status).

; * Addresses 0040-00BF are reserved for the Data Communication Buffer
;    (128 bytes).

;      BIT POSITIONS

       ; Bits in BSTAT byte (Data Communication Buffer Status):
pass=  0        ; Data is ready to be passed to the CPU.
passng=         1       ; Indicates that some of the data in the buffer is being
                ;   passed to the CPU.
stop=  2        ; Indicates that host has been requested to stop transmitting.
```

TL/DD/9977-18

```
cpubsy=        3       ; Indicates that CPU is not able to receive any more data.
ifcbsy=        4       ; Indicates that the interface is considered busy by CPU.
full= 5        ; Indicates that the interface is completely full.  Any more
               ;   characters will overflow it.

       ; Bits in CPS (Centronics Port Status byte)
cack= 0        ; ACK/ Strobe.
caux1= 1       ; AUXOUT1 Signal.
cbusy= 2       ; BUSY Signal.
cselct=        3       ; SELECT Signal.
ccall= 4       ; CALL Signal.
cfault=        5       ; FAULT/ Signal.
caux2= 6       ; AUXOUT2 Signal.
enprm= 7       ; 1 enables INPUT PRIME/ interrupt from Centronics port.

       ; Bits in ACKMD (Centronics Acknowledge Mode byte)
; (Bits 0 and 1 give timing relationship between BUSY and ACK/.)
clinmd=        2       ; 1 = Centronics Line Mode.  Buffer limits must also both be 1.
; (Other bits unassigned.)

       ; ALERT status word (low-order byte) bits:
aflush =       0       ; Flush Data Buffer.
artc   =       1       ; Real-Time Interrupt detected.
aprime =       3       ; INPUT PRIME detected from Centronics interface.
alcdak =       7       ; LCD Panel Write Acknowledge.
       ; ALERT status word (high-order byte, named alerth) bits:
abutton        =       0       ; Pushbutton switch state change.
austat =       1       ; UART status change.
aerr   =       2       ; Error detected (UART or buffer overflow).
auack  =       3       ; UART output acknowledge:  character sent.
adiag  =       5       ; Diagnostic interrupt.
; (Other bits not defined.)

       ; ERRFGS error flags byte sent to CPU with !BAD-DATA interrupt:
doe=   7       ; Data Overrun Error on UART.
frm=   6       ; Framing Error on UART.
par=   5       ; Parity error on UART.
bufovf=        4       ; Buffer Overflow condition (flow control did not work).
errovf=        3       ; Error Overflow condition.  Two or more errors occurred
               ;   so close together that the first error could not be
               ;   reported before the second error occurred.  Details
               ;   of the second error are lost.
brk=   2       ; Break condition detected in addition to Framing error.
; (Other bits not defined.)

       ; CURCMD byte:  Current CPU command.  The lower 5 bits contain a code
       ;                 in the range 0-10 (hex).  The upper two bits contain
       ;                 further information about command collection:
cmdemp=        7       ; Bit 7 (MSB) of curcmd = 1 means that no command is being
               ;  processed and curcmd byte is "empty".
getcnt=        6       ; Bit 6 of curcmd = 1 means that the count is being received
               ;  for a variable-length command.

       ; LCVS byte:  LCD Voltage (Contrast) Latch memory image.
       ;               Contains voltage value in its least-significant 3 bits,
       ;               RS signal to LCD controller in bit 3, and debugging
       ;               information in its top 4 bits.
pnlrs= 3       ; Bit 3 is (inverted) RS signal to panel.

       ; UPS byte:  Status of UART output and flow control.
```

TL/DD/9977-19

```
usel=  7        ; When set, means that UART port is selected.
mcemd= 6        ; Receiver disabled due to multiple character error.
brkmd= 5        ; BREAK signal has been detected and is still active; receiver
                ;  disabled.
onebrk=         4       ; One character which is possibly a BREAK has been seen.
icpu=  3        ; When set, means that CPU should be informed of next UART
                ;  transmitter interrupt.
schr=  2        ; Request to send a character from uschr location (from CPU).
cus=   1        ; Current UART status:  1 = stopped.
luss=  0        ; Last UART Status Sent:  Indicates what the external system
                ;  thinks the UART's status is.

       ; UFLOW byte:  Modes for UART flow control.
flemp= 7        ; 1 = No flow control yet provided since reset.
;(intervening bits not defined.)
xonb=  3        ; 1 = XON/XOFF protocol mode selected.
dtrb1= 2        ; DTR Mode field:  00 = permanently low.
dtrb0= 1        ;                  01 = permanently high.
                ;                  10 = low when ready.
                ;                  11 = high when ready.
dsrb=  0        ; 1 = characters received while DSR low will not be accepted.

       ; USTAT byte:  Status of UART reported to CPU.
dsrflg=         0       ; State of DSR signal.  1 = Data Set Ready condition.
brkflg=         1       ; 1 = End of BREAK condition detected.

       ; RTEVS byte:  Events to check for at 10-millisecond intervals.
       ;              (T1 Underflows)
rtcenb=         0       ; 1 = Real-Time Clock interrupts enabled to CPU.
brkenb=         1       ; 1 = UART Break mode;  report end of break.


       .sect    STACK,RAM16,REL         ; On-chip RAM in addresses 01C0-01FF.
stackb:         .dsw    16      ; Space for 8 words beyond
                        ;  interrupt context.
avail: .dsw     12      ; Spare portion of this space.
lcdbuf:         .dsw    4       ; LCD String Buffer.

       .form    'Code Section'
       .sect    CSECT,ROM16,REL ; Code space.  (On-chip ROM)

       ; Declarations of subroutines called by one-byte JSRP instruction.

       .spt     rdwait          ; Waits for CPU to read a value from UPI port.
       .spt     wrpnl           ; Writes to LCD panel (for initialization only).

       ; Program starts at label "start" on reset.  This routine is the fatal
       ;  error handler, located here for convenience in setting breakpoint.

hangup:         rbit    gie,enir        ; Fatal error:  signal it and halt.
       sbit     7,lcvs          ; Signal error on most-significant bit of
                                ;  LCD Contrast Latch.
       sbit     pnlrs,lcvs      ; Select command mode for LCD controller.
       ld       portah,lcvs     ; Place error on Port A for latch.
       sbit     lcvclk,portbh   ; Clock LCD Contrast Latch high,
       rbit     lcvclk,portbh   ;  then low to load it.
       sbit     t6stp,pwmdh     ;
       rbit     t6tie,pwmdh     ; Set up Timer T6 for non-interrupt use.
       nop
       rbit     t6pnd,pwmdh     ; Clear Pending bit.
```

TL/DD/9977-20

```
        pop     0.w             ; Get error address from stack.
        ld      sp.w,#stackb    ; In case of stack underflow, re-initialize SP.
        ld      A,#x'01
        jsrl    wrpnl           ; Clear LCD panel.
        rbit    pnlrs,lcvs      ; Set up panel for data.
        ld      portah,lcvs     ; Place error on Port A for latch.
        sbit    lcvclk,portbh   ; Clock LCD Contrast Latch high,
        rbit    lcvclk,portbh   ;  then low to load it.
        ld      A,1.b           ; Process first character of return address.
        swap    A
        and     A,#x'0F
        ld      A,hextab[A].b
        jsrl    wrpnl           ; Display it on LCD panel.
        ld      A,1.b           ; Process second character of return address.
        and     A,#x'0F
        ld      A,hextab[A].b
        jsrl    wrpnl           ; Display it on LCD panel.
        ld      A,0.b           ; Process third character of return address.
        swap    A
        and     A,#x'0F
        ld      A,hextab[A].b
        jsrl    wrpnl           ; Display it on LCD panel.
        ld      A,0.b           ; Process last character of return address.
        and     A,#x'0F
        ld      A,hextab[A].b
        jsrl    wrpnl           ; Display it on LCD panel.

hgupi:  ifbit   rdrdy,upic      ; Check to see if OBUF register is full.
        ld      obuf,#vdiag     ; If not, fill it with !DIAG vector
                                ;  continuously.
        ifbit   i3,irpd         ; Check for UPI data ready.
        jp      hgupi1
        jp      hgupi

hgupi1:         ifeq    ibuf,#x'A5      ; Check for RESET command.
        jp      hgrst
        jp      hgupi2
hgrst:  ifbit   la0,upic
        jp      hgupi2
        jmpl    xreset          ; If so, then go reset the HPC.

                                ; This is part of the outer loop, waiting for
                                ;  the RESET command.
hgupi2:         ld      irpd,#x'F7      ; Clear the UWR detector,
        jp      hgupi           ;  and keep looking.  This is an
                                ;  infinite loop until RESET is seen.

hextab:         .byte   '0','1','2','3','4','5','6','7'
        .byte   '8','9','A','B','C','D','E','F'

        .form   'Hardware Initialization'

start:  ld      psw.b,#x'08     ; Set one WAIT state.

srfsh:                  ; Start dynamic RAM refreshing,
                        ;  as quickly as possible.
        sbit    t4out,portpl    ; Trigger first refresh
                                ;  immediately.
        sbit    t4stp,pwmdl     ; Stop timer T4 to
                                ;  allow loading,
```

TL/DD/9977-21

```
        ld      t4,#8           ;  then load it.
        rbit    t4stp,pwmdl     ; Start timer T4.
        sbit    t4tfn,portpl    ; Enable pulses out.
        ld      r4,#8           ; Load R4.

supi:                           ; Set up UPI port.
        ld      upic,#x'18      ; 8-Bit UPI Mode
                                ;  enabled.

        sbit    uwrrdy,bfunh    ; Enable UWRRDY/ out.
        sbit    uwrrdy,dirbh
        ld      A,ibuf          ; Empty IBUF register,
                                ;  in case of false trigger.

        sbit    urdrdy,bfunh    ; Enable URDRDY/ out.
        sbit    urdrdy,dirbh

                                ; Set up UREAD/ interrupt.
        sbit    i2,ircd         ; Detects rising edges.
        ld      irpd,#x'FB      ; Clear any false interrupt
                                ;  due to mode change.

                                ; Set up UWRITE/ interrupt.
        sbit    i3,ircd         ; Detects rising edges.
        ld      irpd,#x'F7      ; Clear any false interrupt
                                ;  due to mode change.

sram:                   ; Clear all RAM locations.
                        ; Clear Basepage bank:
        ld      BK,#x'0000,#x'00BE      ; Establish loop base and limit.
sraml1:         clr     A
        xs      A,[B+].w
        jp      sraml1

                        ; Clear Non-Basepage bank:
        ld      BK,#x'01C0,#x'01FE      ; Establish loop base and limit.
sraml2:         clr     A
        xs      A,[B+].w
        jp      sraml2

sskint:                         ; Set up Stack and remove
                        ;  individual interrupt enables.
        ld      sp.w,#stackb+2  ; Move stack to high
                                ;  bank of on-chip RAM.
        ld      stackb.w,#hangup  ; Safeguard against
                                  ;  stack underflow.
        ld      enir,#x'00      ; Disable interrupts
                                ;  individually.

tminit:         ld      t0con,#x'08
        ld      tmmode,#x'4440  ; Stop timers T1, T2, T3.
        ld      divby,#x'0055   ; UART set to 9600 Baud.
        ld      tmmode,#x'CCC8  ; Clear and disable timer
                                ;  T0-T3 interrupts.

scent:                          ; Set up Centronics parallel
                                ;  port.
        ld      dirah,#x'FF     ; Enable multiplexed outputs.
        sbit    astts,portbh    ; Enable and remove ENASTTS/ signal.
        sbit    astts,dirbh
```

TL/DD/9977-22

```
        sbit    cdata,portbh    ; Enable and remove ENCDATA/ signal.
        sbit    cdata,dirbh
        ld      cps,#x'25       ; Set up Port A data for
                                ;  Centronics Control.
        jsrl    setcen          ; Send to Centronics latch and to Busy flag.

                                ; Set up I4 interrupt on
        sbit    i4,ircd         ;  CINTR/ (rising edge).
        ld      irpd,#x'EF      ; Clear any false interrupt
                                ;  caused by mode change.

suart:                          ; Set up RS-232 port.
        sbit    txd,bfunl       ; Enable TXD output.
        sbit    txd,dirbl
        rbit    dtr,portbl      ; Set up DTR signal.  (State is arbitrary:
                                ;  low typically means not ready.)
        sbit    dtr,dirbl       ; Enable it as an output pin.
        ld      enu,#x'0        ; 8-bit Mode.
        ld      enur,#x'0       ; Clear Wake-Up Mode.
        ld      enui,#x'80      ; Internal baud;  2 stop
                                ;  bits;  no interrupts.

sled:   ld      portah,#x'FF    ; Set up to turn off LED's.
        rbit    ledclk,portbh   ; Start with LEDCLK low,
        sbit    ledclk,dirbh    ;  (enable output),
        sbit    ledclk,portbh   ;  then high,
        rbit    ledclk,portbh   ;  then low again.

stmrs:                          ; Set up remaining timers.
                                ;  (T1-T3 already stopped
                                ;   and pending bits cleared
                                ;   at tminit above.)

        ld      t1,#12287       ; T1 runs at 10-millisecond real-time interval.
        ld      r1,#12287
                                ; Timer remains stopped, and interrupt
                                ;  disabled, until INITIALIZE command.

        ld      pwmode,#x'4440  ; Stop timers T5-T7.
        nop                     ; Wait for valid PND
        nop                     ;  bits.
        ld      pwmode,#x'CCC8  ; Clear and disable
                                ;  interrupts from all
                                ;  PWM timers.

        ld      r6,#x'FFFF      ; No modulus for LCD Display Ready timer.

        ld      t7,#204 ; Set T7 to underflow at 6 KHz rate
        ld      r7,#204 ;  (= 3 KHz at pin).
        rbit    t7tfn,portph    ; Disable beep tone to panel speaker.
        rbit    t7stp,pwmdh     ; Start T7 running.


slcd:                   ; Set up LCD display.
                        ; Requires use of timer T6, so
                        ;  appears after timer initialization.

                        ; First, set up LCD contrast.
        ld      lcvs,#x'0A      ; Initialize memory image of LCD Voltage
                                ;  latch, containing RS (PAUX0) bit also.
```

TL/DD/9977-23

```
        ld      portah,lcvs     ; Arbitrary initial contrast level of 5,
                                ;  and RS/ (PAUX0/) is high (="command").
        rbit    lcvclk,portbh   ; Start with LCVCLK low,
        sbit    lcvclk,dirbh    ;  (enable output)
        sbit    lcvclk,portbh   ;  then high,
        rbit    lcvclk,portbh   ;  then low to get it into LCV latch.

                        ; Initialize PNLCLK (Panel "E" signal).
        sbit    pnlclk,portbl   ; Start with PNLCLK high
        sbit    pnlclk,dirbl    ;  (enable output).

                        ; Wait for worst-case command
                        ;  execution time (4.9 ms, twice), in case
                        ;  a panel command was triggered while
                        ;  PNLCLK was floating.
        sbit    t6ack,pwmdh     ; Clear T6 PND bit.
        ld      t6,#13000       ; Set T6 to twice 4.9 milliseconds.
        rbit    t6stp,pwmdh     ; Start timer T6.
lcdlpl:         ifbit   t6pnd,pwmdh     ; Wait for T6 PND bit
                                ;  to be set.
        jp      lcdgol
        jp      lcdlpl
lcdgol:         sbit    t6stp,pwmdh     ; Stop timer T6.
        sbit    t6ack,pwmdh     ; Clear T6 PND bit.

                        ; Reset Panel controller (per Hitachi HD44780
                        ;   User's Manual).

                        ; (Panel RS signal was set
                        ;  in LCD Contrast initialization above,
                        ;  so no change needed here to
                        ;  flag these as a commands.)

        ld      A,#x'38         ; Send "8-Bit Mode, 2 Lines" command:  one;
        jsrl    wrpnl
        ld      A,#x'38         ; two;
        jsrl    wrpnl
        ld      A,#x'38         ; three;
        jsrl    wrpnl
        ld      A,#x'38         ; four times.
        jsrl    wrpnl
        ld      A,#x'08         ; Disable display.
        jsrl    wrpnl
        ld      A,#x'01         ; Clear display RAM.
        jsrl    wrpnl

                        ; Initial default mode settings.

        ld      A,#x'06         ; Set mode to move cursor to the right, no
        jsrl    wrpnl           ;   automatic shifting of display.
        ld      A,#x'0E         ; Enable display:  non-blinking cursor mode.
        jsrl    wrpnl


;       CONTINUES TO MAIN PROGRAM INITIALIZATION
        .form   'Main Program Initialization'


minit:
                        ; Once-only initializations.
```

TL/DD/9977-24

```
        ld      curcmd,#x'80    ; Current Command:  top bit set means "none".
        ld      cpuad,#cpubuf   ; Set CPU command index to beginning of buffer.
        ld      numexp,#8       ; Arbitrary starting value.

                        ; Arbitrary set of initialization values for variables,
                        ;  in effect until receipt of the first INITIALIZE
                        ;  command.

        ld      numchr,#0       ; Clear count of characters received.
        ld      cadin,#botad    ; Next character in from comm port goes to
                                ;  first byte of buffer.
        ld      cadout,#botad   ; Next port data character out (to CPU)
                                ;  comes from first byte of buffer.
        ld      numout,#0       ; No characters being sent to CPU.
        ld      cntout,#0       ; No characters being sent to CPU.
        ld      pascnt,#125     ; Send to CPU when 125 characters received.
        ld      stpcnt,#126     ; Stop host when 126 characters received.
        ld      bstat,#0        ; Set buffer ready to receive.
        ld      alert,#0        ; No events pending.
        ld      ackmd,#1        ; BUSY will fall during ACK/ pulse.
        ld      errchr,#55      ; Arbitrary fill for error character.
        ld      errfgs,#0       ; Clear error detail flags.
        ld      uflow,#x'80     ; Set UART flow control mode byte empty.

runsys:                         ; Enable interrupts, start timers and go to main loop.

        sbit    tmrs,enir       ; Enable timer interrupts.  (Done here
                                ;  to allow certain commands without an
                                ;  INITIALIZE command first.)
        sbit    i3,enir         ; Enable CPU Command interrupt.
        sbit    gie,enir        ; Enable interrupt system.


        .form   'Main Scan Loop'

;       Declarations

vdata  =        x'10    ; CPU DATA vector number.
vrtc   =        x'11    ; Real-Time Clock vector number.
vprime =        x'13    ; Centronics INPUT PRIME signal.
vlcdak =        x'17    ; Acknowledge finished writing to LCD panel.
vbutton         =       x'18    ; Pushbutton status change: a button pressed or
                        ;  released.
vustat =        x'19    ; Change in UART DSR signal, or end of BREAK.
verr   =        x'1A    ; Character received with error from UART, or gross
                        ;   error condition in buffering or flow control on
                        ;   either port.
vuack  =        x'1B    ; UART output acknowledge:  character sent.
vdiag  =        x'1D    ; Diagnostic Interrupt.

mainlp:
                ; Error Vectors for unimplemented or
                ;   unexpected interrupts.

        .ipt    1,hangup        ; NMI:    never expected.
        .ipt    2,hangup        ; UPI READ READY:  never expected.
        .ipt    7,hangup        ; EI:     never expected.

chkdta:
```

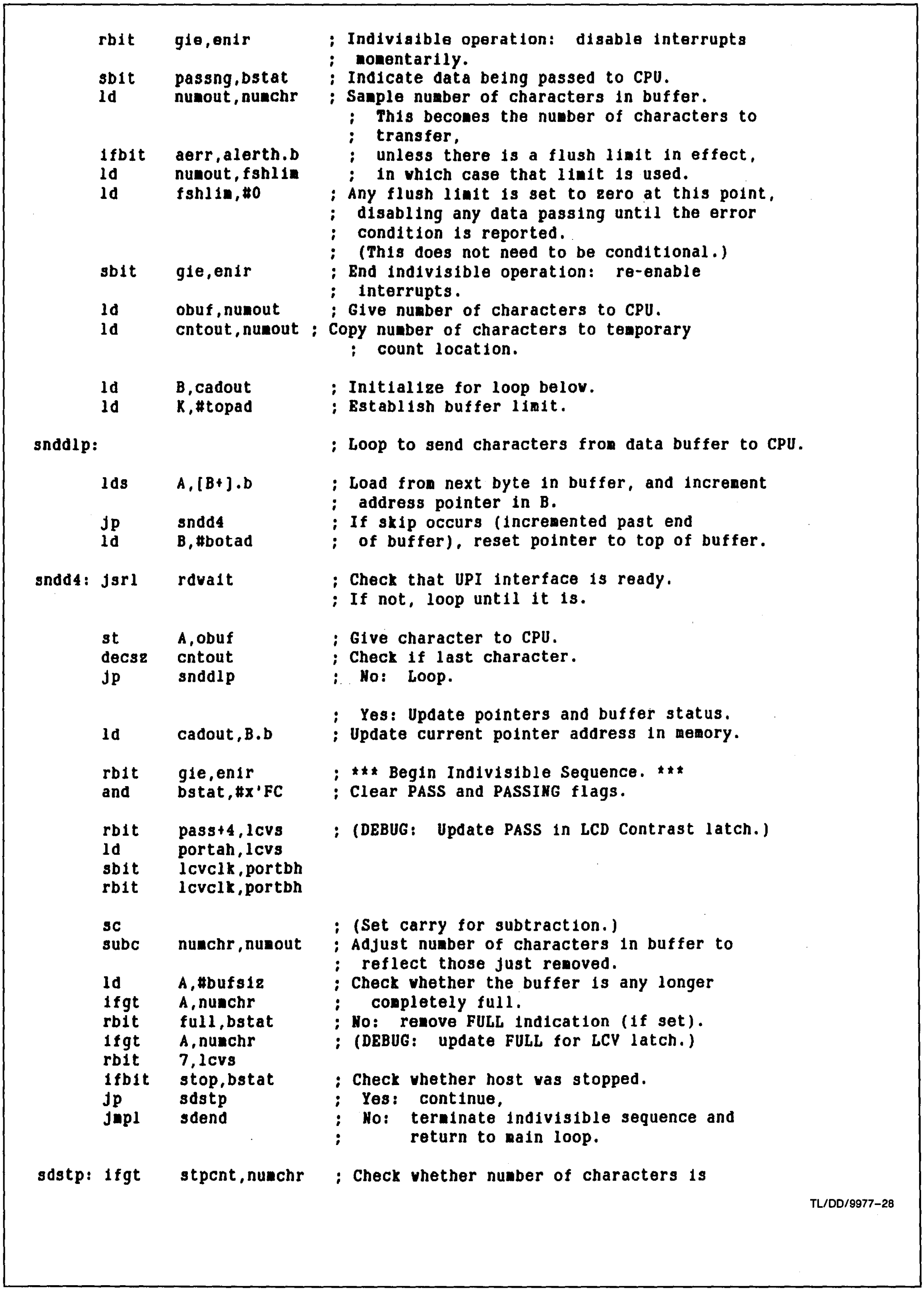
TL/DD/9977-25

```
        ld      A,bstat         ; Test state of buffer.
        and     A,#x'09         ; Check PASS and CPUBUSY bits.
        ifeq    A,#x'01         ; If PASS and not CPUBUSY,
        jsrl    snddta          ;  then go send a block of data to CPU.

chkalt:         ifeq    alert.w,#x'00   ; Check for alert conditions.
        jmpl    chkrsp          ; If none, go check for response ready.

        ifbit   artc,alert.b    ; Check for RTC interrupt request.
        jsrl    sndrtc          ; If so, then send Real-Time Clock Interrrupt.

        ifbit   aprime,alert.b  ; Check for Centronics Input Prime signal.
        jsrl    sndprm          ; If so, send Input Prime interrupt.

        ifbit   alcdak,alert.b  ; Check for LCD Panel write done.
        jsrl    sndlak          ; If so, then send LCD Acknowledge interrupt.

        ifbit   aflush,alert.b  ; Check for Flush Buffer request.
        jsrl    sndfsh          ; If so, then send data in buffer to CPU.

        ifbit   abutton,alerth.b ; Check for a pushbutton change.
        jsrl    sndbtn          ; If so, then report the change to the CPU.

        ifbit   austat,alerth.b ; Check for a UART status change.
        jsrl    sndust          ; If so, then report the change to the CPU.

        ifbit   aerr,alerth.b   ; Check for a data error condition.
        jp      cherr
        jp      nocher
cherr:  ifbit   cpubsy,bstat    ; Suppress if CPU busy.  (CPU needs to
        jp      nocher          ;  receive flushed characters first.)
        ifgt    fshlim,#0
        jsrl    sndfsh          ; If a flush is still needed, then do it first.
        jsrl    snderr          ; If so, then report the error to the CPU.
nocher:                                 ;  (This line deliberately empty.)

        ifbit   auack,alerth.b  ; Check for UART output done.
        jsrl    snduak          ; If so, then send UART-ACKNOWLEDGE interrupt.

        ifbit   adiag,alerth.b  ; Check for Diagnostic Interrupt.
        jsrl    sndiag          ; If so, then send interrupt and data.

chkrsp:         jmpl    chkdta          ; No "responses" defined yet;  just close loop.

        .form   'Main:  Send Real-Time Clock Interrupt'

; No data transfer;  just trigger interrupt and continue.

sndrtc:
        rbit    artc,alert.b    ; Clear ALERT bit.
        jsrl    rdwait          ; Check that UPI interface is ready.
                                ; If not, loop until it is.

        ld      obuf,#vrtc      ; Load Real-Time Clock vector into OBUF for CPU.
        ret                     ; Return to main loop.
```

TL/DD/9977-26

```
; No data transfer;  just trigger interrupt and continue.

sndlak:
        rbit    alcdak,alert.b  ; Clear ALERT bit.
        jsrl    rdwait          ; Check that UPI interface is ready.
                                ; If not, loop until it is.

        ld      obuf,#vlcdak    ; Load LCD-Acknowledge vector into OBUF for CPU.
        ret                     ; Return to main loop.

        .form   'Main:  Send Pushbutton Status to CPU'

sndbtn:
        jsrl    rdwait          ; Check that UPI interface is ready.
                                ; If not, loop until it is.

        ld      obuf,#vbutton   ; Load BUTTON-DATA vector into OBUF for CPU.

        jsrl    rdwait          ; Check that UPI interface is ready.
                                ; If not, loop until it is.

        rbit    gie,enir        ; *** Begin Indivisible Sequence ***
        ld      obuf,swlsnt     ; Load Pushbutton Data Byte into OBUF for CPU.
        rbit    abutton,alerth.b ; Clear ALERT bit.
        sbit    gie,enir        ; *** End Indivisible Sequence ***
        ret                     ; Return to main loop.


        .form   'Main:  Send Data from Data Buffer to CPU'

; Trashes A, B, K (limit), and C flag.  May trash X in future.

        ;  Buffer Flush request serviced here.
sndfsh:
        rbit    aflush,alert.b  ; Reset Flush request.
        ifeq    numchr,#0       ; If no characters to send, just return,
        ret                     ; else go to Send Data routine.
        jmpl    snddta

        ;  Automatic Pass condition serviced here.
snddta:
        ifbit   aerr,alerth.b   ; Check for a communication or buffer error.
        jp      chkflm          ; If so, there is a limit on the number of
                                ;  characters to send.  Investigate further.
        jp      snddl           ; Else, go ahead and perform automatic pass.

chkflm:         ifeq    fshlim,#0       ; Here, a flush limit is in effect due to an
        ret                     ;  error condition.  Check that the limit is
                                ;  non-zero before initiating the pass.  If
                                ;  zero, then simply return without passing.

snddl:  jsrl    rdwait          ; Check that UPI interface is ready.
                                ; If not, loop until it is.

        ld      obuf,#vdata     ; Load DATA vector into OBUF for CPU.

        jsrl    rdwait          ; Check that UPI interface is ready
                                ;  (CPU has acknowledged DATA interrupt).
                                ; If not, loop until it is.
```

TL/DD/9977-27



```
        rbit    gie,enir         ; Indivisible operation:  disable interrupts
                                 ;  momentarily.
        sbit    passng,bstat     ; Indicate data being passed to CPU.
        ld      numout,numchr    ; Sample number of characters in buffer.
                                   ;  This becomes the number of characters to
                                   ;  transfer,
        ifbit   aerr,alerth.b      ;  unless there is a flush limit in effect,
        ld      numout,fshlim      ;  in which case that limit is used.
        ld      fshlim,#0        ; Any flush limit is set to zero at this point,
                                 ;  disabling any data passing until the error
                                 ;  condition is reported.
                                 ;  (This does not need to be conditional.)
        sbit    gie,enir         ; End indivisible operation:  re-enable
                                 ;  interrupts.
        ld      obuf,numout      ; Give number of characters to CPU.
        ld      cntout,numout ; Copy number of characters to temporary
                                   ;  count location.

        ld      B,cadout         ; Initialize for loop below.
        ld      K,#topad         ; Establish buffer limit.

snddlp:                          ; Loop to send characters from data buffer to CPU.

        lds     A,[B+].b         ; Load from next byte in buffer, and increment
                                 ;  address pointer in B.
        jp      sndd4            ; If skip occurs (incremented past end
        ld      B,#botad         ;  of buffer), reset pointer to top of buffer.

sndd4:  jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, loop until it is.

        st      A,obuf           ; Give character to CPU.
        decsz   cntout           ; Check if last character.
        jp      snddlp           ;  No:  Loop.

                                 ;  Yes: Update pointers and buffer status.
        ld      cadout,B.b       ; Update current pointer address in memory.

        rbit    gie,enir         ; *** Begin Indivisible Sequence. ***
        and     bstat,#x'FC      ; Clear PASS and PASSING flags.

        rbit    pass+4,lcvs      ; (DEBUG:  Update PASS in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh

        sc                       ; (Set carry for subtraction.)
        subc    numchr,numout    ; Adjust number of characters in buffer to
                                 ;  reflect those just removed.
        ld      A,#bufsiz        ; Check whether the buffer is any longer
        ifgt    A,numchr         ;   completely full.
        rbit    full,bstat       ; No:  remove FULL indication (if set).
        ifgt    A,numchr         ; (DEBUG:  update FULL for LCV latch.)
        rbit    7,lcvs
        ifbit   stop,bstat       ; Check whether host was stopped.
        jp      sdstp            ;  Yes:  continue,
        jmpl    sdend            ;  No:  terminate indivisible sequence and
                                 ;       return to main loop.

sdstp:  ifgt    stpcnt,numchr    ; Check whether number of characters is
```

TL/DD/9977-28

```
                                  ;  now less than "Stop" value to host.
        jp      sdstpl
        jmpl    sdend             ; If not, then return to main loop.

sdstpl:         rbit    stop,bstat      ; Clear "Stop Host" flag.
        rbit    5,lcvs

                                  ; Check which port to enable for more data.
        ifbit   usel,ups          ; Check if UART is selected.
        jmpl    sdusts            ; If so, go set up flow control.
        ifbit   enprm,cps         ; Check if Centronics port is selected.
        jmpl    sdcsts            ; If so, go set up Centronics BUSY.
        jmpl    sdend             ; Otherwise, do nothing more and return.

sdcsts:         ifbit   clinmd,ackmd    ; Check if in Centronics Line Mode.  If so,
        jmpl    sdend             ;   the CPU itself must command the ACK action.
        ld      A,bstat           ; Test whether data communication with
                                  ;   host should be allowed to continue.
        and     A,#x'3C           ;   Bits involved are STOP, CPUBSY, IFCBSY and
                                  ;   FULL.
        ifeq    A,#x'00           ;   If no stop conditions are in effect,
        rbit    cbusy,cps         ;   clear the BUSY indication in CPS
                                  ;   (Centronics Port Status) byte in memory.
        ifbit   i4,ircd           ; If not between the two interrupt services
                                  ;   of a Centronics strobe, then
        jsrl    setcen            ;   call Centronics port control setup routine,
                                  ;   to generate ACK/ pulse and clear BUSY.
                                  ; (If this sequence does occur between the
                                  ;  leading and trailing edge interrupts for
                                  ;  STROBE/, then the trailing edge routine
                                  ;  will pulse ACK/ when it is allowed to run.)
        jmpl    sdend

sdusts:         rbit    cus,ups         ; Set UART not busy.
        jsrl    dtron             ; Set DTR handshake appropriately.
        ifbit   eti,enui          ; Check if a UART transmitter interrupt will
                                  ;  be occurring.
        jmpl    sdend             ; If so, then no further action is required.
        ifbit   xonb,uflow        ; Otherwise, if XON protocol is in effect,
        jsrl    setuar            ;  then check and perform flow control.
        jmpl    sdend             ; Then exit to main program.

sdend:
        ld      portah,lcvs       ; (DEBUG: Update LCV latch.)
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh
        sbit    gie,enir          ; *** End Indivisible Sequence. ***

        ret                       ; Return to main program loop.

        .form   'Main:  Send Input Prime interrupt to CPU'

sndprm:                           ; Send INPUT PRIME interrupt to CPU.
        rbit    aprime,alert.b    ; Clear ALERT bit.
        jsrl    rdwait            ; Check that UPI interface is ready.
                                  ; If not, loop until it is.

        ld      obuf,#vprime      ; Load PRIME vector into OBUF for CPU.
        ret                       ; Return to main program loop.
```

TL/DD/9977-29

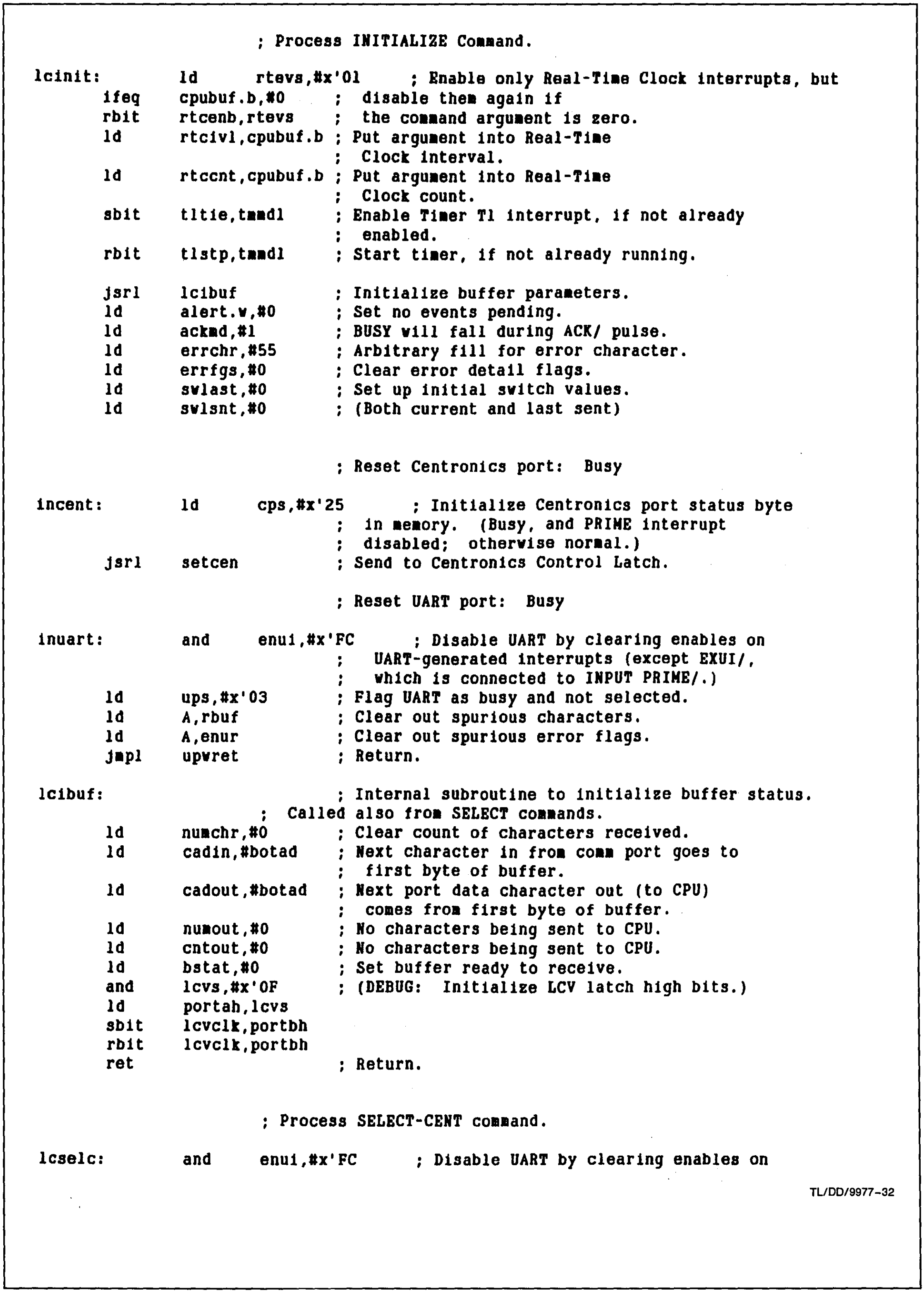
```
        .form   'Main:  Report a UART DSR change or END OF BREAK'

sndust:
        jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, loop until it is.

        ld      obuf,#vustat     ; Load UART-STATUS vector into OBUF for CPU.

        jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, loop until it is.

        rbit    gie,enir         ; * INDIVISIBLE SEQUENCE *
        rbit    austat,alerth.b  ; Clear ALERT bit.
        ld      obuf,ustat       ; Load UART Status Byte into OBUF for CPU.
        rbit    brkflg,ustat     ; Clear END OF BREAK indication.
        sbit    gie,enir         ; * END INDIVISIBLE SEQUENCE *
        ret                      ; Return to main loop.

        .form   'Main:  Report a Data Error Condition to CPU'

snderr:                          ; Send DATA-ERR interrupt to CPU.
        rbit    aerr,alerth.b    ; Clear ALERT bit.
        jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, wait until it is.

        ld      obuf,#verr       ; Load DATA-ERR vector into OBUF for CPU.
        jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, wait until it is.

        ld      obuf,errchr      ; Give CPU the offending character.
        jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, wait until it is.

        ld      obuf,errfgs      ; Give CPU the error flags.
        ret                      ; Return to main program loop.

        .form   'Main:  Send UART Acknowledge interrupt to CPU'

snduak:                          ; Send ACK-UART interrupt to CPU.
        rbit    auack,alerth.b   ; Clear ALERT bit.
        jsrl    rdwait           ; Check that UPI interface is ready.
                                 ; If not, loop until it is.

        ld      obuf,#vuack      ; Load ACK-UART vector into OBUF for CPU.
        ret                      ; Return to main program loop.


        .form   'Main:  Send Diagnostic Interrupt to CPU'

sndiag:
        jsrl    rdwait           ; Wait for UPI interface ready.
        ld      obuf,#vdiag      ; Load vector into OBUF for CPU.
        jsrl    rdwait           ; Wait for UPI interface ready.
        rbit    gie,enir         ; *** Begin Indivisible Sequence ***
        ld      obuf,dsevc       ; Transfer Severity Code.
        ld      dsevc,#0         ; Clear it.
        ld      A,derrc          ; Get Error Code.
        ld      derrc,#0         ; Clear it.
        rbit    adiag,alerth.b   ; Clear ALERT bit.
        sbit    gie,enir         ; *** End Indivisible Sequence ***
```

TL/DD/9977-30

```
        jsrl    rdwait          ; Wait for UPI interface ready.
        st      A,obuf          ; Transfer Error Code.
        jsrl    rdwait          ; Wait for UPI interface ready.
                        ; Remaining bytes will have meaning only for
                        ;  command errors.
        ld      obuf,dbyte      ; Transfer Byte Received.
        jsrl    rdwait          ; Wait for UPI interface ready.
        ld      obuf,dccmd      ; Transfer Current Command.
        jsrl    rdwait          ; Wait for UPI interface ready.
        ld      obuf,dqual      ; Transfer Command Count.
        ret                     ; Return to main program loop.

        .form   'UPI (I3) Interrupt:  Data from CPU'

        .ipt    3,upiwr         ; Declare upiwr as vector for Interrupt 3.

upiwr:                          ; Write Strobe received from CPU.
        push    A               ; Save Context
        push    psw

        ld      upicsv.b,upic   ; Save UPIC register image for LA0 bit test.

        ifbit   cmdemp,curcmd   ; If expecting first byte of a command,
        jmpl    firstc          ;  then go process it as such.

        ld      A,ibuf          ; If not, input it for entry into cpubuf.

        ifeq    A,#x'A5         ; Check for RESET command.
        jp      lcrst
        ifbit   la0,upicsv.b    ; Check for command argument written to proper
                                ;  address.
        jp      lcord           ; If so, go process as a normal argument.
        jsrl    hangup          ; If not, process as a FATAL error, generating
                                ;  !DIAG interrupt.

lcrst:  ifbit   la0,upic        ; Continue checking for a RESET command.
        jp      lcord
        jmpl    xreset          ; If so, go reset the HPC.

lcord:  x       A,[cpuad].b     ; If not, place it in next available cpubuf
                                ;  entry.
        inc     cpuad
        decsz   numexp
        jmpl    upwret          ; If not final byte of command, then return.

lastc:  ld      A,curcmd        ; Else, process current command.
        ifbit   getcnt,A.b      ; Check if extended collection is being made.
        jp      lastcl          ; If not, then:
        sbit    cmdemp,curcmd   ;   Set command slot available again.
        ld      cpuad,#cpubuf   ;   Reset CPU buffer pointer to beginning.

lastcl:         and     A,#x'1F         ; Mask off flag bits.
        shl     A               ; Scale by two, and then
        .odd
        jidw                    ;  jump based on command value.
        .ptw    lcinit,lcselc,lcselu,illc
        .ptw    illc,illc,illc,illc  ; (All these are one-byte commands.)
        .ptw    lcscst,lcslcv,lcslcd,lcsled
        .ptw    illc,lcsndu,illc,illc
        .ptw    illc,illc
```

TL/DD/9977-31

5



```
                        ; Process INITIALIZE Command.

lcinit:         ld      rtevs,#x'01     ; Enable only Real-Time Clock interrupts, but
        ifeq    cpubuf.b,#0       ;  disable them again if
        rbit    rtcenb,rtevs      ;  the command argument is zero.
        ld      rtcivl,cpubuf.b   ; Put argument into Real-Time
                                  ;  Clock interval.
        ld      rtccnt,cpubuf.b   ; Put argument into Real-Time
                                  ;  Clock count.
        sbit    tltie,tmmdl       ; Enable Timer Tl interrupt, if not already
                                  ;  enabled.
        rbit    tlstp,tmmdl       ; Start timer, if not already running.

        jsrl    lcibuf            ; Initialize buffer parameters.
        ld      alert.w,#0        ; Set no events pending.
        ld      ackmd,#1          ; BUSY will fall during ACK/ pulse.
        ld      errchr,#55        ; Arbitrary fill for error character.
        ld      errfgs,#0         ; Clear error detail flags.
        ld      swlast,#0         ; Set up initial switch values.
        ld      swlsnt,#0         ; (Both current and last sent)


                                  ; Reset Centronics port:  Busy

incent:         ld      cps,#x'25       ; Initialize Centronics port status byte
                                  ;  in memory.  (Busy, and PRIME interrupt
                                  ;  disabled;  otherwise normal.)
        jsrl    setcen            ; Send to Centronics Control Latch.

                                  ; Reset UART port:  Busy

inuart:         and     enui,#x'FC      ; Disable UART by clearing enables on
                                  ;   UART-generated interrupts (except EXUI/,
                                  ;   which is connected to INPUT PRIME/.)
        ld      ups,#x'03         ; Flag UART as busy and not selected.
        ld      A,rbuf            ; Clear out spurious characters.
        ld      A,enur            ; Clear out spurious error flags.
        jmpl    upwret            ; Return.

lcibuf:                           ; Internal subroutine to initialize buffer status.
                        ;  Called also from SELECT commands.
        ld      numchr,#0         ; Clear count of characters received.
        ld      cadin,#botad      ; Next character in from comm port goes to
                                  ;  first byte of buffer.
        ld      cadout,#botad     ; Next port data character out (to CPU)
                                  ;  comes from first byte of buffer.
        ld      numout,#0         ; No characters being sent to CPU.
        ld      cntout,#0         ; No characters being sent to CPU.
        ld      bstat,#0          ; Set buffer ready to receive.
        and     lcvs,#x'0F        ; (DEBUG:  Initialize LCV latch high bits.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh
        ret                       ; Return.


                        ; Process SELECT-CENT command.

lcselc:         and     enui,#x'FC      ; Disable UART by clearing enables on
```

TL/DD/9977-32

```
                                ;   UART-generated interrupts (except EXUI/,
                                ;   which is connected to INPUT PRIME/.)
        rbit    usel,ups        ; Flag UART not selected.
        ifbit   flemp,uflow     ; If valid UART mode exists,
        jp      lcsecl
        jsrl    dtroff          ;   use it to set DTR to "not ready" state.

lcsecl:         ld      ackmd,cpubuf.b ; Accept ACK/ mode from command buffer.
        ld      pascnt,cpubuf+1.b       ; Put "Buffer Pass" value into
                                        ;  the PASCNT slot.
        ld      stpcnt,cpubuf+2.b       ; Put "Host Stop" value into
                                        ;  the STPCNT slot.
        jsrl    lcibuf          ; Initialize buffer parameters.

primlp:         ifbit   uart,irpd       ; Check to see if INPUT PRIME/ interrupt is
        jp      primlp          ;   still asserted.  If so, wait here.

        sbit    i4,ircd         ; Set up STROBE detector to see leading edge.
        ld      irpd,#x'EF      ; Clear any spurious interrupt triggered by
                                ;  polarity change.
        sbit    i4,enir         ; Enable interrupts on I4 (STROBE).
        sbit    uart,enir       ; Enable INPUT PRIME/ interrupt (through
                                ;   UART vector).
        ld      cps,#x'A9       ; Set Centronics interface byte not busy,
                                ;   selected, and all status bits normal.
        jsrl    setcen          ; Clears BUSY signal and generates ACK/ pulse
                                ;  according to current mode in ACKMD.
        jmpl    upwret          ; Return.


                        ; Process SELECT-UART command.

lcselu:         ld      A,divby.b       ; Process UART baud selection.
        and     A,#x'0F         ; Strip out old baud rate selector.
        st      A,cpubuf+7.b    ; Save (in unused area of the command buffer),
                                ;  and start processing new value.
        ifgt    cpubuf.b,#x'08  ; Check if out of range.
        jsrl    hangup
        ld      A,#10
        sc
        subc    A,cpubuf.b      ; Convert to DIVBY field format.
        swap    A               ; Place value in correct field.
        or      A,cpubuf+7.b    ; OR with Microwire rate field.
        st      A,divby.b       ; Place back in DIVBY register.

        ld      uframe,cpubuf+1.b ; Get requested frame format.
        and     uframe,#x'07    ; Discard unused bits.
        sbit    b8or9,enu       ; Set 9-bit mode for 8-bit data plus parity.
        ifgt    uframe,#1       ; If 7-bit plus parity, or 8-bit without parity,
        rbit    b8or9,enu       ;  then change this setting to 8-bit mode.

        rbit    b2stp,enui      ; Initialize to one Stop bit.
        ifeq    uframe,#3       ; Test for number of Stop bits requested,
        sbit    b2stp,enui      ;  and set up UART hardware accordingly.
        ifgt    uframe,#5
        sbit    b2stp,enui

        ld      A,cpubuf+2.b    ; Set up handshaking mode.  This also clears
        and     A,#x'0F         ;  the FLEMP bit automatically.
        st      A,uflow
```

TL/DD/9977-33

5

```
        ld      pascnt,cpubuf+3.b       ; Put "Buffer Pass" value into
                                        ;  the PASCNT slot.
        ld      stpcnt,cpubuf+4.b       ; Put "Host Stop" value into
                                        ;  the STPCNT slot.
        jsrl    lcibuf          ; Initialize buffer parameters.

        ld      cps,#x'25       ; Set up Port A to disable and de-select
                                ;  Centronics port, and disable
                                ;  INPUT PRIME interrupt.
        rbit    clinmd,ackmd    ; Clear the Centronics Line Mode bit.
        jsrl    setcen          ; Send to Centronics latch and to Busy flag.
        rbit    i4,enir         ; Disable Centronics STROBE interrupt.
        ld      A,rbuf          ; Clear any pending character before selection.
        ld      A,enur          ; Clear any error indications before selection.
        sbit    eri,enui        ; Enable receiver interrupt.
        rbit    eti,enui        ; Disable transmitter interrupt.
        ld      ups,#x'80       ; Set UART port selected, not busy, and
                                ;  no characters being sent or waiting to be
                                ;  sent.
        ld      ustat,#x'01     ; Set DSR ready(will trigger interrupt if not).
        sbit    uart,enir       ; Enable UART interrupt.

        ifbit   dtrb0,uflow     ; Initialize DTR pin according to new mode.
        jp      lcslu1
        rbit    dtr,portbl
        jp      lcslu2
lcslu1:         sbit    dtr,portbl
lcslu2:

        jmpl    upwret          ; Return.



                        ; Process SET-CENT-STS Command.

lcscst:         ld      cps,cpubuf.b    ; Load Centronics Port Status from byte
                                ;   provided by CPU.
        jsrl    setcen          ; Perform ACK/ if new status calls for it.
        jmpl    upwret


                        ; Process SET-CONTRAST Command.

lcslcv:         ld      A,cpubuf.b      ; Load LCD Voltage latch (Contrast) from byte
                                ;   supplied by CPU.
        comp    A               ; (3-bit value is in complemented form.)
        and     A,#x'07         ; Use only lower three bits.
        and     lcvs,#x'F8      ; Clear field in memory image.
        or      lcvs,A.b        ; Merge new field into image.
        ld      portah,lcvs     ; Place on Port A (input to latch).
        sbit    lcvclk,portbh   ; Clock latch.
        rbit    lcvclk,portbh
        jmpl    upwret


                        ; Process SEND-LCD Command.

lcslcd:         ifbit   getcnt,curcmd           ; Check for first or second collection
        jmpl    lcslc1                  ;  phase.
```

TL/DD/9977-34

```
lcslc2:                                  ; Second phase:  begins execution of the LCI
                                 ;   command.
        ld      lcdbuf.w,cpubuf.w        ; Copy CPU buffer to LCD string buffer.
        ld      lcdbuf+2.w,cpubuf+2.w
        ld      lcdbuf+4.w,cpubuf+4.w
        ld      lcdbuf+6.w,cpubuf+6.w
        ld      lcdsct,lcdnum            ; Move number of characters to string
                                         ;  count byte
        inc     lcdsct                   ;  (incremented by one because of
                                         ;   extra interrupt occurring after
                                         ;   last character has been sent).
        ld      lcdsix,#lcdbuf           ; Set string pointer to first byte.
        ld      lcdsfg,lcdfgs            ; Move flag bits to string location.

        ld      r6,#x'FFFF               ; Set up R6 and T6 to trigger string
        ld      t6,#0                    ;  transfer.
        sbit    t6tie,pwmdh              ; Enable timer T6 interrupt.
        rbit    t6stp,pwmdh              ; Start timer to trigger (immediate)
                                         ;  interrupt from timer T6.
        jmpl    upwret

lcslcl:                                  ; First phase:  Prepare to collect up to 8
                                 ;   more bytes of command.
        ld      lcdfgs,cpubuf.b          ; Get flag bits supplied by CPU.
        ld      lcdnum,cpubuf+1.b        ; Get character count from CPU.

        ld      numexp,lcdnum            ; Request another collection of
                                         ;  data from the CPU (the string of
                                         ;  data for the panel).
        ld      cpuad,#cpubuf            ; Reset CPU collection pointer to start
                                         ;  of command buffer.
        rbit    getcnt,curcmd            ; Declare that it will be the final
                                         ;  collection.
        jmpl    upwret


                         ; Process SEND-LED Command.

lcsled:         ld      A,cpubuf.b       ; Load LED latch from byte supplied by CPU.
        comp    A               ; (Data goes to LED's in complemented form.)
        st      A,portah        ; Place new value on Port A (input to latch).
        sbit    ledclk,portbh   ; Clock latch.
        rbit    ledclk,portbh
        jmpl    upwret


                         ; Process SEND-UART Command.

lcsndu:
        ld      uschr,cpubuf.b ; Queue this character,
        sbit    schr,ups        ;  and request transmission at next
                                ;  transmitter interrupt.
        ifbit   eti,enui        ;  Check to see if another character is
        jmpl    upwret          ;  already being sent (transmitter interrupt
                                ;  enabled).
        jsrl    setuar          ;  If not, then call flow control routine to
                                ;    send it.
        jmpl    upwret          ; Return.
```

TL/DD/9977-35

```
        .form   'Processing of First Byte of Command (Code)'

                ;  One-byte commands are processed in this section.
                ;  Longer commands are scheduled for collection of
                ;    remaining bytes, and are processed in routines
                ;    above.

firstc:         ld      A,ibuf          ; Get command from UPI port.
        ifbit   la0,upicsv.b    ; Check for out-of-sequence condition
                                ;  (argument instead of command).
        jsrl    hangup          ; If so, process as a FATAL error (previous
                                ;  command was too short).

                        ; Processing of RESET command.

        ifeq    A,#x'A5         ; Check for RESET command.
        jp      xreset
        jp      fcord

                                ; This code is entered whenever a RESET
                                ;  command is received.
xreset:
        ld      obuf,#vdiag     ; Present dummy value for CPU,
                                ;  (in case a value was already in OBUF),
        jsrl    rdwait          ;  and wait for it to be read by CPU.
        ld      A,#0            ; Initialize registers.
        st      A,upic.b
        st      A,ibuf.w        ; (Actually all of DIRA.)
        st      A,dirb.w
        st      A,bfun.w
        st      A,ircd.b
        st      A,portp.w
        st      A,sp.w          ; Then, through RESET vector,
        st      A,psw.w
        ret                     ;  jump to start of program.

                        ; Here, process an ordinary command (not RESET).

fcord:
        and     A,#x'1F         ; Use only least-significant 5 bits.
        ifgt    A,#x'11         ; Check for command out of range.
        jmpl    illc
        st      A,curcmd        ; Save as current command.

        shl     A               ; Scale by two, and then
        .odd
        jidw                    ;  jump based on command value.
        .ptw    fcinit,fcselc,fcselu,illc
        .ptw    fcflsh,fccbsy,fccnby,fcifby
        .ptw    fcscst,fcslcv,fcslcd,fcsled
        .ptw    fcbeep,fcsndu,fcusts,illc
        .ptw    illc,illc

fcinit:         ld      numexp,#1       ; First byte of INITIALIZE command.
                                ;  Expects 1 more byte (RTC interval).
        jmpl    upwret          ; Return.

fcselc:         ld      numexp,#3       ; First byte of SELECT-CENTRONICS command.
```

TL/DD/9977-36

```
                                ;  Expects 3 more bytes (ACK-Mode, Pass-Count,
                                ;  Stop-Count).
       jmpl    upwret           ; Return.

fcselu:        ld      numexp,#5       ; First byte of SELECT-UART command.
                                ;  Expects 5 more bytes (baud, frame,
                                ;  handshake, Pass-Count, Stop-Count)
       jmpl    upwret           ; Return.

                       ; Processing of one-byte FLUSH-BUF command.
fcflsh:        sbit    aflush,alert.b  ; Set flush request bit in ALERT byte.
       sbit    cmdemp,curcmd    ; Set command byte empty (end of command).
       jmpl    upwret

                       ; Processing of one-byte CPU-BUSY command.
fccbsy:        sbit    cpubsy,bstat    ; Set CPU Busy bit in BSTAT byte.
       sbit    6,lcvs           ; (DEBUG:  set also CPU Busy bit in LCV latch.)
       ld      portah,lcvs
       sbit    lcvclk,portbh
       rbit    lcvclk,portbh
       sbit    cmdemp,curcmd    ; Set command byte empty (end of command).
       jmpl    upwret

                       ; Processing of one-byte CPU-NOT-BUSY command.
fccnby:        rbit    cpubsy,bstat    ; Reset CPU Busy bit in BSTAT byte.
       rbit    6,lcvs           ;(DEBUG: reset also CPU Busy bit in LCV latch.)
       ld      portah,lcvs
       sbit    lcvclk,portbh
       rbit    lcvclk,portbh
       sbit    cmdemp,curcmd    ; Set command byte empty (end of command).
       jmpl    upwret

fcifby:                         ; Processing of one-byte SET-IFC-BUSY command.
                       ; This command (one byte) sets the interface busy
                       ;   immediately, to stop characters from the external
                       ;   system.
       sbit    cmdemp,curcmd    ; Set command byte empty (end of command).
       ifbit   usel,ups         ; Check if UART is selected.
       jmpl    fcibyu           ; If so, go set up flow control.
       ifbit   enprm,cps        ; Check if Centronics port is selected.
       jmpl    fcibyc           ; If so, go set up Centronics BUSY status.
       jsrl    hangup           ; Otherwise, error.  Stop.

fcibyu:                                 ; Set UART port busy.
       sbit    cus,ups          ; Set UART input port status busy.
       jsrl    dtroff           ; Set DTR handshake appropriately.
       ifbit   eti,enui         ; Check if UART transmitter busy.
       jp      fcibyl           ;   If so, flow control will happen
                                ;     automatically.
       ifbit   xonb,uflow       ;   If not, then if XON mode is selected,
       jsrl    setuar           ;     invoke flow control routine.
fcibyl:        jmpl    upwret

                                ; Set Centronics port busy.
fcibyc:        sbit    ifcbsy,bstat    ; Set Interface Busy bit in BSTAT byte.
       sbit    cbusy,cps        ; Set BUSY bit in Centronics Port Status byte.
       jsrl    setcen           ; Change Centronics port control latch
                                ;   accordingly.
       sbit    cmdemp,curcmd    ; Set command byte empty (end of command).
       jmpl    upwret
```

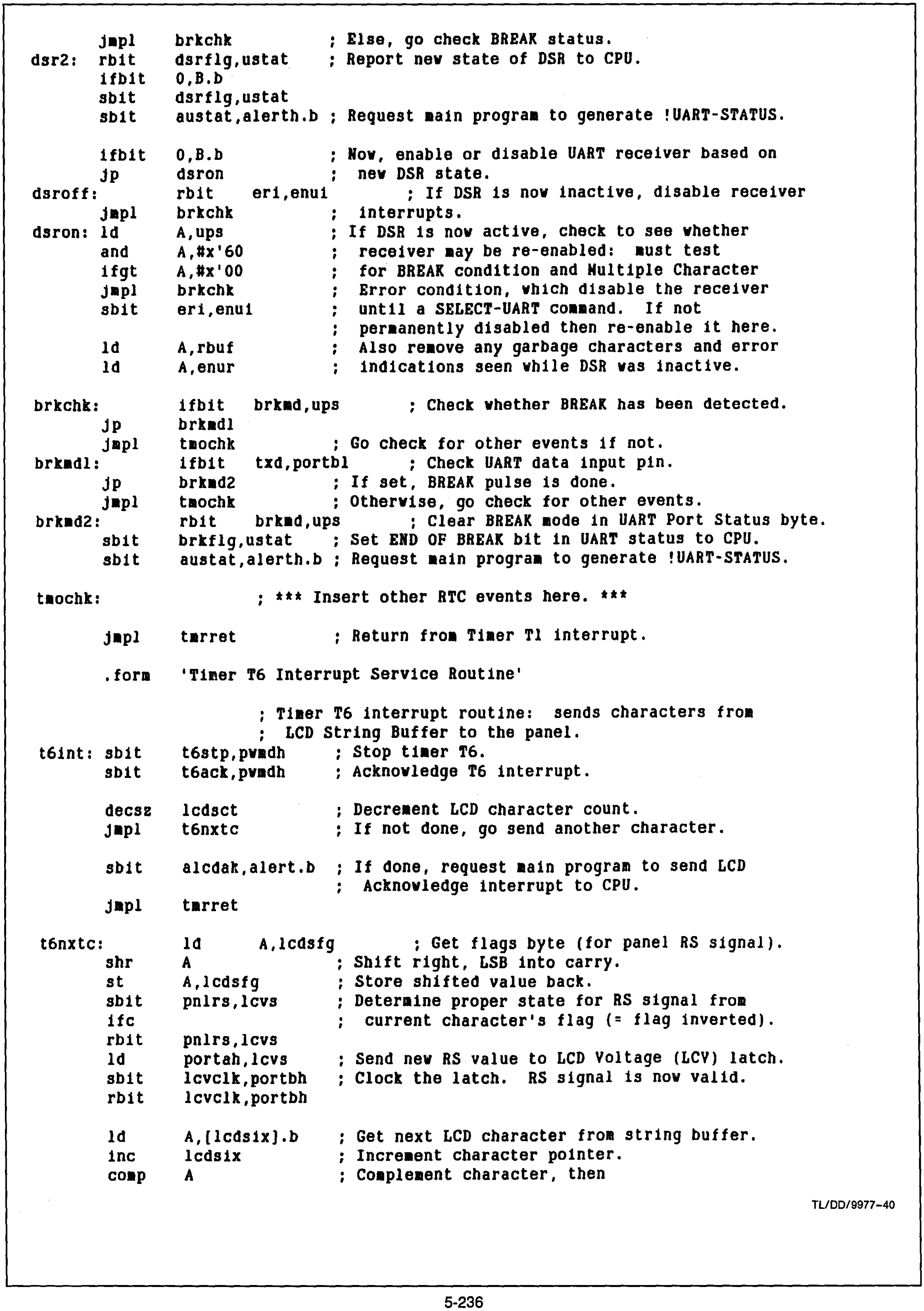
TL/DD/9977-37

```
                                ; First byte of SET-CENT-STS command.
fcscst:         ld      numexp,#1       ; Set up to expect one more byte.
        jmpl    upwret

                                ; First byte of SET-CONTRAST command.
fcslcv:         ld      numexp,#1       ; Set up to expect one more byte.
        jmpl    upwret

                                ; First byte of SEND-LCD command.
fcslcd:         ld      numexp,#2       ; Set up to expect one more byte.
        sbit    getcnt,curcmd   ; Note extended collection mode in Current
                                ;  Command byte.
        jmpl    upwret

                                ; First byte of SEND-LED command.
fcsled:         ld      numexp,#1       ; Send to LED's: Set up to expect one more byte.
        jmpl    upwret

                                ; Process one-byte BEEP command.
fcbeep:         sbit    cmdemp,curcmd   ; No arguments; set CURCMD byte empty.
        sbit    t7tfn,portph    ; Enable beep tone to panel speaker.
        sbit    t0tie,tmmdl     ; Enable Timer T0 interrupt.
        ld      beepct,#19      ; Initialize duration count (approximately
                                ;  1 second, in units of Timer T0 overflows).
        jmpl    upwret

                                ; First byte of SEND-UART command.
fcsndu:         ld      numexp,#1       ; Send to UART:  Set up to expect one more byte.
        jmpl    upwret

                                ; Process one-byte TEST-UART command.
fcusts:         sbit    cmdemp,curcmd   ; No arguments;  set CURCMD byte empty.
        sbit    austat,alerth.b ; Force UART Status interrupt.
        jmpl    upwret


illc:   jsrl    hangup          ; Process illegal command codes.


                                ; Return from UPI Write interrupt.
upwret:                                 ; Restore Context
        pop     psw
        pop     A
        reti

        .form   'Timer Interrupt Handler'

        .ipt    5,tmrint        ; Declare entry point for Timer Interrupt.

tmrint:         push    A               ; Save context.
        push    B               ;
        push    psw             ;

t1poll:         ifbit   t1pnd,tmmdl     ; Poll for Timer T1 interrupt (Real-Time Clock).
        jmpl    t1int           ; If set, go service it.

t6poll:         ifbit   t6pnd,pwmdh     ; Poll for Timer T6 interrupt (LCD Panel Timing
        jmpl    t6int           ;  Interrupt).
```

TL/DD/9977-38

```
t0poll:        ifbit   t0pnd,tmmdl     ; Poll for Timer T0 interrupt (Beep Duration).
       jp      t0pdg           ; If set, check the Enable bit;  T0 is not
       jp      t0notp          ;  always enabled to interrupt when it runs.
t0pdg: ifbit   t0tie,tmmdl     ; If enable is also set, then go service T0.
       jmpl    t0int
t0notp:                                ; (This label is deliberately here.)

noint: jsrl    hangup          ; Error:  no legal timer interrupt pending.

       .form   'Timer T1 Interrupt Service Routine'

t1int: sbit    t1ack,tmmdl     ; Acknowledge T1 interrupt.
       ifbit   rtcenb,rtevs    ; Check if RTC interrupts are enabled.
       jp      t1int1
       jmpl    kbdchk          ; If not, then go check other events.
t1int1:        decsz   rtccnt          ; Decrement interval value.
       jmpl    kbdchk          ; If interval has not elapsed, then go check
                               ;  for other events.
       ld      rtccnt,rtcivl   ; Reload counter value for next interval.
       ifbit   artc,alert.b    ; Check if CPU has received previous interrupt
       jp      t1rerr          ;  request;  report error if not.
       sbit    artc,alert.b    ; Set Real-Time Interrupt request to main
       jp      kbdchk          ;  program.
t1rerr:        sbit    0,dsevc         ; Signal NOTE severity.
       sbit    7,derrc         ; Signal multiple-RTC error.
       sbit    adiag,alerth.b  ; Request !DIAG interrupt from main program.

kbdchk:                                ; Check keyboard switches.
       rbit    astts,portbh    ; Enable pushbutton data to Port D.
       ld      A,portd         ; Sample pushbutton switches.
       sbit    astts,portbh    ; Disable pushbutton data to Port D.
       xor     A,#x'FF         ; Complement low-order 8 bits of A.
       x       A,swlast        ; Exchange with last sample.
       ifeq    A,swlast        ; Check if the data is stable (same as last
                               ;  sample).
       jp      kbint1
       jmpl    dsrchk          ; If not, go check other events.

kbint1:        ifeq    A,swlsnt        ; Check if the data differs from the last
                               ;  pattern sent to the CPU.
       jmpl    dsrchk          ; If not, go check other events.

       st      A,swlsnt        ; Place new pattern in "last sent" location.
       sbit    abutton,alerth.b ; Request "BUTTON-DATA" interrupt to CPU.

dsrchk:                        ; Check for status of DSR signal if mode selected.
       ifbit   usel,ups        ; Check if UART is selected.
       jp      dsr0
       jmpl    tmochk          ; If not, skip both DSR and BREAK checking.
dsr0:  ifbit   dsrb,uflow      ; Check if DSR input should be checked.
       jp      dsr1
       jmpl    brkchk
dsr1:  ld      A,#x'01         ; Initialize Accumulator to check DSR.
       ifbit   dsr,porti       ; Check current state of DSR pin.
       rbit    0,A             ; Clear LSB of A if DSR pin set.
       st      A,B             ; Register B holds DSR state (1 = DSR Ready).
       ifbit   dsrflg,ustat    ; Check last DSR state given to CPU.
       xor     A,#x'01         ; Toggle LSB of A if set.
       ifbit   0,A             ; If LSB of A is still set, then must send
       jp      dsr2            ;  UART-STATUS interrupt to CPU.
```

TL/DD/9977-39

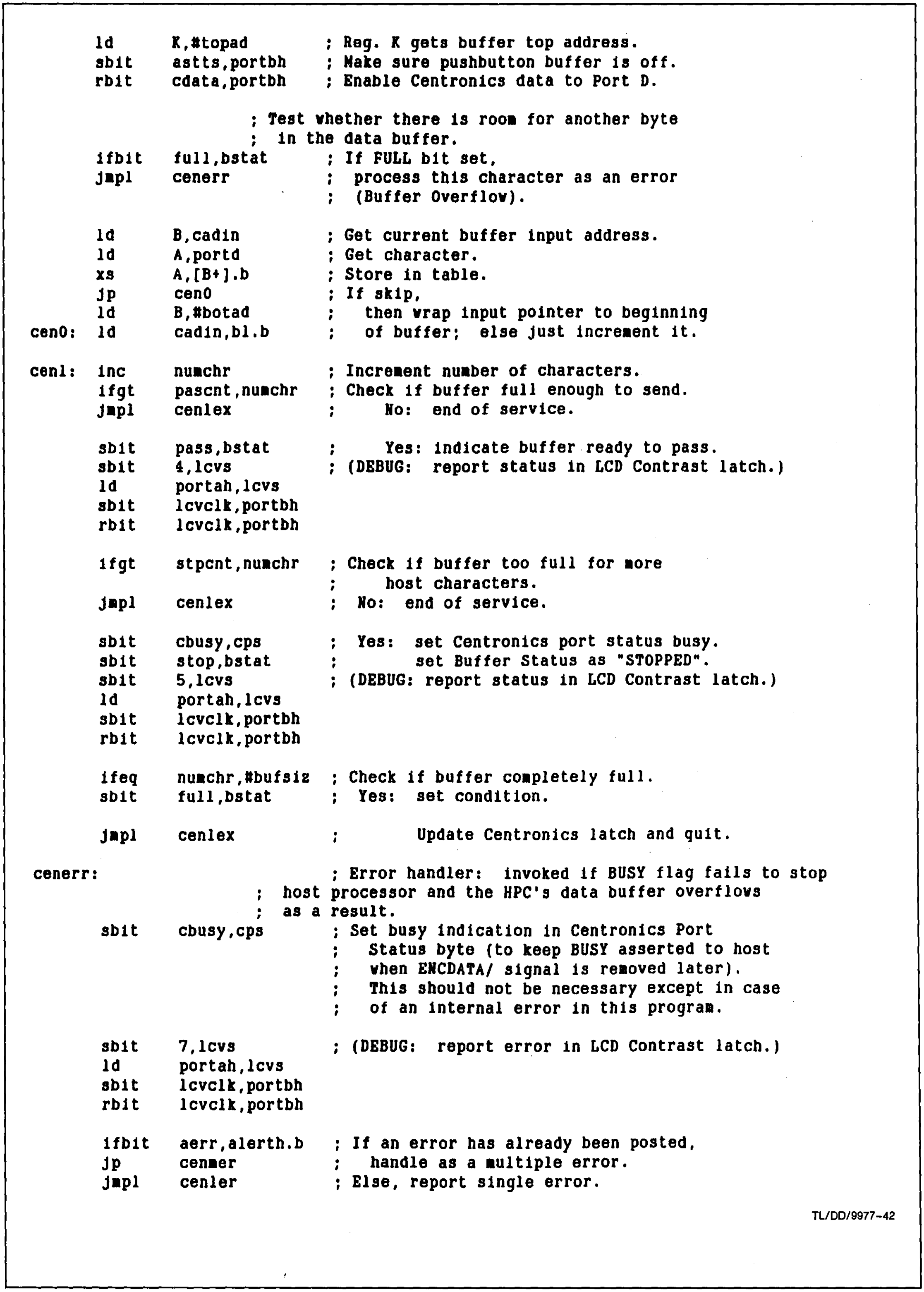


```
        jmpl    brkchk          ; Else, go check BREAK status.
dsr2:   rbit    dsrflg,ustat    ; Report new state of DSR to CPU.
        ifbit   0,B.b
        sbit    dsrflg,ustat
        sbit    austat,alerth.b ; Request main program to generate !UART-STATUS.

        ifbit   0,B.b           ; Now, enable or disable UART receiver based on
        jp      dsron           ;  new DSR state.
dsroff:         rbit    eri,enui        ; If DSR is now inactive, disable receiver
        jmpl    brkchk          ;  interrupts.
dsron:  ld      A,ups           ; If DSR is now active, check to see whether
        and     A,#x'60         ;  receiver may be re-enabled:  must test
        ifgt    A,#x'00         ;  for BREAK condition and Multiple Character
        jmpl    brkchk          ;  Error condition, which disable the receiver
        sbit    eri,enui        ;  until a SELECT-UART command.  If not
                                ;  permanently disabled then re-enable it here.
        ld      A,rbuf          ;  Also remove any garbage characters and error
        ld      A,enur          ;  indications seen while DSR was inactive.

brkchk:         ifbit   brkmd,ups       ; Check whether BREAK has been detected.
        jp      brkmd1
        jmpl    tmochk          ; Go check for other events if not.
brkmd1:         ifbit   txd,portbl      ; Check UART data input pin.
        jp      brkmd2          ; If set, BREAK pulse is done.
        jmpl    tmochk          ; Otherwise, go check for other events.
brkmd2:         rbit    brkmd,ups       ; Clear BREAK mode in UART Port Status byte.
        sbit    brkflg,ustat    ; Set END OF BREAK bit in UART status to CPU.
        sbit    austat,alerth.b ; Request main program to generate !UART-STATUS.

tmochk:                 ; *** Insert other RTC events here. ***

        jmpl    tmrret          ; Return from Timer T1 interrupt.

        .form   'Timer T6 Interrupt Service Routine'

                        ; Timer T6 interrupt routine:  sends characters from
                        ;  LCD String Buffer to the panel.
t6int:  sbit    t6stp,pwmdh     ; Stop timer T6.
        sbit    t6ack,pwmdh     ; Acknowledge T6 interrupt.

        decsz   lcdsct          ; Decrement LCD character count.
        jmpl    t6nxtc          ; If not done, go send another character.

        sbit    alcdak,alert.b  ; If done, request main program to send LCD
                                ;  Acknowledge interrupt to CPU.
        jmpl    tmrret

t6nxtc:         ld      A,lcdsfg        ; Get flags byte (for panel RS signal).
        shr     A               ; Shift right, LSB into carry.
        st      A,lcdsfg        ; Store shifted value back.
        sbit    pnlrs,lcvs      ; Determine proper state for RS signal from
        ifc                     ;  current character's flag (= flag inverted).
        rbit    pnlrs,lcvs
        ld      portah,lcvs     ; Send new RS value to LCD Voltage (LCV) latch.
        sbit    lcvclk,portbh   ; Clock the latch.  RS signal is now valid.
        rbit    lcvclk,portbh

        ld      A,[lcdsix].b    ; Get next LCD character from string buffer.
        inc     lcdsix          ; Increment character pointer.
        comp    A               ; Complement character, then
```

TL/DD/9977-40

```
       st      A,portah        ;  place it on Port A for LCD display.
       rbit    pnlclk,portbl   ; Clock it into panel.
       sbit    pnlclk,portbl
       comp    A               ; Restore A to uncomplemented form for
                               ;  test performed below.

       ld      t6,#148         ; Set up normal delay time in timer T6
                               ;  (120 microseconds).
       ifgt    A,#x'03         ; Check whether the longer delay
       jp      t6nxt2          ;  (4.9 milliseconds) is necessary.
                               ;  This happens if RS=0 and the byte sent to
       ifnc                    ;  the panel is a value of hex 03 or less.
       ld      t6,#6022        ; If so, change timer to 4.9 milliseconds.

t6nxt2:        rbit    t6stp,pwmdh     ; Start Timer T6 to time out the character.
       jmpl    tmrret          ; Return from the interrupt.

       .form   'Timer T0 Interrupt Service Routine'

t0int:                 ; Count duration of beep tone.  Restore beep signal
                       ;  to zero and re-enable switch sampling interrupt
                       ;  when done.
       sbit    t0ack,tmmdl     ; Acknowledge interrupt from Timer T0.
       decsz   beepct          ; Check whether beep time has finished.
       jmpl    tmrret          ;  No:  return from interrupt.
       rbit    t0tie,tmmdl     ;  Yes:  disable Timer T0 interrupts and
                               ;        continue.
       and     portph,#x'0F    ; Disable speaker output.
       jmpl    tmrret          ; Return from interrupt.

                       ; Common return for timer interrupt service routines.
tmrret:        pop     psw             ; Restore context.
       pop     B
       pop     A
       reti

       .form   'Centronics Port Interrupt Handler'
;
;      Centronics Port Interrupt Handler
;              (Pin I4 rising edge)
;
;      Note that cadin is an 8-bit quantity;  buffer must be
;              contiguous within the basepage area.
;

       .ipt    4,cenint

cenint:        push    psw             ; Save context.
       push    A
       push    B
       push    K

               ; Decide whether to process leading or trailing edge interrupt.
       ifbit   i4,ircd         ; Check polarity of detector.
       jmpl    cstrbl          ; Leading edge (rising on I4 pin).
       jmpl    cstrbt          ; Trailing edge (falling on I4 pin).


cstrbl:                        ; STROBE/ leading edge service routine.
```

TL/DD/9977–41

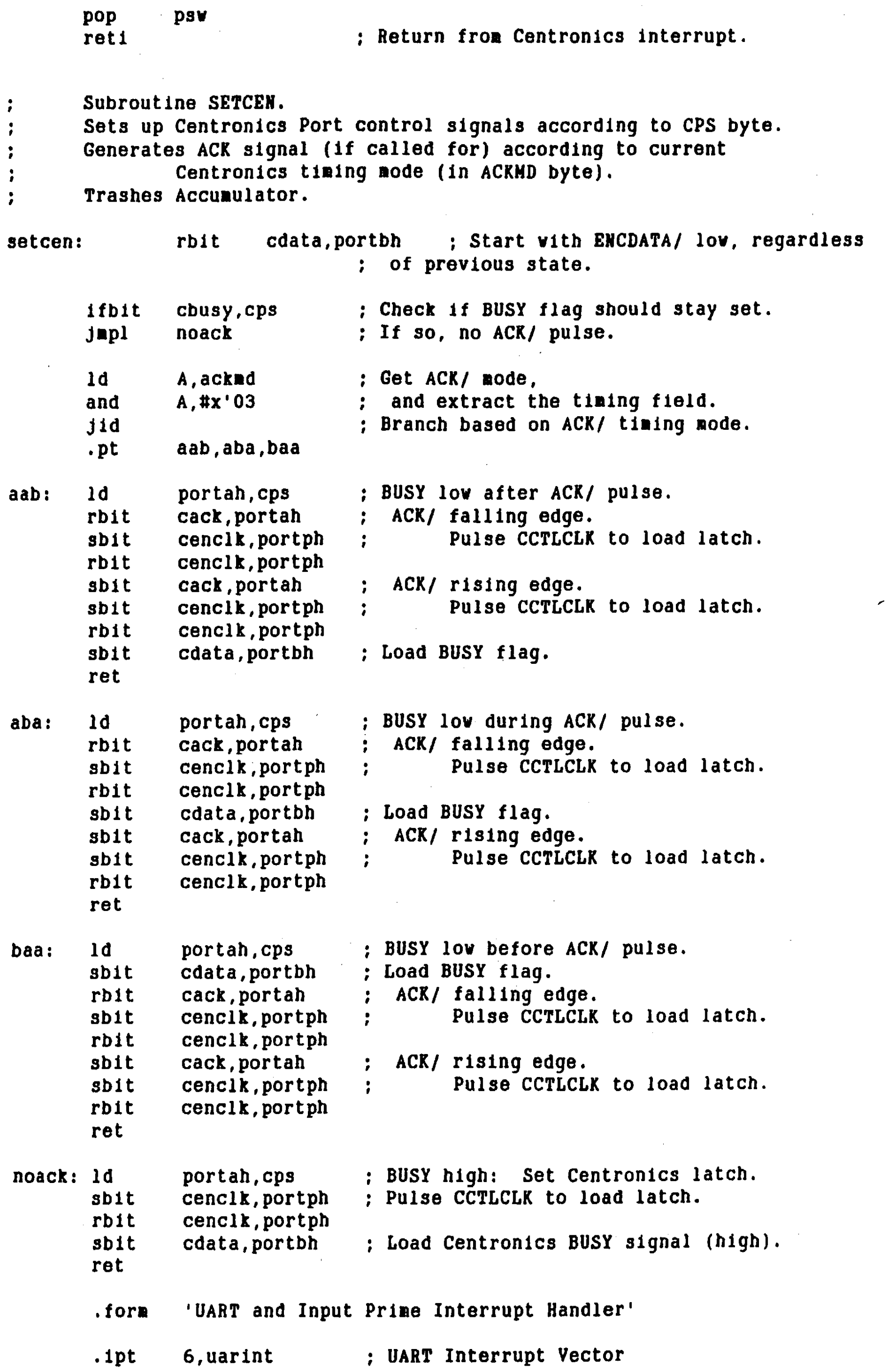


```
        ld      K,#topad        ; Reg. K gets buffer top address.
        sbit    astts,portbh    ; Make sure pushbutton buffer is off.
        rbit    cdata,portbh    ; Enable Centronics data to Port D.

                        ; Test whether there is room for another byte
                        ;  in the data buffer.
        ifbit   full,bstat      ; If FULL bit set,
        jmpl    cenerr          ;  process this character as an error
                                ;  (Buffer Overflow).

        ld      B,cadin         ; Get current buffer input address.
        ld      A,portd         ; Get character.
        xs      A,[B+].b        ; Store in table.
        jp      cen0            ; If skip,
        ld      B,#botad        ;   then wrap input pointer to beginning
cen0:   ld      cadin,bl.b      ;   of buffer;  else just increment it.

cenl:   inc     numchr          ; Increment number of characters.
        ifgt    pascnt,numchr   ; Check if buffer full enough to send.
        jmpl    cenlex          ;     No:  end of service.

        sbit    pass,bstat      ;     Yes: indicate buffer ready to pass.
        sbit    4,lcvs          ; (DEBUG:  report status in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh

        ifgt    stpcnt,numchr   ; Check if buffer too full for more
                                ;     host characters.
        jmpl    cenlex          ;  No:  end of service.

        sbit    cbusy,cps       ;  Yes:  set Centronics port status busy.
        sbit    stop,bstat      ;        set Buffer Status as "STOPPED".
        sbit    5,lcvs          ; (DEBUG: report status in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh

        ifeq    numchr,#bufsiz  ; Check if buffer completely full.
        sbit    full,bstat      ;  Yes:  set condition.

        jmpl    cenlex          ;        Update Centronics latch and quit.

cenerr:                         ; Error handler:  invoked if BUSY flag fails to stop
                        ;  host processor and the HPC's data buffer overflows
                        ;  as a result.
        sbit    cbusy,cps       ; Set busy indication in Centronics Port
                                ;   Status byte (to keep BUSY asserted to host
                                ;   when ENCDATA/ signal is removed later).
                                ;   This should not be necessary except in case
                                ;   of an internal error in this program.

        sbit    7,lcvs          ; (DEBUG:  report error in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh

        ifbit   aerr,alerth.b   ; If an error has already been posted,
        jp      cenmer          ;   handle as a multiple error.
        jmpl    cenler          ; Else, report single error.
```

TL/DD/9977-42

```
cenmer:         sbit    bufovf,errfgs   ; OR in the buffer overflow condition.
        sbit    errovf,errfgs   ; Update error conditions byte to also report
                                ;  an error overflow.
        rbit    i4,enir         ; Disable STROBE interrupt until re-initialized
                                ;  by CPU.
        jmpl    cenlex          ; Return from the interrupt.

cenler:         sbit    aerr,alerth.b   ; Signal an error.
        ld      errfgs,#x'10    ; Report buffer overflow as reason.
        ld      errchr,portd    ; Place character in ERRCHR slot for report to
                                ;   CPU.
        ld      fshlim,numchr   ; Establish limit on future flushes.
        jmpl    cenlex          ; Return from the interrupt.


cenlex:                         ; Exit from Centronics STROBE/ leading edge.
        ld      A,cps           ; Prepare to keep BUSY active when ENCDATA/
        sbit    cbusy,A.b       ;  is removed.
        st      A,portah        ; Send CPS byte (with BUSY set) to Centronics
                                ;  status latch.
        sbit    cenclk,portph   ;  (Pulse latch strobe.)
        rbit    cenclk,portph
        sbit    cdata,portbh    ; Remove Centronics data enable;  loads BUSY
                                ;  signal with a "1".
        rbit    i4,ircd         ; Set I4 strobe pin to trigger on STROBE/
                                ;  trailing edge.
        ifbit   i4,porti        ; Check if strobe has already gone away.
        jmpl    cenend          ;  If not, just return (no ACK/ pulse).
                                ;   The "cstrbt" routine will be activated then
                                ;   whenever STROBE/ goes away, by means of the
                                ;   I4 interrupt.
        jmpl    cstrbt          ;  If so, there is a very small possibility
                                ;   that the interrupt request may have been
                                ;   lost due to it changing while the polarity
                                ;   bit in IRCD was being changed above.
                                ;   Jump to trailing edge service routine
                                ;   directly from here.


cstrbt:                         ; Centronics STROBE/ trailing edge.

        sbit    i4,ircd         ; Set up for leading edge detection again.
        ld      irpd,#x'EF      ; Clear interrupt I4, in case the leading edge
                                ;  routine came directly here.  (No hardware
                                ;  clear of the request occurs in that case.)
        jmpl    cenupd          ; Go update Centronics port, with ACK/ pulse
                                ;  if necessary.

;       Return from interrupt.

        ; With Centronics Port update.
cenupd:         jsrl    setcen          ; Update Centronics Control signals
                                ;  from CPS byte.

        ; Without Centronics Port update.
cenend:         pop     K               ; Restore context from stack and return from
                                ;  Centronics interrupt.
        pop     B
        pop     A
```

TL/DD/9977-43



```
        pop     psw
        reti                    ; Return from Centronics interrupt.


;       Subroutine SETCEN.
;       Sets up Centronics Port control signals according to CPS byte.
;       Generates ACK signal (if called for) according to current
;               Centronics timing mode (in ACKMD byte).
;       Trashes Accumulator.

setcen:         rbit    cdata,portbh    ; Start with ENCDATA/ low, regardless
                                ;  of previous state.

        ifbit   cbusy,cps       ; Check if BUSY flag should stay set.
        jmpl    noack           ; If so, no ACK/ pulse.

        ld      A,ackmd         ; Get ACK/ mode,
        and     A,#x'03         ;  and extract the timing field.
        jid                     ; Branch based on ACK/ timing mode.
        .pt     aab,aba,baa

aab:    ld      portah,cps      ; BUSY low after ACK/ pulse.
        rbit    cack,portah     ;  ACK/ falling edge.
        sbit    cenclk,portph   ;       Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        sbit    cack,portah     ;  ACK/ rising edge.
        sbit    cenclk,portph   ;       Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        sbit    cdata,portbh    ; Load BUSY flag.
        ret

aba:    ld      portah,cps      ; BUSY low during ACK/ pulse.
        rbit    cack,portah     ;  ACK/ falling edge.
        sbit    cenclk,portph   ;       Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        sbit    cdata,portbh    ; Load BUSY flag.
        sbit    cack,portah     ;  ACK/ rising edge.
        sbit    cenclk,portph   ;       Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        ret

baa:    ld      portah,cps      ; BUSY low before ACK/ pulse.
        sbit    cdata,portbh    ; Load BUSY flag.
        rbit    cack,portah     ;  ACK/ falling edge.
        sbit    cenclk,portph   ;       Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        sbit    cack,portah     ;  ACK/ rising edge.
        sbit    cenclk,portph   ;       Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        ret

noack:  ld      portah,cps      ; BUSY high:  Set Centronics latch.
        sbit    cenclk,portph   ; Pulse CCTLCLK to load latch.
        rbit    cenclk,portph
        sbit    cdata,portbh    ; Load Centronics BUSY signal (high).
        ret

        .form   'UART and Input Prime Interrupt Handler'

        .ipt    6,uarint        ; UART Interrupt Vector
```

TL/DD/9977-44

```
                  ;  This interrupt can indicate any of three conditions:
                  ;    1)  A character has been sent, and the transmitter
                  ;           is again ready (label "uarout").
                  ;    2)  A character has been received (label "uartin").
                  ;    3)  A Centronics INPUT PRIME event has been detected
                  ;           (label "uarprm").


uarint:           push    psw
        push      A
        push      B
        push      K
        push      X

        ifbit     usel,ups        ; Check if UART selected.
        jmpl      uarchr          ; If so, go process a character interrupt.
        ifbit     enprm,cps       ; Check if PRIME interrupt enabled
        jmpl      uarprm          ;  from Centronics port.  If so,
                                  ;  this means that the Centronics port
                                  ;  is selected, and it must be a PRIME
                                  ;  event.
        jsrl      hangup          ; Else, there is an error.  Stop.

uarchr:           ifbit   rbfl,enu        ; Check for Receiver interrupt.
        jmpl      uartin          ; Go process input character if so.
        ifbit     tbmt,enu        ; Check for Transmitter interrupt.
        jmpl      uarout          ; Go process output interrupt if so.
        jsrl      hangup          ; Else, there is an error.  Stop.


        .form     'UART Output Routine'

uarout:                           ; Here, the interrupt is because a character has just
                      ;   been sent and the transmitter buffer is now empty.
        ifbit     icpu,ups        ; Check if the CPU needs to be informed.
        jmpl      uicpu
        jmpl      unicpu
uicpu:  sbit      auack,alerth.b  ; Request main program to interrupt CPU for
                                  ;  UART acknowledge.
        rbit      icpu,ups        ; Reset "Interrupt CPU" status on UART.
        jmpl      unicpu          ; Continue processing of interrupt.

unicpu:           ifbit   xonb,uflow      ; If XON mode selected,
        jsrl      setuar          ;  check UART handshake status and take any
                                  ;  appropriate action.
        jmpl      uarret          ; Return.


        .form     'UART Input Routine'

uartin:                                   ; UART data input routine.

        ld        A,enur          ; Get image of error flags and RBIT9.
        ld        uinchr,rbuf     ; Get character.
        st        A,enrimg        ; Save image of ENUR for further processing.
                                  ; Check for hardware-detected errors.
        and       A,#x'C0         ; Mask for error bits (Overrun/Framing).

        ld        X,uinchr        ; Prepare for parity check.
```

TL/DD/9977-45

```
          ld      B,#evntbl       ; Initialize B to point to Even Parity table.
          x       A,uframe        ; Parity processing depends on selected
                                  ;  frame format, so branch to proper
          jid                     ;  parity processing routine.
          .pt     uiod8,uiev8,unopar,unopar
          .pt     uiod7,uiev7,uiod7,uiev7

                          ; Processing for 8-bit characters with parity.
uiod8:    ld      B,#oddtbl       ; For odd processing, change parity table base.
uiev8:    x       A,uframe        ; Recover cumulative errors in accumulator.
          ifbit   frm,A.b         ; Check for BREAK condition:  if framing error,
          jp      ufer8
          jp      u8nbrk
ufer8:    ifgt    uinchr,#0       ;  and data field is all zeroes,
          jp      u8nbrk
          ifbit   rbit9,enrimg    ;  and 9th bit also zero,
          jp      u8nbrk
          ifbit   onebrk,ups      ;  then check if this is the second
          jp      u82brk          ;   consecutive BREAK.
          sbit    onebrk,ups      ;  If not, then flag only the framing error,
          jp      u8dopr          ;   and do not report break status yet.
u82brk:           sbit    brk,A.b         ;  If so, then set Break bit in error image and
          rbit    eri,enui        ;   disable UART receiver until re-selected.
          sbit    brkmd,ups       ;   Also show receiver disabled in UPS byte.
u8nbrk:           rbit    onebrk,ups
u8dopr:           ifbit   X,[B].b         ; Check parity of 8-bit character.  Set "par"
          sbit    par,A.b         ;  bit of Accumulator if it would be incorrect
                                  ;  without parity bit.
          ifbit   rbit9,enrimg    ; Check parity bit for 8-bit character.  Toggle
          xor     A,#x'20         ;  parity error indication if set.
uinpok:           ifeq    A,#x'00         ; Branch based on presence of error.
          jmpl    uingd
          jmpl    uinerc

                          ; Processing for 7-bit characters with parity.
uiod7:    ld      B,#oddtbl       ; For odd processing, change parity table base.
uiev7:    x       A,uframe        ; Recover cumulative errors in accumulator.
          ifbit   frm,A.b         ; Check for BREAK condition:  if framing error,
          jp      ufer7
          jp      u7nbrk
ufer7:    ifgt    uinchr,#0       ;  and data field is all zeroes (incl. parity),
          jp      u7nbrk
          ifbit   onebrk,ups
          jp      u72brk
          sbit    onebrk,ups
          jp      u7dopr
u72brk:           sbit    brk,A.b         ;  then set Break bit in error image and
          rbit    eri,enui        ;  disable receiver.
          sbit    brkmd,ups       ;  Also show receiver disabled in UPS byte.
u7nbrk:           rbit    onebrk,ups
u7dopr:           rbit    7,uinchr        ; Seven-bit data:  clear parity bit in memory.
          ifbit   X,[B].b         ; Perform bit-table lookup:  1 means error.
          jp      uipe7
          jmpl    uinpok
uipe7:    sbit    par,A.b         ; Set parity error indication in A.
          jmpl    uinerc

                          ; For 8-bit character frames with no parity:
unopar:           x       A,uframe        ; Restore frame value to UFRAME, and continue
                                  ;  (no parity check in these modes).
```

TL/DD/9977-46

```
        ifbit   frm,A.b         ; Check for BREAK condition:  if framing error,
        jp      uferr
        jp      unbrk
uferr:  ifgt    uinchr,#0       ;  and data field is all zeroes (incl. parity),
        jmp     unbrk
        ifbit   onebrk,ups      ;  then BREAK condition:  if previous character
        jmp     un2brk
        sbit    onebrk,ups      ;    was not a BREAK, then just note this one.
        jp      unobrk
un2brk:         sbit    brk,A.b         ;  If it was, then set Break bit in error image
        rbit    eri,enui        ;  and disable receiver.
        sbit    brkmd,ups       ;  Also show receiver disabled in UPS byte.
unbrk:  rbit    onebrk,ups
unobrk:         jmpl    uinpok

uingd:                  ; Here, a good character was received.  Start buffer
                        ;  processing.
        ld      A,uinchr        ; Get character again.
        ld      K,#topad        ; Reg. K gets buffer top address.

                        ; Test whether there is room for another byte
                        ;  in the data buffer.
        ifbit   full,bstat      ; If FULL bit set,
        jmpl    uinerf          ;  process this character as an error
                                ;  (Buffer Overflow).

        ld      B,cadin         ; Get current buffer input address.
        xs      A,[B+].b        ; Store character in table.
        jp      uin0            ; If skip,
        ld      B,#botad        ;   then wrap input pointer to beginning
uin0:   ld      cadin,bl.b      ;   of buffer;  else just increment it.

uin1:   inc     numchr          ; Increment number of characters.
        ifgt    pascnt,numchr   ; Check if buffer full enough to send.
        jmpl    uinex           ;     No:  end of service.

        sbit    pass,bstat      ;     Yes: indicate buffer ready to pass.
        sbit    4,lcvs          ; (DEBUG: report status in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh

        ifgt    stpcnt,numchr   ; Check if buffer too full for more
                                ;     host characters.
        jmpl    uinex           ;  No:  end of service.

        sbit    cus,ups         ;  Yes:  set UART input port status busy.
        sbit    stop,bstat      ;        set Buffer Status as "STOPPED".
        jsrl    dtroff          ;        set DTR handshake appropriately.
        ifbit   eti,enui        ;        check if UART transmitter busy.
        jp      uin2
        ifbit   xonb,uflow      ;          if not, then if XON mode selected,
        jsrl    setuar          ;          then invoke flow control routine.
                                ;          (otherwise it will happen on next
                                ;           UART transmitter interrupt
                                ;           automatically).
uin2:
        sbit    5,lcvs          ; (DEBUG: report status in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
```

TL/DD/9977-47

5

```
        rbit    lcvclk,portbh

        ifeq    numchr,#bufsiz  ; Check if buffer completely full.
        sbit    full,bstat      ; Yes:  set condition.

        jmpl    uinex

uinerc:                         ; Character error handler.
        ifbit   aerr,alerth.b   ; If an error has already been posted,
        jp      uinmce          ;   handle as a multiple error.
        jmpl    uinlce          ; Else, report single error.

uinmce:         sbit    errovf,errfgs   ; Update error conditions byte to also report
                                ;  a lost error.
        or      errfgs,A.b      ; OR in the errors from this character.
        sbit    cus,ups         ;  Yes:  set UART input port status busy.
        ifbit   eti,enui        ;        check if UART transmitter busy.
        jp      uinmc2
        ifbit   xonb,uflow      ;          if not, then if XON mode selected,
        jsrl    setuar          ;          then invoke flow control routine.
                                ;          (otherwise it will happen on next
                                ;           UART transmitter interrupt
                                ;           automatically).
uinmc2:         jsrl    dtroff          ; Remove DTR handshake if flow mode requires it.
        rbit    eri,enui        ; Disable UART input interrupt until
                                ;  re-initialized by CPU.
        sbit    mcemd,ups       ;  Also flag receiver disabled in UPS byte.
        jmpl    uinex           ; Return from the interrupt.

uinlce:
        sbit    aerr,alerth.b   ; Request CPU interrupt from main program.
        st      A,errfgs        ; Report error flags from Accumulator.
        ld      errchr,uinchr   ; Report error character.
        ld      fshlim,numchr   ; Establish limit on future flushes.
        jmpl    uinex           ; Return from the interrupt.

uinerf:                         ; FULL error handler:  invoked if HPC's data buffer
                        ;  overflows.

        sbit    7,lcvs          ; (DEBUG:  report error in LCD Contrast latch.)
        ld      portah,lcvs
        sbit    lcvclk,portbh
        rbit    lcvclk,portbh

        ifbit   aerr,alerth.b   ; If an error has already been posted,
        jp      uinmef          ;   handle as a multiple error.
        jmpl    uinlef          ; Else, report single error.

uinmef:         sbit    bufovf,errfgs   ; Signal buffer overflow as another error.
        sbit    errovf,errfgs   ; Update error conditions byte to also report
                                ;  a lost error.
        sbit    cus,ups         ;  Set UART input port status busy.
        rbit    luss,ups        ;  (This is done to force flow control action.)
        ifbit   eti,enui        ;  Check if UART transmitter busy.
        jp      uinme2
        ifbit   xonb,uflow      ;  If not, then if XON mode selected,
        jsrl    setuar          ;   then invoke flow control routine.
                                ;   (otherwise it will happen on next
                                ;    UART transmitter interrupt automatically).
uinme2:         jsrl    dtroff          ; Remove DTR handshake if flow mode needs it.
```

TL/DD/9977-48

```
       rbit     eri,enui        ; Disable UART input interrupt until
                                ;  re-initialized by CPU.
       sbit     mcemd,ups       ;  Also flag receiver disabled in UPS byte.
       jmpl     uinex           ; Return from the interrupt.

uinlef:         sbit    aerr,alerth.b   ; Signal an error.
       ld       errfgs,#x'10    ; Report buffer overflow as reason.
       ld       errchr,uinchr   ; Place character in ERRCHR slot for report to
                                ;   CPU.
       ld       fshlim,numchr   ; Establish limit on future flushes.
       sbit     cus,ups         ;  Set UART input port status busy.
       rbit     luss,ups        ;  (This is done to force flow control action.)
       ifbit    eti,enui        ;  Check if UART transmitter busy.
       jp       uinlf2
       ifbit    xonb,uflow      ;  If not, then if XON mode selected,
       jsrl     setuar          ;   then invoke flow control routine.
                                ;   (otherwise it will happen on next
                                ;    UART transmitter interrupt automatically).
uinlf2:         jsrl    dtroff         ; Remove DTR handshake if flow mode needs it.
       jmpl     uinex           ; Return from the interrupt.


uinex:                  ; Exit from UART input character processing.
       jmpl     uarret          ; Return.

       ; Parity Bit Lookup Table

evntbl:         .byte  X'96,X'69,X'69,X'96,X'69,X'96,X'96,X'69
       .byte  X'69,X'96,X'96,X'69,X'96,X'69,X'69,X'96
oddtbl:         .byte  X'69,X'96,X'96,X'69,X'96,X'69,X'69,X'96
       .byte  X'96,X'69,X'69,X'96,X'69,X'96,X'96,X'69
       .byte  X'96,X'69,X'69,X'96,X'69,X'96,X'96,X'69
       .byte  X'69,X'96,X'96,X'69,X'96,X'69,X'69,X'96
;
; A one in the table means incorrect parity for the mode,
;  the mode being expressed as the base address (evntbl or oddtbl).
       .form    'Centronics INPUT PRIME'


                        ; Centronics INPUT PRIME service.
uarprm:         sbit    aprime,alert.b  ; Set PRIME bit in Alert mailbox to Main prog.
       sbit     cbusy,cps       ; Set BUSY bit in Centronics status byte.
       jsrl     setcen          ; Go set up Centronics port itself.
       rbit     uart,enir       ; Disable interrupt until it goes away.
       jmpl     uarret          ; Return.


uarret:         pop     X       ; Common return from UART interrupt.
       pop      K
       pop      B
       pop      A
       pop      psw
       reti

       .form    'Subroutine to Wait for OBUF Empty'

;      RDWAIT subroutine:  waits until the CPU has read a byte from the
;                          UPI interface.

rdwait:         ifbit   rdrdy,upic      ; Check to see if OBUF register is full.
```

TL/DD/9977-49

```
        ret
        jp      rdwait

        .form   'Write to Panel Subroutine'

                        ; Write Panel subroutine.
                        ; Used only at initialization or to report a
                        ; fatal protocol error, since it performs
                        ; the timing delay using timer T6 without interrupts.
                        ;   (Panel RS signal must be set up previously in the
                        ;    LCV latch by the calling routine.)

wrpnl:  comp    A               ; Complement value for bus.
        st      A,portah        ; Put value on panel bus.
        rbit    pnlclk,portbl   ; Set Panel Clock low,
        sbit    pnlclk,portbl   ;  then high again;
                                ;  pulse width approx.
                                ;  1.2 microsec.

                        ; Wait for another
                        ;  4.9 milliseconds (twice).
        ld      t6,#13000       ; Twice 4.9 milliseconds.
        rbit    t6stp,pwmdh     ; Start timer T6.
wrplp:  ifbit   t6pnd,pwmdh     ; Wait for PND to be set.
        jp      wrpgo
        jp      wrplp
wrpgo:  sbit    t6stp,pwmdh     ; Stop timer T6.
        sbit    t6ack,pwmdh     ; Clear T6 PND bit.
        ret                     ; Return from subroutine.

        .form   'Set up UART flow control/output'

setuar:                         ; Subroutine SETUAR:  checks status of UART output
                        ;   section, and initiates a transfer if needed.
        ld      A,ups           ; Check if UART handshake status needs update.
        and     A,#x'03
        shl     A
        .odd
        jidw
        .ptw    usmat,usnmat,usnmat,usmat

                        ; Here, UART status last sent does not match
                        ;  current status.  Needs flow control action.
usnmat:
        ifbit   cus,ups
        jmpl    ustop
ugo:    ld      X,#xon          ; Get XON (Control-Q) code.
        jsrl    uecsnd          ; Format it and send.
        rbit    luss,ups
        jmpl    sturet          ; Return.
ustop:  ld      X,#xoff         ; Get XOFF (Control-S) code.
        jsrl    uecsnd          ; Format it and send.
        sbit    luss,ups
        jmpl    sturet          ; Return.

usmat:                  ; No flow control needed.  Check if CPU character is
                        ;  waiting to be sent.
        ifbit   schr,ups
        jmpl    uscpc
```

TL/DD/9977-50

```
unopnd:                           ; Here, no characters pending to be sent.  Turn off
                         ;  transmitter interrupt and return.
        rbit    eti,enui          ; Turn off transmitter interrupts.
        jmpl    sturet            ; Return.

uscpc:                   ; Here, a character is waiting to be sent from CPU.
        ld      X,uschr           ; Get character.
        jsrl    uecsnd            ; Format character for current frame and send.
        rbit    schr,ups          ; Remove character send request.
        sbit    icpu,ups          ; Set CPU interrupt request on completion.
        jmpl    sturet            ; Return.

sturet:         ret                       ; Return from subroutine.

        .form   'Format and transmit UART character'

uecsnd:                           ; Subroutine to encode a character according to the
                         ;   currently-selected frame format and send it.
                         ;   Character is passed in Register X.
        ld      B,#evntbl
        rbit    xbit9,enu
        ld      A,uframe          ; Jump based on frame format.
        jid
        .pt     su8odd,su8evn,su8,su8
        .pt     su7odd,su7evn,su7odd,su7evn

su8odd:         ld      B,#oddtbl
su8evn:         ifbit   X,[B].b
        sbit    xbit9,enu
        ld      tbuf,X.b
        sbit    eti,enui
        ret

su7odd:         ld      B,#oddtbl
su7evn:         ifbit   X,[B].b
        xor     X.b,#x'80         ; Toggle parity to ignore bad top bit.
        ld      tbuf,X.b
        sbit    eti,enui
        ret

su8:    ld      tbuf,X.b
        sbit    eti,enui
        ret

        .form   'DTR Handshake Routines'

                         ; Subroutine DTROFF - Sets printer not ready using DTR.
dtroff:         ifbit   dtrbl,uflow     ; Action taken depends on UFLOW mode.
        jp      doff              ; If DTR is in a permanent state, return.
        ret
doff:   ifbit   dtrb0,uflow
        jp      d2off
        sbit    dtr,portbl        ; For low-active DTR mode.
        ret
d2off:  rbit    dtr,portbl        ; For high-active DTR mode.
        ret

                         ; Subroutine DTRON - Sets printer ready using DTR.
dtron:  ifbit   dtrbl,uflow       ; Action taken depends on UFLOW mode.
        jp      dton              ; If DTR is in a permanent state, return.
```

TL/DD/9977-51

```
       ret
dton:  ifbit   dtrb0,uflow
       jp      d2on
       rbit    dtr,portbl      ; For low-active DTR mode.
       ret
d2on:  sbit    dtr,portbl      ; For high-active DTR mode.
       ret

       .end    start
```

TL/DD/9977-52

Interfacing A Serial EEPROM to the National HPC16083

National Semiconductor
Application Note 552
Brian Marley

ABSTRACT

This application note describes how to interface the HPC16083 High-Performance microController to a MICROWIRE™ serial EEPROM (Electrically Erasable Programmable Read-Only Memory) device. The technique uses interrupt-driven scheduling from one of the eight on-chip timers, and so can run in the "background", sharing the HPC gracefully with other control applications running at the same time. Source code is included.

1.0 INTRODUCTION

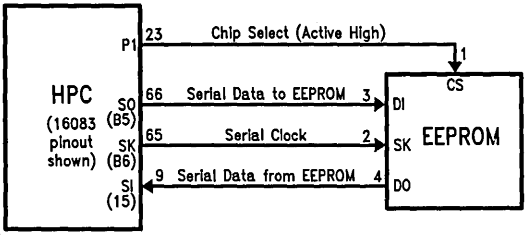
It is often the case in control-oriented applications that a piece of equipment, on being installed, must be set up with certain semi-permanent configuration mode settings. In the past, jumpers and switches have been the methods used, but in recent years these have been largely supplanted by EEPROM devices, which hold more information and are not prone to mechanical problems. In addition, the presence of an EEPROM allows certain information about the status of the equipment (for example, in printers, a page or character count for monitoring the "age" of the cartridge or print head) to be stored to assist in maintenance.

The most cost-effective type of EEPROM device is one with a serial interface, such as the 256-bit NMC9306 (COP494) or the 1024-bit NMC9345 (COP495). These reside in an 8-pin DIP package, and require only four connections (besides V_{CC} and Ground). These connections are provided by the HPC family of High-Performance Microcontrollers, on a serial port called the MICROWIRE/PLUS™ Interface.

Because one of the HPC's strong suits is Concurrent Control applications (applications in which several control tasks are executing simultaneously, scheduled by interrupts), the code given in this exercise is written to be completely interrupt-driven as well. Instead of timing events with software loops, interrupts from HPC Timer T5 are used both to signal the end of each MICROWIRE transfer and to time the ERASE and WRITE pulse durations for the EEPROM.

2.0 CONNECTIONS AND COMMANDS

The connection between the HPC and the EEPROM device is a completely traditional MICROWIRE connection, as shown in *Figure 1.* The SI (Serial Input), SO (Serial Output) and SK (Serial Clock) signals of the HPC connect directly to the DO, DI and SK pins of the EEPROM, respectively. The EEPROM's required Chip Select signal (CS: active high) could come from any port bit of the HPC, but the P1 pin of Port P was chosen because Port P pins present zeroes on reset (instead of floating), and this will automatically deselect the EEPROM.



TL/DD/9978–1

FIGURE 1. MICROWIRE/PLUS Connections

To communicate with the EEPROM, the signal CS (pin P1) is set high, and then each 8-bit serial transfer is triggered by writing a value to the HPC's eight-bit SIO register, which is effectively just a shift register. The data placed into the SIO register is shifted out, most-significant bit first, and eight clock pulses are presented on the SK pin corresponding to each shift. Serial data is simultaneously accepted from the SI pin, and at the end of the eight clock pulses the SIO register has been changed to reflect the value presented by the EEPROM (if any). The timing involved in a single MICROWIRE transfer is shown in *Figure 2*.

While reading from the EEPROM, the value written to SIO doesn't matter, since it is ignored by the EEPROM. The CS signal must be active throughout a command (which may involve more than one eight-bit transfer), and it must be set inactive between commands for at least one microsecond. Also, the time between an ERASE or WRITE command and the following command (as measured by the amount of time the CS signal remains low between them) determines the length of the corresponding ERASE or WRITE pulse within the EEPROM chip. These pulse widths have strict limits which, if exceeded, can damage some EEPROMs.

EEPROM commands are 8-bit values. However, they must start with an additional "1" bit (the Start bit), and READ commands require a trailing "pad" bit, to provide timing control for the access. Since HPC MICROWIRE transfers must consist of integral numbers of 8-bit transfers, at least two such transfers must be used per command.

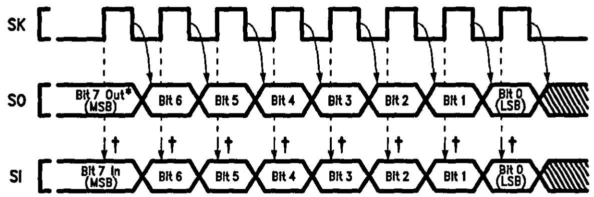
Note that the formats shown below (with 6 address bits) support an EEPROM with up to 1K bits (64 16-bit words). To use a 256-bit EEPROM, one would not specify an address greater than binary 001111, because the two most-significant address bits are ignored by the EEPROM.

2.1 Read Commands

Reading a 16-bit word from the EEPROM is accomplished with a single READ command. For the READ command, the format is:

```
        0 0 0 0 0 0 1 1  0 A A A A A A 0
        |- - - - - -| |                |
             |        start bit        pad bit
  leading zeroes
    (ignored)
```

where the bits marked "A" constitute the address of the EEPROM word to be accessed. These two command transfers are followed by two additional 8-bit transfers, in which the 16 bits of data from the addressed EEPROM word are read by the HPC (most significant bit first).



TL/DD/9978-2

*This bit becomes valid immediately when the transmitting device loads its SIO register. The HPC guarantees it to be valid for at least 1 full SK period before the rising edge of the first SK pulse presented.

† Arrows indicate points at which SI is sampled.

FIGURE 2. MICROWIRE/PLUS Transfer

Master presents eight pulses on SK pin; each pulse transfers one bit in and out.

2.2 Write Commands

To write data into the EEPROM, a sequence of commands is entered:

```
an EWEN command (Erase/Write Enable):
        0 0 0 0 0 0 0 1     0 0 1 1 0 0 0 0
an ERASE command:
        0 0 0 0 0 0 0 1     1 1 A A A A A A
              ("A" = Address bits,
                 most-significant bit first)
a pause of 16 to 25 milliseconds, with CS
low,
a WRITE command:
        0 0 0 0 0 0 0 1     0 1 A A A A A A
        D D D D D D D D     D D D D D D D D
("A" = Address bits,
              "D" = Data bits,
                 most-significant bit first)
a pause of 16 to 25 milliseconds, with CS
low,
and, finally, an EWDS command (Erase/Write
Disable):
        0 0 0 0 0 0 0 1     0 0 0 0 0 0 0 0
```

3.0 LISTING AND COMMENTARY

The listing provided shows three necessary segments of a program to access the EEPROM device:

1) initialization of the MICROWIRE/PLUS port on the HPC,
2) two program fragments of a Main Program which would initiate a Read or a Write operation,
3) an interrupt service routine (attached to Timer T5) which actually performs the transfers.

3.1 Initialization

On receiving a Reset signal, the HPC begins execution at the label "start". It loads the PSW register (to select 1 Wait state), and then removes all interrupt enables.

At label "sram", all RAM within the HPC is initialized to zero.

At "suwire", the MICROWIRE/PLUS interface pins are initialized. The MICROWIRE/PLUS interface is then set to the CKI/128 bit rate (125 KHz clocking at 16 MHz crystal frequency). The internal interface is not completely cleared by the Reset signal, so the firmware must set it up and wait (at label "suwlp") for the interface to become ready. Once this has been done, a byte of all zeroes is sent to the EEPROM to terminate any Write operation that might have been in progress when the Reset was received.

At "tminit", the timers T1–T7 are stopped and any interrupts pending from timers T0–T7 are cleared. The individual timer interrupt enables are then cleared.

The program then continues to label "minit", which initializes the variables in the HPC's on-chip RAM to their proper contents.

At label "runsys", the necessary interrupt is enabled (from the timers), and execution continues to the body of the Main Program.

There follow now two fragments of illustrative main program code which can be used to trigger the process of reading and writing the EEPROM.

3.2 Reading

The main program and interrupt routines given here enable reading from one to eight bytes from the EEPROM, starting at the beginning of any word.

At label "rnvr", an EEPROM READ command is constructed from the EEPROM starting address and placed in the variable "nvrcmd". The number of bytes to be transferred is placed in the variable "nvrnum". Control is then transferred to the label "nvrx", where Timer T5 is set up to generate scheduling interrupts for reading data from the EEPROM.

The variable "nvrs" indicates the state of an EEPROM access from one interrupt to another: its top bit ("nvravl") shows whether the EEPROM is already being used, bit 6 ("nvrwr") shows whether it is being written or read, and the low-order 4 bits hold a state number, which is used to transfer control to the appropriate code within the Timer T5 interrupt service routine.

On each Timer T5 interrupt (see labels "tmrint", "t5poll", "t5int"), the timer is stopped, a check is made to determine whether the EEPROM is being read or written (T5 interrupts are used for both), and then a multiway branch (jidw) is performed based on the state number in the variable "nvrs". The state number is incremented on each interrupt. On a Read transfer, five states are entered, at the following labels:

t5rd0 activates the chip select to the EEPROM and initiates the MICROWIRE transfer to send the first byte of a READ command. Timer T5 is started to time out the MICROWIRE transfer.

t5rd1 sends the second byte of the READ command. Timer T5 is started to time out the MICROWIRE transfer.

t5rd2 initiates the MICROWIRE transfer to read the first byte of data from the current EEPROM word. Timer T5 is started to time out the MICROWIRE transfer.

t5rd3 accepts the first byte of the data into the high-order byte of the variable "nvword", and initiates the transfer to read the second byte of the current EEPROM word. Timer T5 is started to time out the MICROWIRE transfer.

t5rd4 accepts the second byte from the EEPROM into the low-order byte of the variable "nvword", and then moves the word into the EEPROM string buffer, called "nvrbuf", using a pointer called "nvrptr". It then checks whether the requested number of bytes has been read (by decrementing the "nvrnum" variable). If so, it leaves Timer T5 stopped, disables its interrupt and returns. This would also be the proper place to set a semaphore flag to acknowledge to the main program that the reading is complete. (Code for this is not included here; it would vary from system to system.) If the requested number of bytes has not yet been read, it increments the address field of the READ command in "nvrcmd", resets the state field in "nvrs" to zero, leaves Timer T5 interrupts enabled, and jumps directly to the "t5rd0" routine to continue.

3.3 Writing

At label "wnvr", an EEPROM ERASE command is constructed from the word address supplied by the CPU. The 16-bit value to be written is placed in the variable "nvword". As in the READ-NVR command above, the "nvrs" variable is initialized to select the first state of an EEPROM write operation, and Timer T5 is used to provide the interrupts that schedule the steps. There are 13 states involved in writing a word to the EEPROM, at the following labels:

t5wr0 activates the chip select signal to the EEPROM, and sends the first byte of an EWEN command to enable ERASE and WRITE commands. Timer T5 is started to time out the MICROWIRE transfer.

t5wr1 sends the second byte of the EWEN command. Timer T5 is started to time out the MICROWIRE transfer.

t5wr2 removes the chip select signal briefly (to signal the beginning of a new command), then sends the first byte of an ERASE command. Timer T5 is started to time out the MICROWIRE transfer.

t5wr3 sends the second byte of the ERASE command, from the variable "nvrcmd". Timer T5 is started to time out the MICROWIRE transfer.

t5wr4 removes the chip select signal, then sets up the Timer T5 interval to 20 milliseconds, to time the duration of the EEPROM's internal Erase pulse.

t5wr5 (entered 20 milliseconds after "t5wr4") re-asserts the chip select signal to the EEPROM, and transfers the first byte of a WRITE command. Timer T5 is started to time out the MICROWIRE transfer.

t5wr6 alters the command in "nvrcmd" to a WRITE command, then transfers it as the second command byte to the EEPROM. Timer T5 is started to time out the MICROWIRE transfer.

t5wr7 transfers the first byte of data to be written. Timer T5 is started to time out the MICROWIRE transfer.

t5wr8 transfers the second byte of data to be written. Timer T5 is started to time out the MICROWIRE transfer.

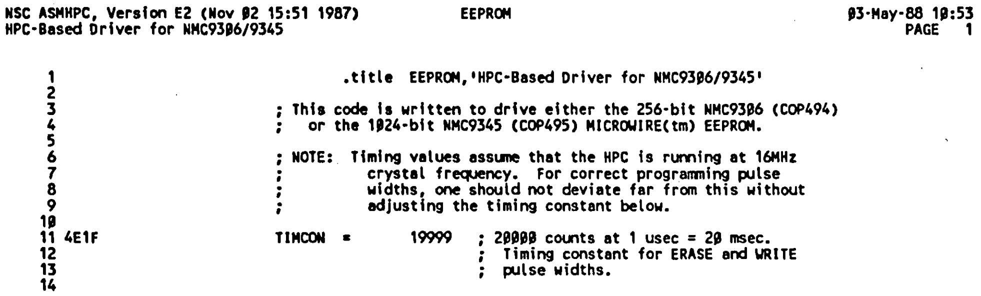
t5wr9 removes the chip select signal, then sets up the Timer T5 interval to 20 milliseconds, to time the duration of the EEPROM's internal Write pulse.

t5wr10 (entered 20 milliseconds after "t5wr9") re-asserts the chip select signal to the EEPROM, and transfers the first byte of an EWDS command (Erase/Write Disable). Timer T5 is started to time out the MICROWIRE transfer.

t5wr11 transfers the second byte of the EWDS command. Timer T5 is started to time out the MICROWIRE transfer.

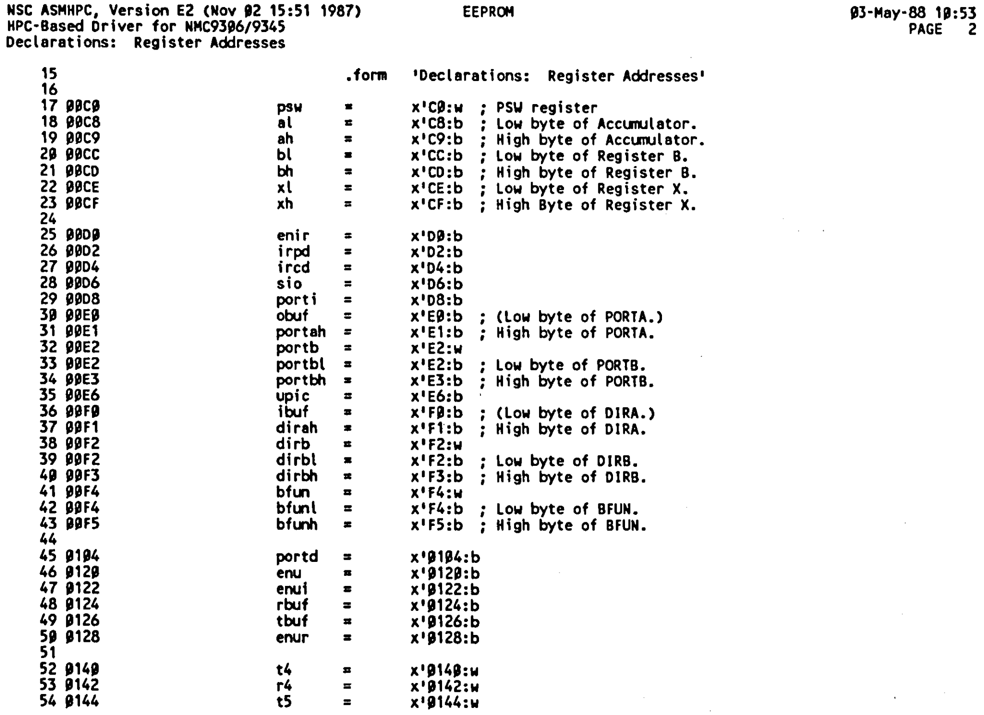
t5wr12 removes the chip select signal to the EEPROM, keeps Timer T5 stopped, disables its interrupt, and returns. This would also be the proper place to set a semaphore flag to acknowledge to the main program that the writing is complete. (Code for this is not included here; it would vary from system to system.)

3.4 Source Listing



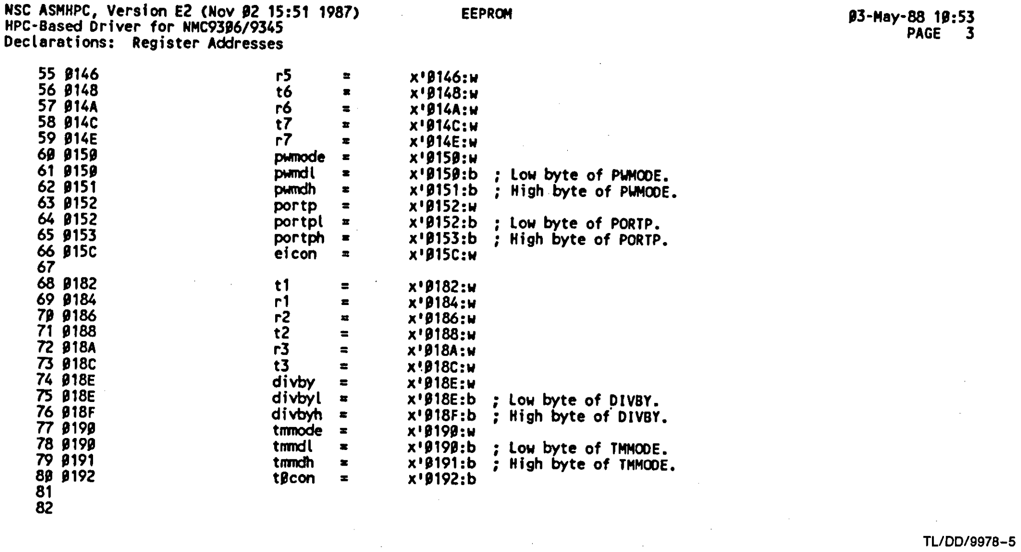
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                           PAGE   1

                                      .title  EEPROM,'HPC-Based Driver for NMC9306/9345'

                          ; This code is written to drive either the 256-bit NMC9306 (COP494)
                          ;   or the 1024-bit NMC9345 (COP495) MICROWIRE(tm) EEPROM.

                          ; NOTE:  Timing values assume that the HPC is running at 16MHz
                          ;          crystal frequency.  For correct programming pulse
                          ;          widths, one should not deviate far from this without
                          ;          adjusting the timing constant below.

     4E1F                 TIMCON  =       19999   ; 20000 counts at 1 usec = 20 msec.
                                                  ;  Timing constant for ERASE and WRITE
                                                  ;  pulse widths.
```

TL/DD/9978-3



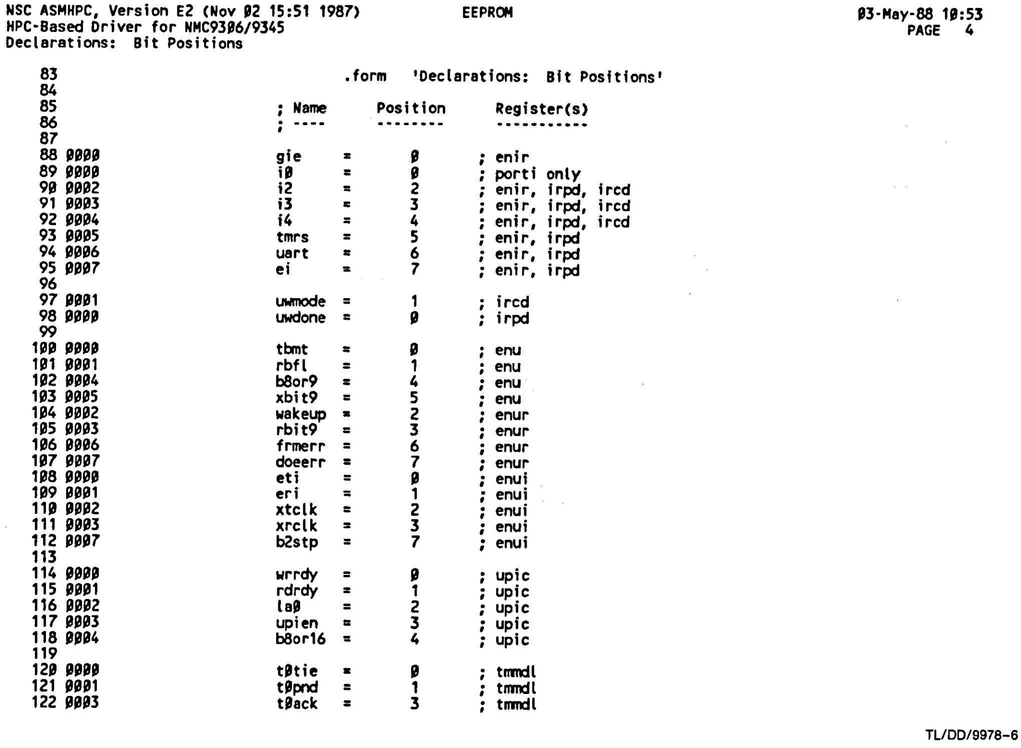
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                           PAGE   2
Declarations:  Register Addresses

                                      .form   'Declarations:  Register Addresses'

     00C0                 psw     =       x'C0:w  ; PSW register
     00C8                 al      =       x'C8:b  ; Low byte of Accumulator.
     00C9                 ah      =       x'C9:b  ; High byte of Accumulator.
     00CC                 bl      =       x'CC:b  ; Low byte of Register B.
     00CD                 bh      =       x'CD:b  ; High byte of Register B.
     00CE                 xl      =       x'CE:b  ; Low byte of Register X.
     00CF                 xh      =       x'CF:b  ; High Byte of Register X.

     00D0                 enir    =       x'D0:b
     00D2                 irpd    =       x'D2:b
     00D4                 ircd    =       x'D4:b
     00D6                 sio     =       x'D6:b
     00D8                 porti   =       x'D8:b
     00E0                 obuf    =       x'E0:b  ; (Low byte of PORTA.)
     00E1                 portah  =       x'E1:b  ; High byte of PORTA.
     00E2                 portb   =       x'E2:w
     00E2                 portbl  =       x'E2:b  ; Low byte of PORTB.
     00E3                 portbh  =       x'E3:b  ; High byte of PORTB.
     00E6                 upic    =       x'E6:b
     00F0                 ibuf    =       x'F0:b  ; (Low byte of DIRA.)
     00F1                 dirah   =       x'F1:b  ; High byte of DIRA.
     00F2                 dirb    =       x'F2:w
     00F2                 dirbl   =       x'F2:b  ; Low byte of DIRB.
     00F3                 dirbh   =       x'F3:b  ; High byte of DIRB.
     00F4                 bfun    =       x'F4:w
     00F4                 bfunl   =       x'F4:b  ; Low byte of BFUN.
     00F5                 bfunh   =       x'F5:b  ; High byte of BFUN.

     0104                 portd   =       x'0104:b
     0120                 enu     =       x'0120:b
     0122                 enui    =       x'0122:b
     0124                 rbuf    =       x'0124:b
     0126                 tbuf    =       x'0126:b
     0128                 enur    =       x'0128:b

     0140                 t4      =       x'0140:w
     0142                 r4      =       x'0142:w
     0144                 t5      =       x'0144:w
```

TL/DD/9978-4



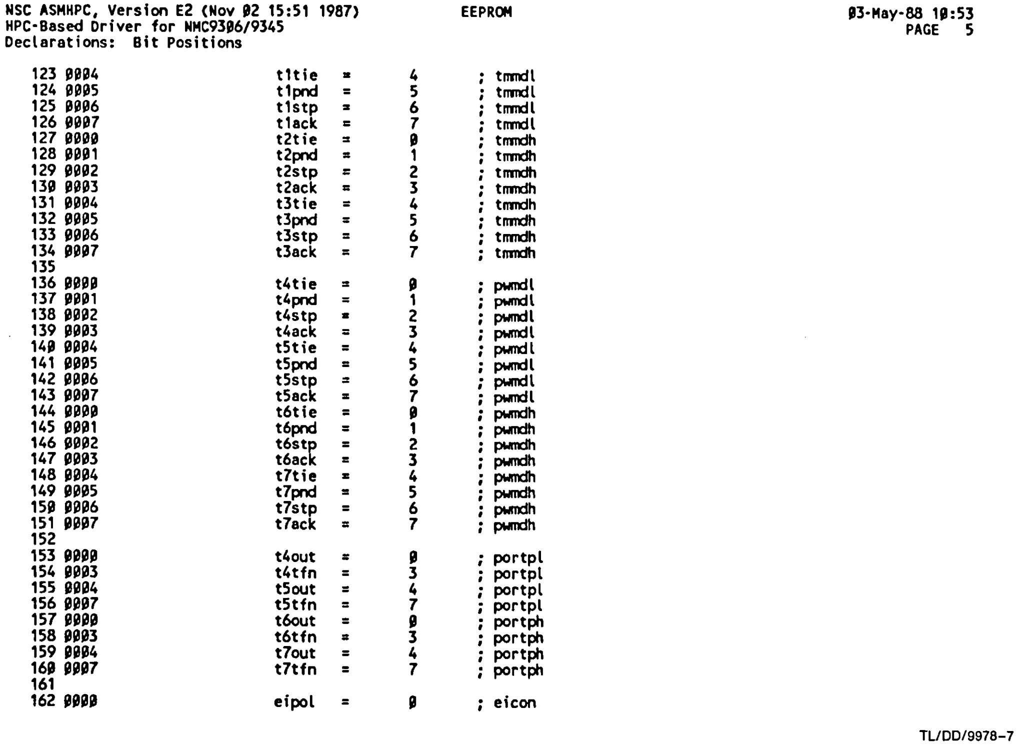
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                           PAGE   3
Declarations:  Register Addresses

   0146                      r5       =      x'0146:w
   0148                      t6       =      x'0148:w
   014A                      r6       =      x'014A:w
   014C                      t7       =      x'014C:w
   014E                      r7       =      x'014E:w
   0150                      pwmode   =      x'0150:w
   0150                      pwmdl    =      x'0150:b  ; Low byte of PWMODE.
   0151                      pwmdh    =      x'0151:b  ; High byte of PWMODE.
   0152                      portp    =      x'0152:w
   0152                      portpl   =      x'0152:b  ; Low byte of PORTP.
   0153                      portph   =      x'0153:b  ; High byte of PORTP.
   015C                      eicon    =      x'015C:w

   0182                      t1       =      x'0182:w
   0184                      r1       =      x'0184:w
   0186                      r2       =      x'0186:w
   0188                      t2       =      x'0188:w
   018A                      r3       =      x'018A:w
   018C                      t3       =      x'018C:w
   018E                      divby    =      x'018E:w
   018E                      divbyl   =      x'018E:b  ; Low byte of DIVBY.
   018F                      divbyh   =      x'018F:b  ; High byte of DIVBY.
   0190                      tmmode   =      x'0190:w
   0190                      tmmdl    =      x'0190:b  ; Low byte of TMMODE.
   0191                      tmmdh    =      x'0191:b  ; High byte of TMMODE.
   0192                      t0con    =      x'0192:b


```

TL/DD/9978-5



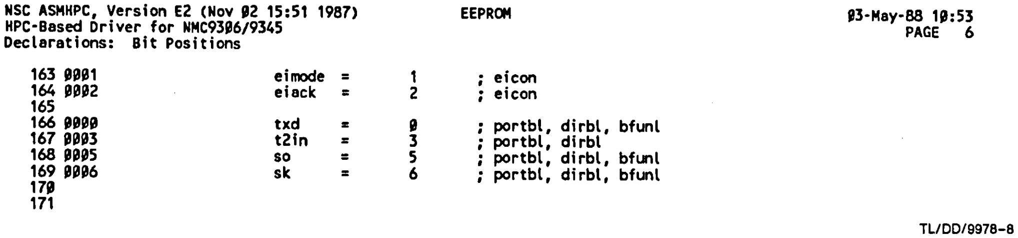
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                           PAGE   4
Declarations:  Bit Positions

                                  .form   'Declarations:  Bit Positions'

                             ; Name      Position      Register(s)
                             ; ----      --------      -----------

   0000                      gie      =      0        ; enir
   0000                      i0       =      0        ; porti only
   0002                      i2       =      2        ; enir, irpd, ircd
   0003                      i3       =      3        ; enir, irpd, ircd
   0004                      i4       =      4        ; enir, irpd, ircd
   0005                      tmrs     =      5        ; enir, irpd
   0006                      uart     =      6        ; enir, irpd
   0007                      ei       =      7        ; enir, irpd

   0001                      uwmode   =      1        ; ircd
   0000                      uwdone   =      0        ; irpd

   0000                      tbmt     =      0        ; enu
   0001                      rbfl     =      1        ; enu
   0004                      b8or9    =      4        ; enu
   0005                      xbit9    =      5        ; enu
   0002                      wakeup   =      2        ; enur
   0003                      rbit9    =      3        ; enur
   0006                      frmerr   =      6        ; enur
   0007                      doeerr   =      7        ; enur
   0000                      eti      =      0        ; enui
   0001                      eri      =      1        ; enui
   0002                      xtclk    =      2        ; enui
   0003                      xrclk    =      3        ; enui
   0007                      b2stp    =      7        ; enui

   0000                      wrrdy    =      0        ; upic
   0001                      rdrdy    =      1        ; upic
   0002                      la0      =      2        ; upic
   0003                      upien    =      3        ; upic
   0004                      b8or16   =      4        ; upic

   0000                      t0tie    =      0        ; tmmdl
   0001                      t0pnd    =      1        ; tmmdl
   0003                      t0ack    =      3        ; tmmdl
```

TL/DD/9978-6



```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                          03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                        PAGE    5
Declarations:  Bit Positions

   123 0004                  t1tie   =     4      ; tmmdl
   124 0005                  t1pnd   =     5      ; tmmdl
   125 0006                  t1stp   =     6      ; tmmdl
   126 0007                  t1ack   =     7      ; tmmdl
   127 0000                  t2tie   =     0      ; tmmdh
   128 0001                  t2pnd   =     1      ; tmmdh
   129 0002                  t2stp   =     2      ; tmmdh
   130 0003                  t2ack   =     3      ; tmmdh
   131 0004                  t3tie   =     4      ; tmmdh
   132 0005                  t3pnd   =     5      ; tmmdh
   133 0006                  t3stp   =     6      ; tmmdh
   134 0007                  t3ack   =     7      ; tmmdh
   135
   136 0000                  t4tie   =     0      ; pwmdl
   137 0001                  t4pnd   =     1      ; pwmdl
   138 0002                  t4stp   =     2      ; pwmdl
   139 0003                  t4ack   =     3      ; pwmdl
   140 0004                  t5tie   =     4      ; pwmdl
   141 0005                  t5pnd   =     5      ; pwmdl
   142 0006                  t5stp   =     6      ; pwmdl
   143 0007                  t5ack   =     7      ; pwmdl
   144 0000                  t6tie   =     0      ; pwmdh
   145 0001                  t6pnd   =     1      ; pwmdh
   146 0002                  t6stp   =     2      ; pwmdh
   147 0003                  t6ack   =     3      ; pwmdh
   148 0004                  t7tie   =     4      ; pwmdh
   149 0005                  t7pnd   =     5      ; pwmdh
   150 0006                  t7stp   =     6      ; pwmdh
   151 0007                  t7ack   =     7      ; pwmdh
   152
   153 0000                  t4out   =     0      ; portpl
   154 0003                  t4tfn   =     3      ; portpl
   155 0004                  t5out   =     4      ; portpl
   156 0007                  t5tfn   =     7      ; portpl
   157 0000                  t6out   =     0      ; portph
   158 0003                  t6tfn   =     3      ; portph
   159 0004                  t7out   =     4      ; portph
   160 0007                  t7tfn   =     7      ; portph
   161
   162 0000                  eipol   =     0      ; eicon
```

TL/DD/9978-7



```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                          03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                        PAGE    6
Declarations:  Bit Positions

   163 0001                  eimode  =     1      ; eicon
   164 0002                  eiack   =     2      ; eicon
   165
   166 0000                  txd     =     0      ; portbl, dirbl, bfunl
   167 0003                  t2in    =     3      ; portbl, dirbl
   168 0005                  so      =     5      ; portbl, dirbl, bfunl
   169 0006                  sk      =     6      ; portbl, dirbl, bfunl
   170
   171
```

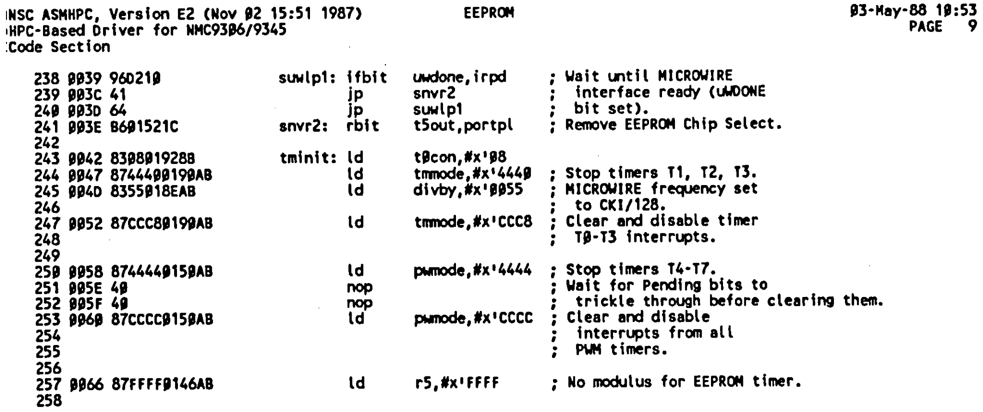
TL/DD/9978-8

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                         PAGE   7
Space Declarations

  172                             .form   'Space Declarations'
  173 0000                        .sect   DSECT,BASE,REL
  174
  175                        ;WORD-ALIGNED VARIABLES
  176
  177 0000                   stackb: .dsw    16      ; Space for 16 words.
  178 0020                   nvrbuf: .dsw    4       ; EEPROM String Buffer.
  179 0028                   nvrptr: .dsw 1  ; Pointer into EEPROM Data buffer.
  180 002A                   nvword: .dsw 1  ; Scratch location for gathering EEPROM data as words.
  181
  182                        ;BYTE-ALIGNED VARIABLES
  183
  184 002C                   nvrcmd: .dsb 1  ; Current EEPROM command.
  185 002D                   nvrnum: .dsb 1  ; Byte count for current EEPROM Read command.
  186 002E                   nvrs:   .dsb 1  ; EEPROM status byte:  phase number for sequencing MICROWIRE
  187                                        ;  transfers.
  188
  189                        ;BIT DEFINITIONS
  190
  191                                 ; NVRS byte:  Status of EEPROM MICROWIRE transfers.
  192                                 ;                 Contains phase (step number) of current EEPROM command
  193                                 ;                 in low-order 4 bits.  Top two bits are as follows:
  194 0007                   nvravl= 7       ; When set, indicates that no EEPROM command is in progress.
  195 0006                   nvrwr=  6       ; 0 means an EEPROM Read is in progress;  1 means EEPROM Write.
  196
  197
```

TL/DD/9978-9

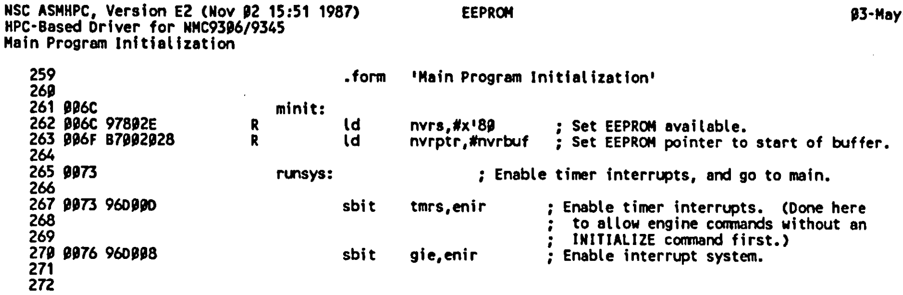
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                         PAGE   8
Code Section

  198                             .form   'Code Section'
  199 0000                        .sect   CSECT,ROM16,REL ; Code space.
  200
  201 0000 B70008C0          start:  ld      psw,#x'08       ; Set one WAIT state.
  202 0004 9700D0                    ld      enir,#x'00      ; Disable interrupts
  203                                                         ;  individually.
  204
  205 0007                   sram:                   ; Clear all RAM locations.
  206                                                ; Basepage bank:
  207 0007 8D00BE                    ld      BK,#x'0000,#x'00BE      ; Establish loop base and limit.
  208 000A 00                sraml1: clr     A
  209 000B E1                        xs      A,[B+].w
  210 000C 62                        jp      sraml1
  211
  212                                                ; Non-basepage bank:
  213 000D A701C001FE                ld      BK,#x'01C0,#x'01FE      ; Establish loop base and limit.
  214 0012 00                sraml2: clr     A
  215 0013 E1                        xs      A,[B+].w
  216 0014 62                        jp      sraml2
  217
  218 0015                   suwire:                         ; MICROWIRE setup.
  219                                                        ; (EEPROM is automatically
  220                                                        ;  deselected on reset, since
  221                                                        ;  Port P is cleared.)
  222
  223 0015 96F40D                    sbit    so,bfunl        ; Enable SO output.
  224 0018 96F20D                    sbit    so,dirbl
  225 001B 96E21E                    rbit    sk,portbl       ; Set up SK output.
  226 001E 96F20E                    sbit    sk,dirbl
  227 0021 96F40E                    sbit    sk,bfunl
  228 0024 96D409                    sbit    uwmode,ircd     ; Set Master Mode.
  229 0027 872225018EAB              ld      divby,#x'2225   ; Set MICROWIRE frequency.
  230
  231 002D 96D210            suwlp:  ifbit   uwdone,irpd     ; Wait until MICROWIRE
  232 0030 41                        jp      snvr1           ;  interface ready (uWDONE
  233 0031 64                        jp      suwlp           ;  bit set).
  234
  235 0032                   snvr1:                          ; Cancel any EEPROM Write in progress:
  236 0032 B601520C                  sbit    t5out,portpl    ; Set EEPROM Chip Select active.
  237 0036 9700D6                    ld      sio,#0          ; Send a byte of zeroes.
```

TL/DD/9978-10



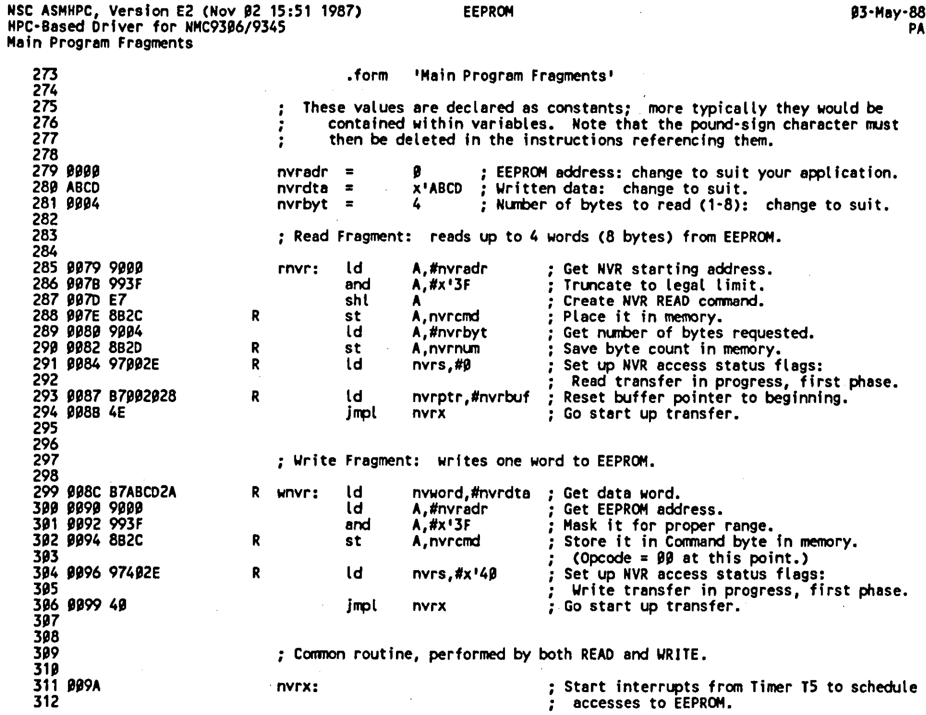
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)           EEPROM                                   03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                                   PAGE   9
Code Section

 0039 960210             suwlp1: ifbit   uwdone,irpd        ; Wait until MICROWIRE
 003C 41                         jp      snvr2              ;  interface ready (uWDONE
 003D 64                         jp      suwlp1             ;  bit set).
 003E B601521C           snvr2:  rbit    t5out,portpl       ; Remove EEPROM Chip Select.

 0042 8308019288         tminit: ld      t0con,#x'08
 0047 8744400190AB               ld      tmmode,#x'4440     ; Stop timers T1, T2, T3.
 004D 8355018EAB                 ld      divby,#x'0055      ; MICROWIRE frequency set
                                                            ;  to CKI/128.
 0052 87CCC80190AB               ld      tmmode,#x'CCC8     ; Clear and disable timer
                                                            ;  T0-T3 interrupts.

 0058 874444015OAB               ld      pwmode,#x'4444     ; Stop timers T4-T7.
 005E 40                         nop                        ; Wait for Pending bits to
 005F 40                         nop                        ;  trickle through before clearing them.
 0060 87CCCC0150AB               ld      pwmode,#x'CCCC     ; Clear and disable
                                                            ;  interrupts from all
                                                            ;  PWM timers.

 0066 87FFFF0146AB               ld      r5,#x'FFFF         ; No modulus for EEPROM timer.

```

TL/DD/9978-11



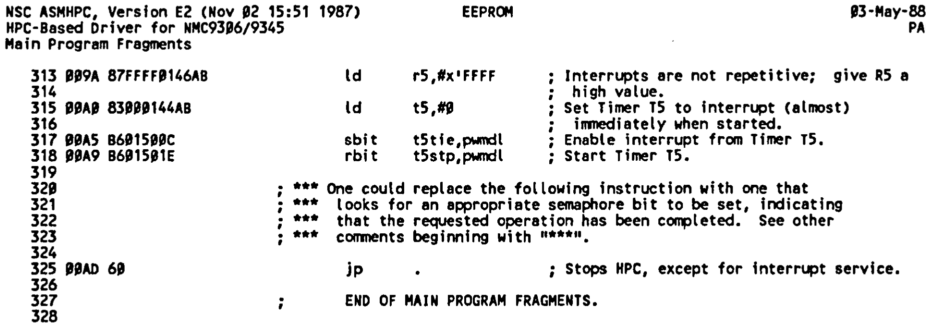
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)           EEPROM                                   03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                                   PAGE  10
Main Program Initialization

                                 .form   'Main Program Initialization'

 006C                    minit:
 006C 97802E         R           ld      nvrs,#x'80          ; Set EEPROM available.
 006F B7002028       R           ld      nvrptr,#nvrbuf      ; Set EEPROM pointer to start of buffer.

 0073                    runsys:                   ; Enable timer interrupts, and go to main.

 0073 960D00D                    sbit    tmrs,enir          ; Enable timer interrupts.  (Done here
                                                            ;  to allow engine commands without an
                                                            ;  INITIALIZE command first.)
 0076 960D008                    sbit    gie,enir           ; Enable interrupt system.


```

TL/DD/9978-12



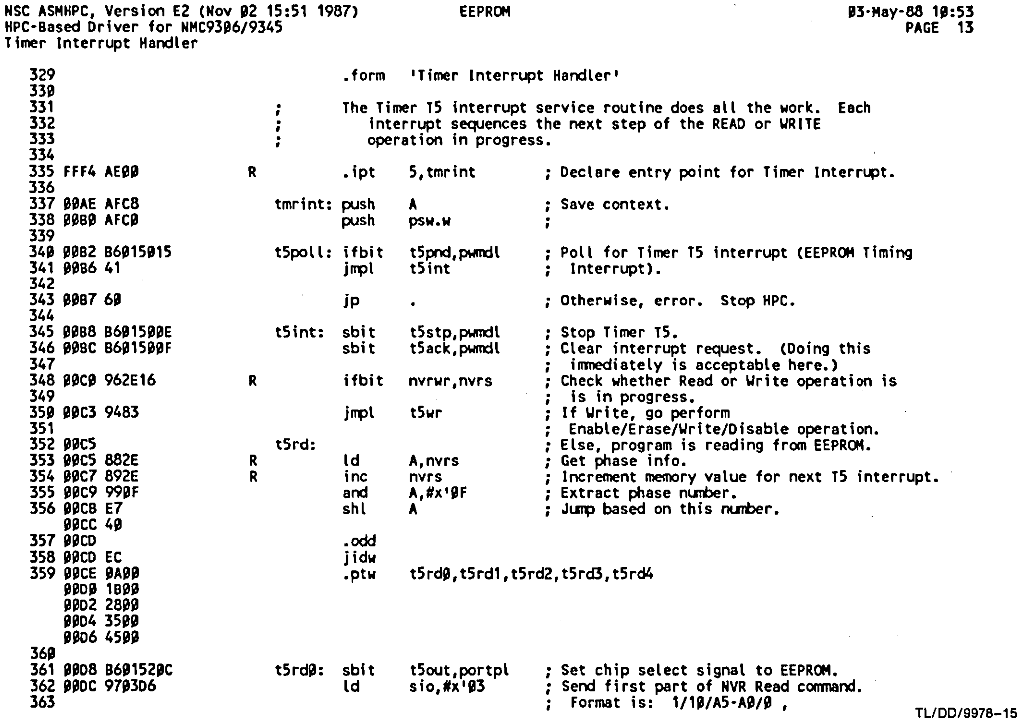
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                           03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                        PAGE  11
Main Program Fragments

                                            .form   'Main Program Fragments'

                                    ;  These values are declared as constants;  more typically they would be
                                    ;      contained within variables.  Note that the pound-sign character must
                                    ;      then be deleted in the instructions referencing them.

     0000                           nvradr  =       0        ; EEPROM address: change to suit your application.
     ABCD                           nvrdta  =       x'ABCD   ; Written data:  change to suit.
     0004                           nvrbyt  =       4        ; Number of bytes to read (1-8):  change to suit.

                                    ; Read Fragment:  reads up to 4 words (8 bytes) from EEPROM.

     0079 9000                      rnvr:   ld      A,#nvradr        ; Get NVR starting address.
     007B 993F                              and     A,#x'3F          ; Truncate to legal limit.
     007D E7                                shl     A                ; Create NVR READ command.
     007E 8B2C               R              st      A,nvrcmd         ; Place it in memory.
     0080 9004                              ld      A,#nvrbyt        ; Get number of bytes requested.
     0082 8B2D               R              st      A,nvrnum         ; Save byte count in memory.
     0084 97002E             R              ld      nvrs,#0          ; Set up NVR access status flags:
                                                                     ;  Read transfer in progress, first phase.
     0087 B7002028           R              ld      nvrptr,#nvrbuf   ; Reset buffer pointer to beginning.
     008B 4E                                jmpl    nvrx             ; Go start up transfer.


                                    ; Write Fragment:  writes one word to EEPROM.

     008C B7ABCD2A           R      wnvr:   ld      nvword,#nvrdta   ; Get data word.
     0090 9000                              ld      A,#nvradr        ; Get EEPROM address.
     0092 993F                              and     A,#x'3F          ; Mask it for proper range.
     0094 8B2C               R              st      A,nvrcmd         ; Store it in Command byte in memory.
                                                                     ;  (Opcode = 00 at this point.)
     0096 97402E             R              ld      nvrs,#x'40       ; Set up NVR access status flags:
                                                                     ;  Write transfer in progress, first phase.
     0099 40                                jmpl    nvrx             ; Go start up transfer.


                                    ; Common routine, performed by both READ and WRITE.

     009A                           nvrx:                            ; Start interrupts from Timer T5 to schedule
                                                                     ;  accesses to EEPROM.
```

TL/DD/9978-13



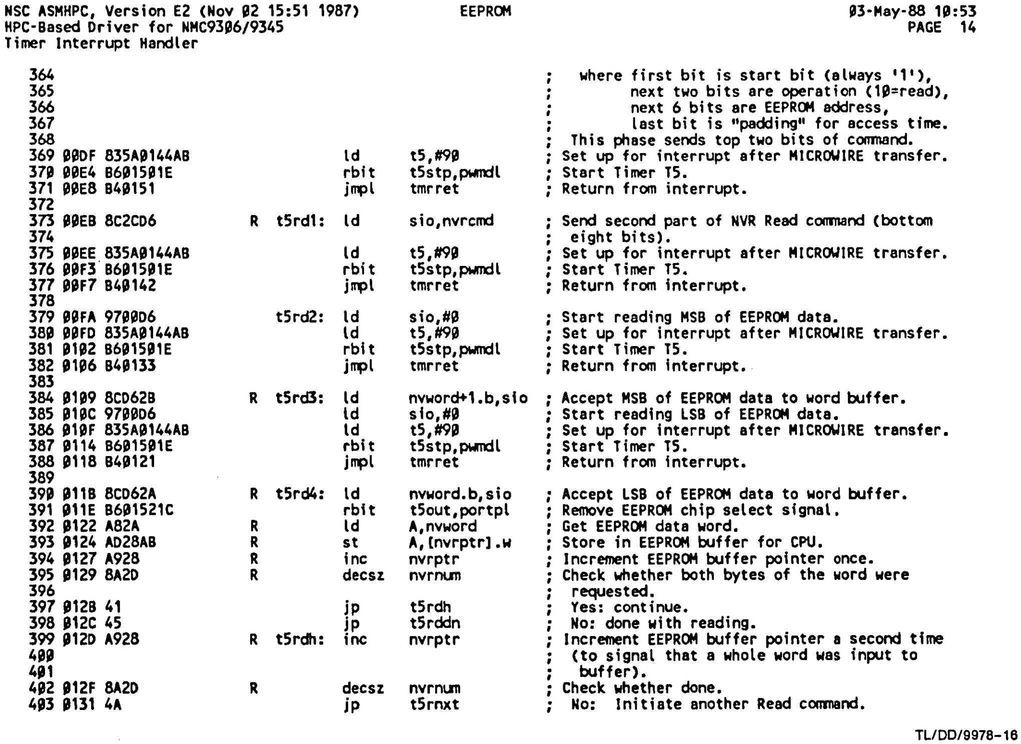
```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                           03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                        PAGE  12
Main Program Fragments

     009A 87FFFF0146AB                      ld      r5,#x'FFFF       ; Interrupts are not repetitive;  give R5 a
                                                                     ;  high value.
     00A0 83000144AB                        ld      t5,#0            ; Set Timer T5 to interrupt (almost)
                                                                     ;  immediately when started.
     00A5 B601500C                          sbit    t5tie,pwmdl      ; Enable interrupt from Timer T5.
     00A9 B601501E                          rbit    t5stp,pwmdl      ; Start Timer T5.

                                    ; *** One could replace the following instruction with one that
                                    ; ***  looks for an appropriate semaphore bit to be set, indicating
                                    ; ***  that the requested operation has been completed.  See other
                                    ; ***  comments beginning with "****".

     00AD 60                                jp      .                ; Stops HPC, except for interrupt service.

                                    ;       END OF MAIN PROGRAM FRAGMENTS.

```

TL/DD/9978-14



```
NSC ASMHPC, Version E2 (Nov Ø2 15:51 1987)          EEPROM                          Ø3-May-88 1Ø:53
HPC-Based Driver for NMC93Ø6/9345                                                          PAGE  13
Timer Interrupt Handler

                                    .form   'Timer Interrupt Handler'


                          ;         The Timer T5 interrupt service routine does all the work.  Each
                          ;            interrupt sequences the next step of the READ or WRITE
                          ;            operation in progress.

FFF4 AEØØ            R              .ipt    5,tmrint      ; Declare entry point for Timer Interrupt.

ØØAE AFC8                 tmrint:   push    A             ; Save context.
ØØBØ AFCØ                           push    psw.w         ;

ØØB2 B6Ø15Ø15             t5poll:   ifbit   t5pnd,pwmdl   ; Poll for Timer T5 interrupt (EEPROM Timing
ØØB6 41                             jmpl    t5int         ;  Interrupt).

ØØB7 6Ø                             jp      .             ; Otherwise, error.  Stop HPC.

ØØB8 B6Ø15ØØE             t5int:    sbit    t5stp,pwmdl   ; Stop Timer T5.
ØØBC B6Ø15ØØF                       sbit    t5ack,pwmdl   ; Clear interrupt request.  (Doing this
                                                          ;  immediately is acceptable here.)
ØØCØ 962E16          R              ifbit   nvrwr,nvrs    ; Check whether Read or Write operation is
                                                          ;  is in progress.
ØØC3 9483                           jmpl    t5wr          ; If Write, go perform
                                                          ;  Enable/Erase/Write/Disable operation.
ØØC5                      t5rd:                           ; Else, program is reading from EEPROM.
ØØC5 882E            R              ld      A,nvrs        ; Get phase info.
ØØC7 892E            R              inc     nvrs          ; Increment memory value for next T5 interrupt.
ØØC9 99ØF                           and     A,#x'ØF       ; Extract phase number.
ØØCB E7                             shl     A             ; Jump based on this number.
ØØCC 4Ø
ØØCD                                .odd
ØØCD EC                             jidw
ØØCE ØAØØ                           .ptw    t5rdØ,t5rd1,t5rd2,t5rd3,t5rd4
ØØDØ 1BØØ
ØØD2 28ØØ
ØØD4 35ØØ
ØØD6 45ØØ

ØØD8 B6Ø152ØC             t5rdØ:    sbit    t5out,portpl  ; Set chip select signal to EEPROM.
ØØDC 97Ø3D6                         ld      sio,#x'Ø3     ; Send first part of NVR Read command.
                                                          ;  Format is:  1/1Ø/A5-AØ/Ø ,
```

TL/DD/9978-15



```
NSC ASMHPC, Version E2 (Nov Ø2 15:51 1987)          EEPROM                          Ø3-May-88 1Ø:53
HPC-Based Driver for NMC93Ø6/9345                                                          PAGE  14
Timer Interrupt Handler

                                                          ;   where first bit is start bit (always '1'),
                                                          ;         next two bits are operation (1Ø=read),
                                                          ;         next 6 bits are EEPROM address,
                                                          ;         last bit is "padding" for access time.
                                                          ;  This phase sends top two bits of command.
ØØDF 835AØ144AB                     ld      t5,#9Ø        ; Set up for interrupt after MICROWIRE transfer.
ØØE4 B6Ø15Ø1E                       rbit    t5stp,pwmdl   ; Start Timer T5.
ØØE8 B4Ø151                         jmpl    tmrret        ; Return from interrupt.

ØØEB 8C2CD6          R    t5rd1:    ld      sio,nvrcmd    ; Send second part of NVR Read command (bottom
                                                          ;  eight bits).
ØØEE 835AØ144AB                     ld      t5,#9Ø        ; Set up for interrupt after MICROWIRE transfer.
ØØF3 B6Ø15Ø1E                       rbit    t5stp,pwmdl   ; Start Timer T5.
ØØF7 B4Ø142                         jmpl    tmrret        ; Return from interrupt.

ØØFA 97ØØD6               t5rd2:    ld      sio,#Ø        ; Start reading MSB of EEPROM data.
ØØFD 835AØ144AB                     ld      t5,#9Ø        ; Set up for interrupt after MICROWIRE transfer.
Ø1Ø2 B6Ø15Ø1E                       rbit    t5stp,pwmdl   ; Start Timer T5.
Ø1Ø6 B4Ø133                         jmpl    tmrret        ; Return from interrupt.

Ø1Ø9 8CD62B          R    t5rd3:    ld      nvword+1.b,sio ; Accept MSB of EEPROM data to word buffer.
Ø1ØC 97ØØD6                         ld      sio,#Ø        ; Start reading LSB of EEPROM data.
Ø1ØF 835AØ144AB                     ld      t5,#9Ø        ; Set up for interrupt after MICROWIRE transfer.
Ø114 B6Ø15Ø1E                       rbit    t5stp,pwmdl   ; Start Timer T5.
Ø118 B4Ø121                         jmpl    tmrret        ; Return from interrupt.

Ø11B 8CD62A          R    t5rd4:    ld      nvword.b,sio  ; Accept LSB of EEPROM data to word buffer.
Ø11E B6Ø1521C                       rbit    t5out,portpl  ; Remove EEPROM chip select signal.
Ø122 A82A            R              ld      A,nvword      ; Get EEPROM data word.
Ø124 AD28AB          R              st      A,[nvrptr].w  ; Store in EEPROM buffer for CPU.
Ø127 A928            R              inc     nvrptr        ; Increment EEPROM buffer pointer once.
Ø129 8A2D            R              decsz   nvrnum        ; Check whether both bytes of the word were
                                                          ;  requested.
Ø12B 41                             jp      t5rdh         ;  Yes: continue.
Ø12C 45                             jp      t5rddn        ;  No: done with reading.
Ø12D A928            R    t5rdh:    inc     nvrptr        ; Increment EEPROM buffer pointer a second time
                                                          ;  (to signal that a whole word was input to
                                                          ;  buffer).
Ø12F 8A2D            R              decsz   nvrnum        ; Check whether done.
Ø131 4A                             jp      t5rnxt        ;  No:  Initiate another Read command.
```

TL/DD/9978-16

```
 404                                                      ;  Yes:  Terminate and pass data to CPU.
 405 0132 B601501C          t5rddn: rbit    t5tie,pwmdl      ; Disable Timer T5 interrupts.
 406 0136 962E0F         R          sbit    nvravl,nvrs      ; Set NVR available for more commands.
 407                            ;*** Here you'll want to set a semaphore bit saying that the READ
 408                            ;***  transfer is done.
 409 0139 B40100                        jmpl    tmrret           ; Return from interrupt.
 410
 411 013C                       t5rnxt:                          ; Here, more data needs to be read from the
 412                                                              ;  EEPROM.  Initiate another read cycle.
 413 013C 97002E         R          ld      nvrs,#x'00       ; Set up new transfer phase = 0.
 414 013F 892C           R          inc     nvrcmd           ; Increment address field of NVR command.
 415 0141 892C           R          inc     nvrcmd           ;  (Two increments are needed:  field starts
 416                                                              ;    in Bit 1.)
 417 0143 962C1F         R          rbit    7,nvrcmd         ; Prevent increments from altering operation
 418                                                              ;  field.  This allows addresses to roll over.
 419 0146 9581                          jmpl    t5rd             ; Rather than triggering a Timer T5 interrupt,
 420                                                              ;  just jump to T5 Read interrupt service again.
 421
 422
 423 0148                       t5wr:                   ; EEPROM Write sequence starts here.
 424 0148 882E           R          ld      A,nvrs           ; Get phase info.
 425 014A 892E           R          inc     nvrs             ; Increment memory value for next T5 interrupt.
 426 014C 990F                          and     A,#x'0F          ; Extract phase number.
 427 014E E7                            shl     A                ; Jump based on this number.
 428 014F                               .odd
 429 014F EC                            jidw
 430 0150 1A00                          .ptw    t5wr0,t5wr1,t5wr2,t5wr3
     0152 2A00
     0154 3600
     0156 4B00
 431 0158 5B00                          .ptw    t5wr4,t5wr5,t5wr6,t5wr7
     015A 6900
     015C 7900
     015E 8800
 432 0160 9400                          .ptw    t5wr8,t5wr9,t5wr10,t5wr11
     0162 A000
     0164 AE00
     0166 BD00
 433 0168 C800                          .ptw    t5wr12
 434
```

TL/DD/9978-17

```
016A B601520C          t5wr0:  sbit    t5out,portpl     ; Set chip select signal to EEPROM.
016E 9701D6                    ld      sio,#x'01        ; Send start bit of EWEN command.
0171 835A0144AB                ld      t5,#90           ; Set up for interrupt at end of MICROWIRE
                                                        ;  transfer.
0176 B601501E                  rbit    t5stp,pwmdl      ; Start timer T5.
017A 94C0                      jmpl    tmrret           ; Return from interrupt.

017C 97300D6           t5wr1:  ld      sio,#x'30        ; Send body of EWEN command.
017F 835A0144AB                ld      t5,#90           ; Set up for interrupt at end of MICROWIRE
                                                        ;  transfer.
0184 B601501E                  rbit    t5stp,pwmdl      ; Start timer T5.
0188 94B2                      jmpl    tmrret           ; Return from interrupt.

018A B601521C          t5wr2:  rbit    t5out,portpl     ; Remove EEPROM select momentarily to signal
018E 40                        nop                      ;  end of EWEN command, then:
018F B601520C                  sbit    t5out,portpl
0193 9701D6                    ld      sio,#x'01        ; Send Start Bit for ERASE command.
0196 835A0144AB                ld      t5,#90           ; Set up for interrupt at end of MICROWIRE
                                                        ;  transfer.
019B B601501E                  rbit    t5stp,pwmdl      ; Start timer T5.
019F 949B                      jmpl    tmrret           ; Return from interrupt.

01A1 82C02CDA        R t5wr3:  or      nvrcmd,#x'C0     ; Change NVR Command byte to ERASE command.
01A5 8C2CD6          R         ld      sio,nvrcmd       ; Send to EEPROM.
01A8 835A0144AB                ld      t5,#90           ; Set up for interrupt at end of MICROWIRE
                                                        ;  transfer.
01AD B601501E                  rbit    t5stp,pwmdl      ; Start timer T5.
01B1 9489                      jmpl    tmrret           ; Return from interrupt.

01B3 B601521C          t5wr4:  rbit    t5out,portpl     ; Remove EEPROM chip select signal, starting
                                                        ;  ERASE pulse inside EEPROM.
01B7 874E1F0144AB              ld      t5,#TIMCON       ; Set up for delay of 20
                                                        ;  milliseconds (erase pulse width).
01BD B601501E                  rbit    t5stp,pwmdl      ; Start timer T5.
01C1 9479                      jmpl    tmrret           ; Return from interrupt.

01C3 B601520C          t5wr5:  sbit    t5out,portpl     ; Set EEPROM chip select signal again, ending
                                                        ;  the ERASE pulse inside EEPROM.
01C7 9701D6                    ld      sio,#x'01        ; Send Start bit for Write command.
01CA 835A0144AB                ld      t5,#90           ; Set up for interrupt at end of MICROWIRE
```

TL/DD/9978-18

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                             PAGE  17
Timer Interrupt Handler

                                                              ;  transfer.
 01CF B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 01D3 9467                      jmpl     tmrret               ; Return from interrupt.

 01D5 962C1F          R  t5wr6: rbit     7,nvrcmd             ; Create WRITE command in NVR Command byte.
 01D8 8C2CD6          R         ld       sio,nvrcmd           ; Send to EEPROM.
 01DB 835A0144AB                ld       t5,#90               ; Set up for interrupt at end of MICROWIRE
                                                              ;  transfer.
 01E0 B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 01E4 9456                      jmpl     tmrret               ; Return from interrupt.

 01E6 8C2BD6          R  t5wr7: ld       sio,nvword+1.b       ; Send MSB of data to EEPROM.
 01E9 835A0144AB                ld       t5,#90               ; Set up for interrupt at end of MICROWIRE
                                                              ;  transfer.
 01EE B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 01F2 9448                      jmpl     tmrret               ; Return from interrupt.

 01F4 8C2AD6          R  t5wr8: ld       sio,nvword.b         ; Send LSB of data to EEPROM.
 01F7 835A0144AB                ld       t5,#90               ; Set up for interrupt at end of MICROWIRE
                                                              ;  transfer.
 01FC B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 0200 943A                      jmpl     tmrret               ; Return from interrupt.

 0202 B601521C           t5wr9: rbit     t5out,portpl         ; Remove EEPROM chip select, starting Write
                                                              ;  pulse within EEPROM.
 0206 874E1F0144AB              ld       t5,#TIMCON           ; Set up for delay of 20
                                                              ;  milliseconds (write pulse width).
 020C B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 0210 942A                      jmpl     tmrret               ; Return from interrupt.

 0212 B601520C          t5wr10: sbit     t5out,portpl         ; Set EEPROM chip select signal, ending Write
                                                              ;  pulse within EEPROM.
 0216 9701D6                    ld       sio,#x'01            ; Send Start bit for EWDS command (Disable
                                                              ;  Write/Erase).
 0219 835A0144AB                ld       t5,#90               ; Set up for interrupt at end of MICROWIRE
                                                              ;  transfer.
 021E B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 0222 59                        jmpl     tmrret               ; Return from interrupt.

 0223 9700D6            t5wr11: ld       sio,#x'00            ; Send body of EWDS command.
```

TL/DD/9978–19

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                              03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                             PAGE  18
Timer Interrupt Handler

 0226 835A0144AB                ld       t5,#90               ; Set up for interrupt at end of MICROWIRE
                                                              ;  transfer.
 022B B601501E                  rbit     t5stp,pwmdl          ; Start timer T5.
 022F 4C                        jmpl     tmrret               ; Return from interrupt.

 0230 B601521C          t5wr12: rbit     t5out,portpl         ; Remove EEPROM chip select signal.
 0234 B601501C                  rbit     t5tie,pwmdl          ; Disable Timer T5 interrupts.
 0238 962E0F          R         sbit     nvravl,nvrs          ; Set EEPROM Available.
                        ;*** Here you'll want to set a semaphore bit saying that the WRITE
                        ;***  transfer is done.
 023B 40                        jmpl     tmrret

 023C 3FC0              tmrret: pop      psw.w                ; Restore context.
 023E 3FC8                      pop      A
 0240 3E                        reti

 0241                           .end     start
```

TL/DD/9978–20

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                          03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                       PAGE  19
Timer Interrupt Handler

 ah      00C9 Abs Byte
 al      00C8 Abs Byte
 b2stp   0007 Abs Null
 b8or16  0004 Abs Null
 b8or9   0004 Abs Null
 bfun    00F4 Abs Word
 bfunh   00F5 Abs Byte
 bfunl   00F4 Abs Byte
 bh      00CD Abs Byte
 bl      00CC Abs Byte
 dirah   00F1 Abs Byte
 dirb    00F2 Abs Word
 dirbh   00F3 Abs Byte
 dirbl   00F2 Abs Byte
 divby   018E Abs Word
 divbyh  018F Abs Byte
 divbyl  018E Abs Byte
 doeerr  0007 Abs Null
 ei      0007 Abs Null
 eiack   0002 Abs Null
 eicon   015C Abs Word
 eimode  0001 Abs Null
 eipol   0000 Abs Null
 enir    00D0 Abs Byte
 enu     0120 Abs Byte
 enui    0122 Abs Byte
 enur    0128 Abs Byte
 eri     0001 Abs Null
 eti     0000 Abs Null
 frmerr  0006 Abs Null
 gie     0000 Abs Null
 i0      0000 Abs Null
 i2      0002 Abs Null
 i3      0003 Abs Null
 i4      0004 Abs Null
 ibuf    00F0 Abs Byte
 ircd    00D4 Abs Byte
 irpd    00D2 Abs Byte
 la0     0002 Abs Null
 minit   006C Rel Null ROM16
```

TL/DD/9978-21

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                          03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                       PAGE  20
Timer Interrupt Handler

 nvradr  0000 Abs Null
 nvravl  0007 Abs Null
 nvrbuf  0020 Rel Word BASE
 nvrbyt  0004 Abs Null
 nvrcmd  002C Rel Byte BASE
 nvrdta  ABCD Abs Null
 nvrnum  002D Rel Byte BASE
 nvrptr  0028 Rel Word BASE
 nvrs    002E Rel Byte BASE
 nvrwr   0006 Abs Null
 nvrx    009A Rel Null ROM16
 nvword  002A Rel Word BASE
 obuf    00E0 Abs Byte
 portah  00E1 Abs Byte
 portb   00E2 Abs Word
 portbh  00E3 Abs Byte
 portbl  00E2 Abs Byte
 portd   0104 Abs Byte
 porti   00D8 Abs Byte
 portp   0152 Abs Word
 portph  0153 Abs Byte
 portpl  0152 Abs Byte
 psw     00C0 Abs Word
 pwmdh   0151 Abs Byte
 pwmdl   0150 Abs Byte
 pwmode  0150 Abs Word
 r1      0184 Abs Word
 r2      0186 Abs Word
 r3      018A Abs Word
 r4      0142 Abs Word
 r5      0146 Abs Word
 r6      014A Abs Word
 r7      014E Abs Word
 rbfl    0001 Abs Null
 rbit9   0003 Abs Null
 rbuf    0124 Abs Byte
 rdrdy   0001 Abs Null
 rnvr    0079 Rel Null ROM16
 runsys  0073 Rel Null ROM16
 sio     00D6 Abs Byte
```

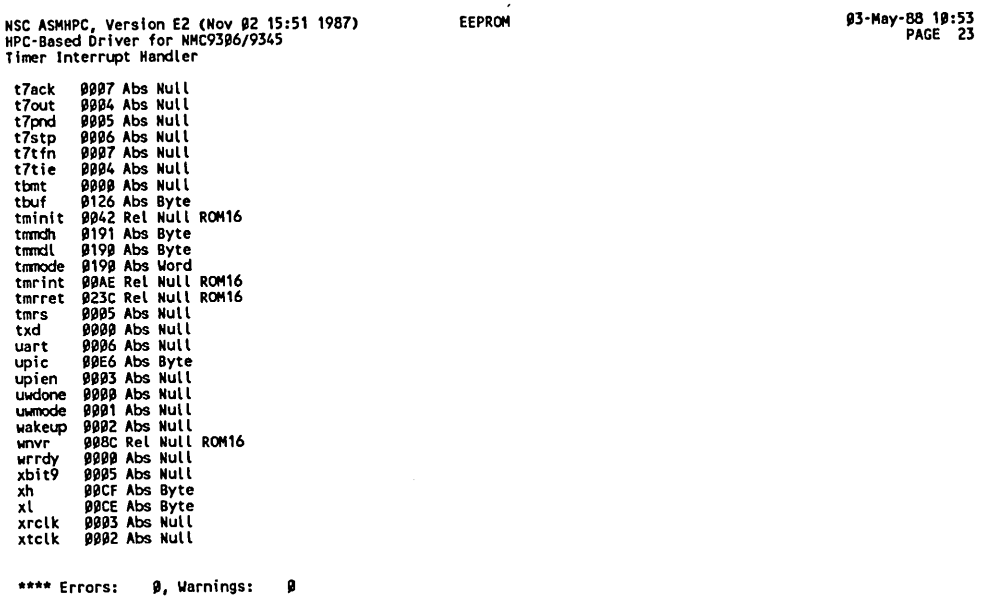
TL/DD/9978-22

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                        03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                        PAGE  21
Timer Interrupt Handler

 sk       0006 Abs Null
 snvr1    0032 Rel Null ROM16
 snvr2    003E Rel Null ROM16
 so       0005 Abs Null
 sram     0007 Rel Null ROM16
 sraml1   000A Rel Null ROM16
 sraml2   0012 Rel Null ROM16
 stackb   0000 Rel Word BASE
 start    0000 Rel Null ROM16
 suwire   0015 Rel Null ROM16
 suwlp    002D Rel Null ROM16
 suwlp1   0039 Rel Null ROM16
 TIMCON   4E1F Abs Null
 t0ack    0003 Abs Null
 t0con    0192 Abs Byte
 t0pnd    0001 Abs Null
 t0tie    0000 Abs Null
 t1       0182 Abs Word
 t1ack    0007 Abs Null
 t1pnd    0005 Abs Null
 t1stp    0006 Abs Null
 t1tie    0004 Abs Null
 t2       0188 Abs Word
 t2ack    0003 Abs Null
 t2in     0003 Abs Null
 t2pnd    0001 Abs Null
 t2stp    0002 Abs Null
 t2tie    0000 Abs Null
 t3       018C Abs Word
 t3ack    0007 Abs Null
 t3pnd    0005 Abs Null
 t3stp    0006 Abs Null
 t3tie    0004 Abs Null
 t4       0140 Abs Word
 t4ack    0003 Abs Null
 t4out    0000 Abs Null
 t4pnd    0001 Abs Null
 t4stp    0002 Abs Null
 t4tfn    0003 Abs Null
 t4tie    0000 Abs Null
```

TL/DD/9978-23

```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                        03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                        PAGE  22
Timer Interrupt Handler

 t5       0144 Abs Word
 t5ack    0007 Abs Null
 t5int    00B8 Rel Null ROM16
 t5out    0004 Abs Null
 t5pnd    0005 Abs Null
 t5poll   00B2 Rel Null ROM16
 t5rd     00C5 Rel Null ROM16
 t5rd0    00D8 Rel Null ROM16
 t5rd1    00EB Rel Null ROM16
 t5rd2    00FA Rel Null ROM16
 t5rd3    0109 Rel Null ROM16
 t5rd4    011B Rel Null ROM16
 t5rddn   0132 Rel Null ROM16
 t5rdh    012D Rel Null ROM16
 t5rnxt   013C Rel Null ROM16
 t5stp    0006 Abs Null
 t5tfn    0007 Abs Null
 t5tie    0004 Abs Null
 t5wr     0148 Rel Null ROM16
 t5wr0    016A Rel Null ROM16
 t5wr1    017C Rel Null ROM16
 t5wr10   0212 Rel Null ROM16
 t5wr11   0223 Rel Null ROM16
 t5wr12   0230 Rel Null ROM16
 t5wr2    018A Rel Null ROM16
 t5wr3    01A1 Rel Null ROM16
 t5wr4    01B3 Rel Null ROM16
 t5wr5    01C3 Rel Null ROM16
 t5wr6    01D5 Rel Null ROM16
 t5wr7    01E6 Rel Null ROM16
 t5wr8    01F4 Rel Null ROM16
 t5wr9    0202 Rel Null ROM16
 t6       0148 Abs Word
 t6ack    0003 Abs Null
 t6out    0000 Abs Null
 t6pnd    0001 Abs Null
 t6stp    0002 Abs Null
 t6tfn    0003 Abs Null
 t6tie    0000 Abs Null
 t7       014C Abs Word
```

TL/DD/9978-24



```
NSC ASMHPC, Version E2 (Nov 02 15:51 1987)          EEPROM                         03-May-88 10:53
HPC-Based Driver for NMC9306/9345                                                          PAGE  23
Timer Interrupt Handler

 t7ack    0007 Abs Null
 t7out    0004 Abs Null
 t7pnd    0005 Abs Null
 t7stp    0006 Abs Null
 t7tfn    0007 Abs Null
 t7tie    0004 Abs Null
 tbmt     0000 Abs Null
 tbuf     0126 Abs Byte
 tminit   0042 Rel Null ROM16
 tmmdh    0191 Abs Byte
 tmmdl    0190 Abs Byte
 tmmode   0190 Abs Word
 tmrint   00AE Rel Null ROM16
 tmrret   023C Rel Null ROM16
 tmrs     0005 Abs Null
 txd      0000 Abs Null
 uart     0006 Abs Null
 upic     00E6 Abs Byte
 upien    0003 Abs Null
 uwdone   0000 Abs Null
 uwmode   0001 Abs Null
 wakeup   0002 Abs Null
 wnvr     008C Rel Null ROM16
 wrrdy    0000 Abs Null
 xbit9    0005 Abs Null
 xh       00CF Abs Byte
 xl       00CE Abs Byte
 xrclk    0003 Abs Null
 xtclk    0002 Abs Null

 **** Errors:    0, Warnings:    0
```

TL/DD/9978-25

Extended Memory Support for HPC

National Semiconductor
Application Note 577
Raja Gopalan

INTRODUCTION

HPC™ family of microcontrollers have maximum addressing capability of 64 kbytes directly by the CPU. If an application requires more than 64k of address space, then the HPC address space can be expanded in terms of banks of memory, using an I/O port to select the memory banks. For example one can use PORTB pins 8, 9, 13 and 14 to select up to 16 banks of memory (which the MOLE development system also supports currently for debugging purposes). Please refer to the application note AN-497 "Expanding the HPC Address Space" by Joe Cocovich for hardware details.

The current version of HPC software package (Compiler, Assembler and Linker) however, does not directly support more than 64k of address space. This is mainly due to the Linker, which currently can handle only 64k of address space.

This report describes a method to handle more than 64k of address space from a software point of view. In order to do this, the user has to do multiple linking of modules in different banks and resolve the inter-bank symbol references.

The rest of the report describes the following:

1. Compiler generated selections (of code and data).
2. Programming conventions for bank switching.
3. Switch function to support bank switching.
4. Linking for bank switching.

SECTIONS GENERATED BY THE COMPILER

The compiler generates sections of relocatable assembly code which can be positioned in absolute address using the Linker in two ways:

1. Using the /SECT directive.
2. Using the /RANGE directive.

The following are the sections generated by the compiler for a source file named "MODULE":

1.	MODULE_code,rom8	Code.
2.	MODULE_code,rom16	
3.	MODULE_ram8_bss,ram8	Data area for uninitialized static variables.
4.	MODULE_ram16_bss,ram16	
5.	MODULE_ram8_data,ram8	Data area for initialized static variables.
6.	MODULE_ram16_data,ram16	
7.	MODULE_ram8_strdata,ram 8	Data area for string literals.
8.	MODULE_ram16_init,rom8	Initial value for static variables.
9.	MODULE_ram8_init,rom8	
10.	MODULE_ram8,strinit,rom8	Initial values for string literals.
11.	MODULE_base16_bss,base	Base page area for uninitialized static variables.
12.	MODULE_base8_bss,base	
13.	MODULE_base16_data,base	Base page area for initialized static variables.
14.	MODULE_base8_data,base	
15.	MODULE_base16_init,rom16	Initial values for Base page initialized variables.
16.	MODULE_base8_init,rom8	
17.	MODULE_rom16_data,rom16	Area for constant storage type.
18.	MODULE_rom8_data,rom8	
19.	c_stack,ram16	Stack area in module containing main( ).
20.	_init_info_	for each module which has any static variables defined.

PROGRAMMING CONVENTIONS TO BE USED FOR BANK SWITCHING

As far as the bank switching hardware is concerned, the HPC addressing space is divided into banks of memory. The Fixed Address space is referred to as shared bank and the switchable address space is called as switchable bank. Any mechanism for bank selection can be used, as long as the conventions mentioned below are strictly followed:

1. All static variables must be placed in the shared memory. Basepage must go in basepage (which is shared).
2. If string literals are not in ROM, they must be placed in the shared memory.
3. Initialization values for static variables or string literals in RAM must be in the shared memory. This includes basepage initializers and __init__info__ sections.
4. If string literals for a bank are in ROM, and are never used as an argument to an inter-bank function call, the literals for that bank can be in the switchable bank.
5. If constants for a bank are never used by passing their address as an argument to an inter-bank function call, the constants for that bank can be in the switchable bank.
6. If the addresses of constants or string literals for a bank are used as arguments to an inter-bank function call, the constants or literals must be in the shared memory.
7. The stack must be in the shared memory.
8. Interrupt vectors must point to routines in the shared memory.
9. Only code and qualified ROM data can be placed in switchable banks.
10. A call to a function placed in the shared memory is always direct.
11. A function call from one switchable bank to another switchable bank must use a switching routine in the shared memory. Such a call cannot pass arguments which are addresses of functions, constants, or string literals in the calling bank. All pointers passed must be to objects in the shared memory.
12. A function which returns a structure cannot be used in an inter-bank function call if the returned structure is in memory in the calling bank. If the returned structure is an argument to another function, has a member of it accessed, or is assigned to a static or local variable, it is legal. If it is placed into switchable memory, by assigning to what is pointed at by a pointer, the operation will fail for an inter-bank function call.
13. The START macro in CRTFIRST must initialize the bankswitching port as necessary, and select the bank containing main( ) if it is not in the shared memory.

FUNCTION IN ASSEMBLER TO HANDLE SWITCHING OF BANKS

When bank switching must occur, the stack is set up by the compiler generated code for a normal function call. Instead of calling the destination function directly, however, the inter-bank link for the destination is called, as a result of the special manipulations with the linker LNHPC. This routine must change banks and then transfer to the destination, and must receive the return from the destination function so as to switch back to the original caller. This must be done transparently—no registers may be modified, and the stack must appear the same.

Included is the code to support the actual switching of banks during inter-bank function calls. This code allows a routine in either the shared memory or one of the switchable banks to make an inter-bank call to a routine in another bank.

The inter-bank link for each destination is created by a macro, invoked for each required linkage. The inter-bank link is simply a subroutine call to a common switching routine, with in-line arguments giving the bank and address of the destination. The common switching routine does the necessary manipulation of the stack to execute the destination and receive the return. The excess information is saved off in a separate software stack; upon return this information is used to restore the situation as if a normal function call had occurred.

Since the inter-bank transfer is completely transparent, it is not limited to handling C function calls. Any subroutine call which does not pass pointers to objects in switchable banks, which does not have in-line arguments, which does not use the Carry bit as either input or return, and which does not use a Return And Skip instruction, can be used with an inter-bank function call. However, the macro generates names using the C convention; an additional form is available for assembly subroutine names.

Also available is a version which allows the bank switching stack to be in 8-bit memory. It differs only in a few places from the 16-bit stack version.

The normal arrangement calls for the common switching routine and all the inter-bank links to be in shared memory. However, order of execution in the bank switching code is such that the inter-bank links for each destination that a bank needs can be in the switchable memory, and only the common routine needs to be in shared memory.

The software stack used by the bank switching is designed to grow downward, in contrast to the hardware stack, which grows upward. This allows the software stack to be placed in the same memory area as the hardware stack, but above it, and the two stacks will share their memory.

LINKAGE PROCEDURE FOR BANK SWITCHING

The actual linking of a multibank program is a series of individual linkages. The result will be a load module representing each bank's code, plus that bank's contribution to the shared memory area. It is essential that command files be used as inputs to LNHPC because each module must be linked several times, and changes would be ruinous:

First, each bank's set of modules must be linked independently. The Map files from each bank's linkage will give the necessary information on:

1. Undefined references, both functions and data.
2. A list of library routines invoked to support the code.
3. The size of the __init__info__ section for the bank.
4. The size of the total code.
5. The entry for the functions defined in that bank.
6. The address of the variables defined in the bank (which is applicable for shared bank only).

This information should be checked and validated. The undefined data references must be only to data which will be in the shared memory. The undefined function references should be for the function calls defined in other banks. The library routines invoked may be reduced by library routines which will be in the shared memory to support code there, or can be placed in shared memory to use the shared version for several banks. The size of each bank's _init_info_ section will be used to make dummy sections for the initial shared memory linkage (see next step below). Finally, the total size of the code, allowing for library routines which will be in the shared memory, must fit in the bank.

Second, an initial linkage of the shared memory is done to determine the addresses of routines and data which will be in there. This requires certain routines to be assembled:

1. The inter-bank switching routine and all the links needed for inter-bank function calls (their bank and address values are left out initially).
2. External references for any additional library routines to be forced into the shared memory.
3. Dummy _init_info_ sections which are each as large as the corresponding bank's real _init_info_ section (or one dummy section as large as all the bank's sections combined).

The shared memory is then linked with all of these items included, and the Map file will give valid addresses of data, functions, and sections.

Third, the banks can be linked to produce actual modules. All entry points in the shared memory are now defined, and need to be provided to the linkages of each bank. Assembly files providing the definitions is the simplest way to go. One file can provide the addresses of all user functions, library routines, and data variables in the shared memory, from the Map of the shared memory. Individual files need to be made to provide the addresses of the inter-bank links, because the links for a bank cannot be given to that bank. Additionally, the next available addresses need to be figured for each memory area. This provides linkage and layout by creating the new names and values to resolve the undefined references in the linkage; the linker will do the work of substituting the link address for the undefined function address. Then each bank can be linked, with the addresses for memory areas given to the linker, and the additional files defining shared memory and the other banks inter-bank links being linked in. After each bank, the next available addresses must be updated. Note that the _init_info_ sections must be contiguous and in the exact space created by the dummy routines.

Finally, the shared memory can be linked to produce the actual module. The banks and addresses must be provided for each inter-bank link and that module reassembled. The external references for additional library routines remains the same, and the dummy section for _init_info_ are unchanged. The Map of this linkage must be checked against the Map of initial linkage and/or against all addresses fed to the bank switched modules.

EXAMPLE CODE DISTRIBUTION

The example is a skeleton for a realtime program which accumulates time data into tables, then processes those tables by regression fit into a table of coefficients. The system then monitors further events and uses the coefficient to predict behavior as it occurs. The following files are to be in a system with two banks, from 0x4000 to 0x7fff.

TABLES.H	Data structure
MAIN.C	Main program, for shared bank
TABLES.C	Table accumulation and processing
MONITOR.C	Monitor external events and predict
ERRORS.C	Error routines
TIMERS.C	Timer initialization and interrupt service
UART.C	UART processing and interrupt service
CRTFIRST.ASM	Modified to set up Port B for Bankswitching
CRTFIRST.INC	<Standard module, unchanged>
BANKSWIT.ASM	<Standard module, unchanged>
BANKLINK.INC	Modified for inter-bank linkages
BANKDEFS.INC	Macro definitions to simplify linkages

The distribution shown in Table I is intended as an initial starting point. The monitor and prediction code is very large, and fills the bank. The table processing code has room left so the error routines (which are seldom called) are fit in there. This bank has RAM in it, which is not known to the compiler but is managed by the program. Main is in shared memory because it is the major loop of the program. Timers and UART are in shared because they contain the Interrupt Service Routines.

Shared	Bank 0	Bank 1
Main	Tables	Monitor
Timers	Errors	Strings
UART	Strings	Constants
Crtfirst	Constants	
Statics	Table RAM	
Initialization		
Printf		
Stack		
Bank Stack		
Crtinit		

All statics will be in shared memory. Initialization data is in shared memory. The string literals are all in ROM, and will be in banks; since these are passed as arguments to printf( ), printf( ) must either be in both banks or in shared memory in this case, to avoid duplication of memory usage and to save room in Bank 1. Constants are in banks, since inter-bank calls can be avoided when using constants and string literals. The stack and the bank stack are in the shared memory. The crtfirst routine is modified, and crtinit is with it in shared memory (although it may be possible to have crtinit in the bank selected by the START macro, this would require more manual linkage for the call in crtfirst).

LINKAGE PROCEDURE

Each bank load module is created by linking the banks separately. The linking is done in two steps. The first step is trial linkage and the parameters are specified in BANK0_1.CMD,BANK1_1.CMD and SHARED_1.CMD for linker. The information from this trial linkage is used in the second attempt where the load module is actually created. The command files used are BANK0_2.CMD, BANK1_2.CMD and SHARED_2.CMD.

Initial linker command files are:

```
SHARED_1.CMD
BANK0_1.CMD
BANK1_1.CMD
```

describing memory as

```
0000-01ff  shared: onchip RAM & I/O
0200-0fff  shared: offchip RAM
1000-3fff  shared: ROM
4000-7fff  banks
           bank 0: 4000-5fff  ROM
                   6000-7fff  RAM, private
           bank 1: 4000-7fff  ROM
8000-ffff  shared: ROM
```

where the private RAM is not mentioned to the linker. The private RAM is defined to the compiler using constants; another alternative would have been to define an assembler module of the proper size allocating the space, and place it with the linker. This would require another piece of assembly code, but would limit the address information to the linker command files.

During the trial linkage Bank 0 links but contains printf( ), which was desired to be in shared memory so it can be passed string literals; putchar( ) will also be there. This leaves only the variable live, which is just fine, will be placed in shared bank. The size of __init__info__ is 0x4136 to 0x4147 or 18 bytes (this information is best taken from the Section Table of the map). The code is not present; it is assumed to fit. For Bank 1, printf( ) will again be defined in shared; putchar( ) will not be referenced. The undefined for live, capture__table( ), and error( ) are correct. The size of __init__info__ is 0x409A to 0x409F or 6 bytes. The code is assumed to fit.

The initial linkage of the shared memory requires the creation of the linkage files. The linkages have to be put into BANKLINK.INC for all inter-bank entry points, including from shared to a bank. The sizes for the __init__info__ sections and the library access forcing requests are put in a file, using BANKDEFS.INC to make things easier. These are linked together, with the C stack and the switch stack in the off-chip RAM, with the switch stack on top so that they can share the same memory. There are few inter-bank calls, so the SWITCH__STACK__DEPTH used is 10. Linking this finishes the initial sequence, and the values are now available for the real second attempt of linking whereby the actual modules will be created.

Now the definitions to complete each bank are created. The module BANKDEFS.INC makes this easier. Each bank defines the linkages to entry points within that bank. The shared defines publics within the shared memory. (These values are best taken from the Symbol Table portion of the map.) Then the linker command files need to be modified (in the example new file names are used, but the user will probably not use new files, rather simply modify the existing files). The definition files needed for each bank will be added; these are the file for shared memory and for every other bank but this one. The No Output option is changed to giving a name for the object file, if desired, and the Ignore Errors is added because there is still no reset vector for a bank.

Finally, the memory addresses have to be determined from the shared load map and put into the command file (these values are best taken from the Memory Order Map, Memory Type Map, or the Section Table). The positioning of __init__info__ is critical, the others can have gaps. A trial linkage shows where the linker places modules, and final adjustments are required to ensure such placements meet the requirements. Bank 0 requires only that the initialization data be moved to shared memory. The updated addresses from Bank 0 are used in Bank 1. Bank 1 is placed acceptably by the linker.

The final linkage of the shared memory can now be done. Address and bank information is added to the linkage list. The remaining parts don't change. This linkage must be checked against the first linkage of shared to be certain no addresses have changed. Finally, the addresses used in each bank or shared should be checked against other banks to check for overlaps, and the types of sections in each memory should be checked to make sure all conventions have been met.

If everything is correct, you have load modules for the system.

The code listed in this Application Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this Application Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

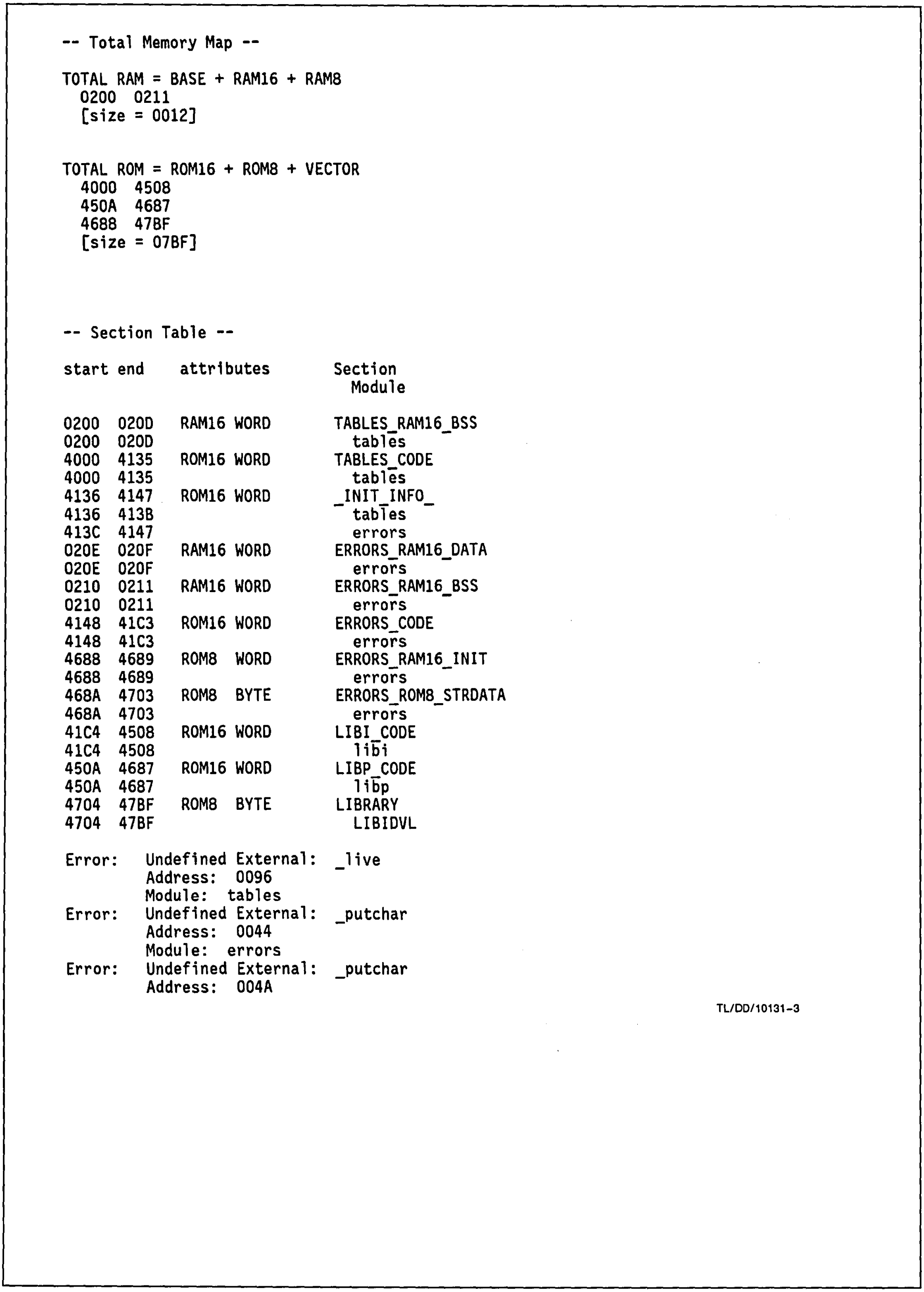
For Additional Information, Please Contact Factory

```
Contents of Linker command file BANKO__1.CMD:

/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank0__1.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x4000:0x5FFF,0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
tables,
errors
/NoOutput
```

TL/DD/10131-1

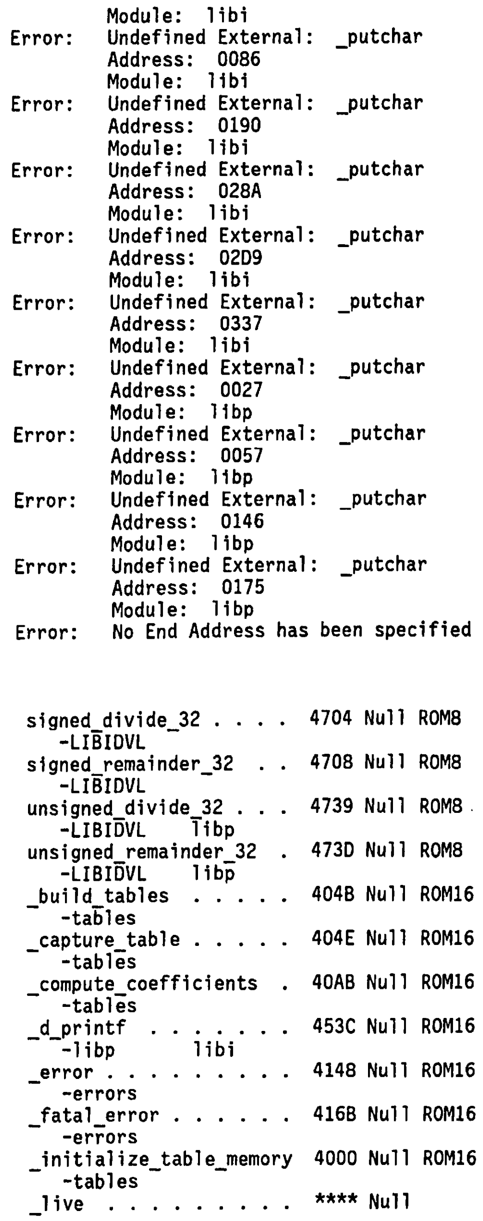
```
Contents of the Linker map file BANKO__1.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)                  09-May-88 08:37



Reset Vector:  0000

-- Range Definitions --

BASE    0002:00BF
ROM16   4000:5FFF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

-- Memory Order Map --

0200  0211   RAM16
4000  4508   ROM16
450A  4687   ROM16
4688  47BF   ROM8



-- Memory Type Map --

BASE
  [size = 0000]

RAM16
  0200  0211
  [size = 0012]

RAM8
  [size = 0000]

ROM16
  4000  4508
  450A  4687
  [size = 0687]

ROM8
  4688  47BF
  [size = 0138]

VECTOR
  [size = 0000]
```

TL/DD/10131-2



```
-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
  0200  0211
  [size = 0012]


TOTAL ROM = ROM16 + ROM8 + VECTOR
  4000  4508
  450A  4687
  4688  47BF
  [size = 07BF]



-- Section Table --

start end    attributes        Section
                                 Module

0200  020D   RAM16 WORD        TABLES_RAM16_BSS
0200  020D                       tables
4000  4135   ROM16 WORD        TABLES_CODE
4000  4135                       tables
4136  4147   ROM16 WORD        _INIT_INFO_
4136  413B                       tables
413C  4147                       errors
020E  020F   RAM16 WORD        ERRORS_RAM16_DATA
020E  020F                       errors
0210  0211   RAM16 WORD        ERRORS_RAM16_BSS
0210  0211                       errors
4148  41C3   ROM16 WORD        ERRORS_CODE
4148  41C3                       errors
4688  4689   ROM8  WORD        ERRORS_RAM16_INIT
4688  4689                       errors
468A  4703   ROM8  BYTE        ERRORS_ROM8_STRDATA
468A  4703                       errors
41C4  4508   ROM16 WORD        LIBI_CODE
41C4  4508                       libi
450A  4687   ROM16 WORD        LIBP_CODE
450A  4687                       libp
4704  47BF   ROM8  BYTE        LIBRARY
4704  47BF                       LIBIDVL

Error:   Undefined External:  _live
         Address:  0096
         Module:  tables
Error:   Undefined External:  _putchar
         Address:  0044
         Module:  errors
Error:   Undefined External:  _putchar
         Address:  004A
```

TL/DD/10131-3



```
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0086
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0190
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  028A
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  02D9
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0337
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0027
         Module:  libp
Error:   Undefined External:  _putchar
         Address:  0057
         Module:  libp
Error:   Undefined External:  _putchar
         Address:  0146
         Module:  libp
Error:   Undefined External:  _putchar
         Address:  0175
         Module:  libp
Error:   No End Address has been specified


 signed_divide_32 . . . .  4704 Null ROM8
    -LIBIDVL
 signed_remainder_32  . .  4708 Null ROM8
    -LIBIDVL
 unsigned_divide_32 . . .  4739 Null ROM8
    -LIBIDVL    libp
 unsigned_remainder_32  .  473D Null ROM8
    -LIBIDVL    libp
 _build_tables  . . . . .  404B Null ROM16
    -tables
 _capture_table . . . . .  404E Null ROM16
    -tables
 _compute_coefficients  .  40AB Null ROM16
    -tables
 _d_printf  . . . . . . .  453C Null ROM16
    -libp       libi
 _error . . . . . . . . .  4148 Null ROM16
    -errors
 _fatal_error . . . . . .  416B Null ROM16
    -errors
 _initialize_table_memory  4000 Null ROM16
    -tables
 _live  . . . . . . . . .  **** Null
```

TL/DD/10131-4

```
    tables
_printf . . . . . . . .  41C4 Null ROM16
   -libi      errors
_putchar . . . . . . . . **** Null
    errors    libi       libp
_quit . . . . . . . . .  41B0 Null ROM16
   -errors
_s_printf . . . . . . .  450A Null ROM16
   -libp      libi
_u_printf . . . . . . .  459A Null ROM16
   -libp      libi
```

TL/DD/10131-5

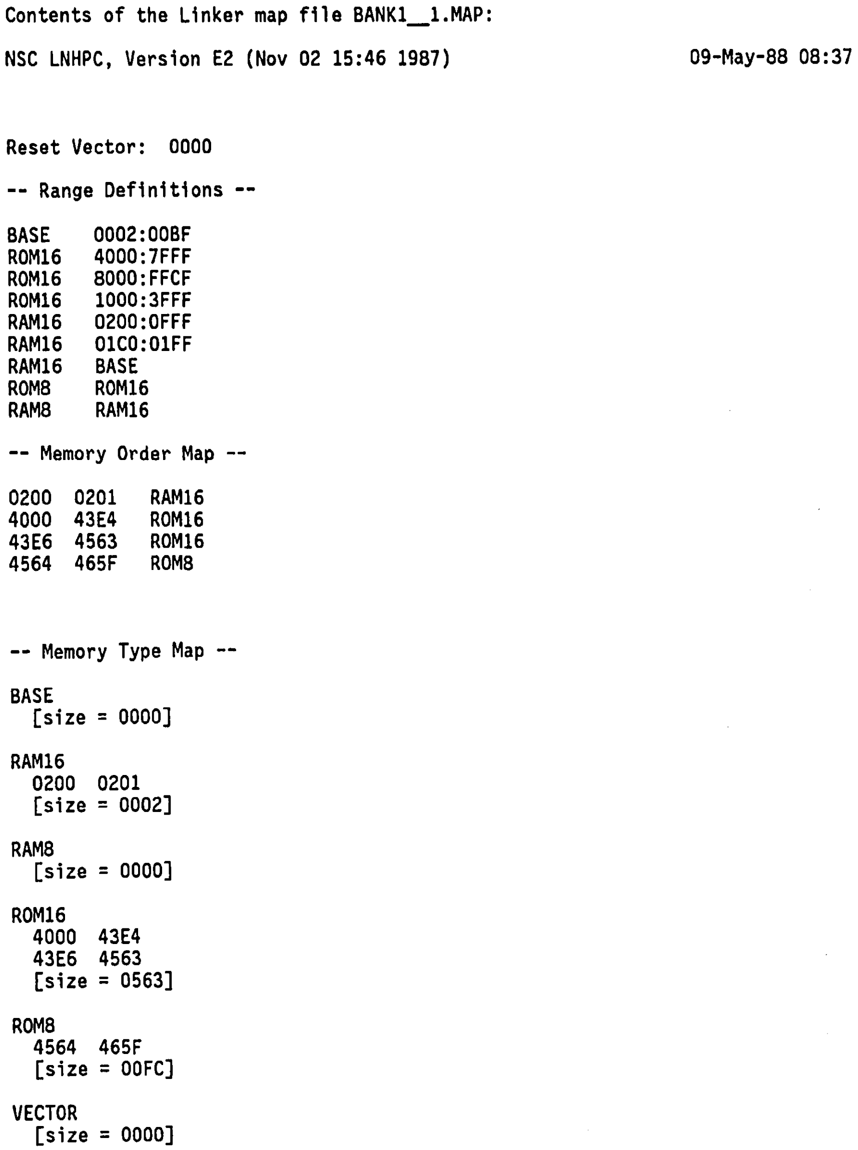
```
Information obtained from BANK0__1.MAP are:

1) The _INIT_INFO_ section size for Bank0 linkage is 18 bytes,
   ie from 0x4136 to 0x4147.

2) The entry address for functions obtained here as follows:
        Function                       Address

        initialize_table_memory        0x4000
        build_tables                   0x404b
        capture_table                  0x404e
        compute_coefficients           0x40ab
        error                          0x4136
        fatal_error                    0x4159

   These addresses will be used by the SWITCH_TO_FUNCTION assembly
   macro calls in the file BANKLINK.INC.

3) The undefined external reference for the variable live is expected,
   which will be defined in the SHARED bank. The undefined function
   putchar will also be defined in the shared bank.
```

TL/DD/10131-6

Contents of Linker command file BANK1__1.CMD:

```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank1__1.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x4000:0x7FFF,0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
monitor
/NoOutput
```

TL/DD/10131-7



```
Contents of the Linker map file BANK1__1.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)                    09-May-88 08:37


Reset Vector:  0000

-- Range Definitions --

BASE    0002:00BF
ROM16   4000:7FFF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

-- Memory Order Map --

0200  0201   RAM16
4000  43E4   ROM16
43E6  4563   ROM16
4564  465F   ROM8


-- Memory Type Map --

BASE
  [size = 0000]

RAM16
  0200  0201
  [size = 0002]

RAM8
  [size = 0000]

ROM16
  4000  43E4
  43E6  4563
  [size = 0563]

ROM8
  4564  465F
  [size = 00FC]

VECTOR
  [size = 0000]
```

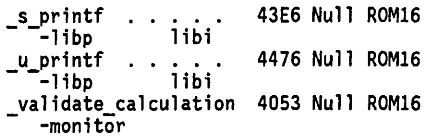
TL/DD/10131-8

```
-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
  0200  0201
  [size = 0002]


TOTAL ROM = ROM16 + ROM8 + VECTOR
  4000  43E4
  43E6  4563
  4564  465F
  [size = 065F]



-- Section Table --

start end    attributes        Section
                                 Module

0200  0201   RAM16 WORD        MONITOR_RAM16_BSS
0200  0201                       monitor
4000  4099   ROM16 WORD        MONITOR_CODE
4000  4099                       monitor
4564  45A3   ROM8  BYTE        MONITOR_ROM8_STRDATA
4564  45A3                       monitor
409A  409F   ROM16 WORD        _INIT_INFO_
409A  409F                       monitor
40A0  43E4   ROM16 WORD        LIBI_CODE
40A0  43E4                       libi
43E6  4563   ROM16 WORD        LIBP_CODE
43E6  4563                       libp
45A4  465F   ROM8  BYTE        LIBRARY
45A4  465F                       LIBIDVL

Error:    Undefined External:  _live
          Address:  0002
          Module:  monitor
Error:    Undefined External:  _live
          Address:  0012
          Module:  monitor
Error:    Undefined External:  _live
          Address:  0026
          Module:  monitor
Error:    Undefined External:  _capture_table
          Address:  0030
          Module:  monitor
Error:    Undefined External:  _error
          Address:  0091
          Module:  monitor
Error:    Undefined External:  _putchar
          Address:  004A
          Module:  libi
```

TL/DD/10131–9

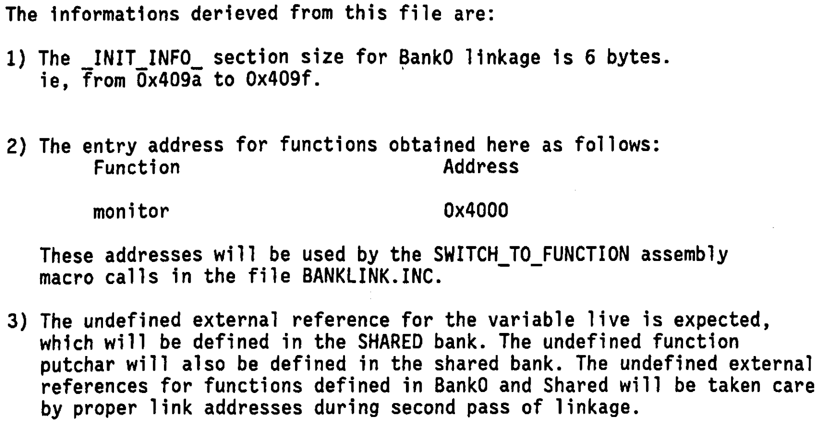
```
Error:   Undefined External:  _putchar
         Address:  0086
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0190
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  028A
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  02D9
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0337
         Module:  libi
Error:   Undefined External:  _putchar
         Address:  0027
         Module:  libp
Error:   Undefined External:  _putchar
         Address:  0057
         Module:  libp
Error:   Undefined External:  _putchar
         Address:  0146
         Module:  libp
Error:   Undefined External:  _putchar
         Address:  0175
         Module:  libp
Error:   No End Address has been specified


 signed_divide_32 . .   45A4 Null ROM8
    -LIBIDVL
 signed_remainder_32    45A8 Null ROM8
    -LIBIDVL
 unsigned_divide_32 .   45D9 Null ROM8
    -LIBIDVL    libp
 unsigned_remainder_32  45DD Null ROM8
    -LIBIDVL    libp
 _capture_table . . .   **** Null
     monitor
 _compute_prediction    4032 Null ROM16
    -monitor
 _d_printf  . . . . .   4418 Null ROM16
    -libp       libi
 _error . . . . . . .   **** Null
     monitor
 _live  . . . . . . .   **** Null
     monitor
 _monitor . . . . . .   4000 Null ROM16
    -monitor
 _printf  . . . . . .   40A0 Null ROM16
    -libi       monitor
 _putchar . . . . . .   **** Null
     libi       libp
```

TL/DD/10131-10



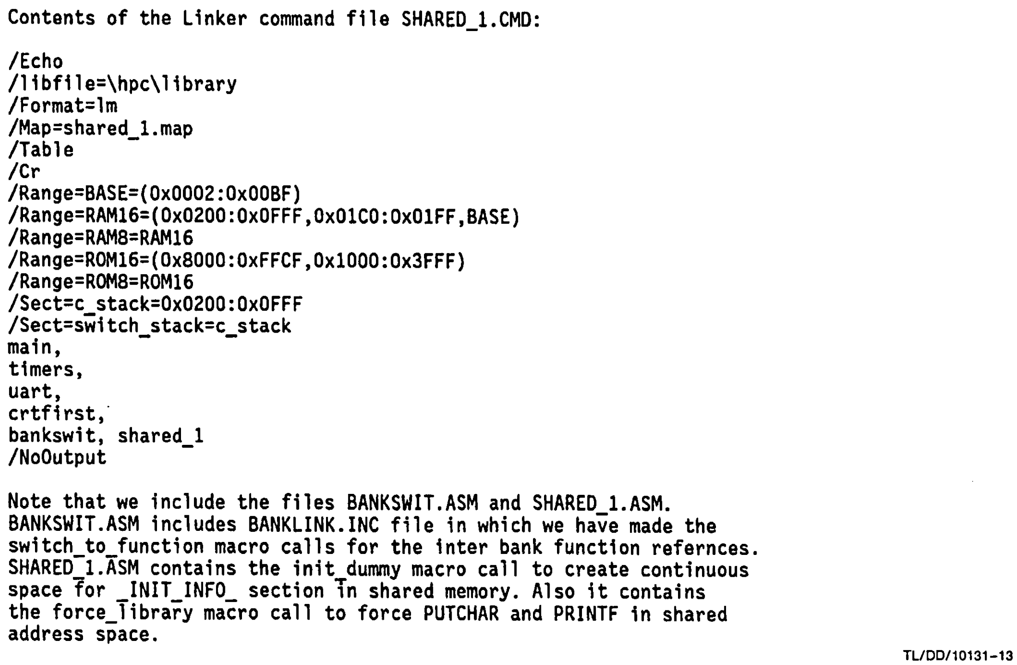
```
_s_printf . . . . .  43E6 Null ROM16
   -libp      libi
_u_printf . . . . .  4476 Null ROM16
   -libp      libi
_validate_calculation  4053 Null ROM16
   -monitor
```

TL/DD/10131-11



```
The informations derieved from this file are:

1) The _INIT_INFO_ section size for Bank0 linkage is 6 bytes.
   ie, from 0x409a to 0x409f.


2) The entry address for functions obtained here as follows:
        Function                        Address

        monitor                         0x4000

   These addresses will be used by the SWITCH_TO_FUNCTION assembly
   macro calls in the file BANKLINK.INC.

3) The undefined external reference for the variable live is expected,
   which will be defined in the SHARED bank. The undefined function
   putchar will also be defined in the shared bank. The undefined external
   references for functions defined in Bank0 and Shared will be taken care
   by proper link addresses during second pass of linkage.
```

TL/DD/10131-12



Contents of the Linker command file SHARED_1.CMD:

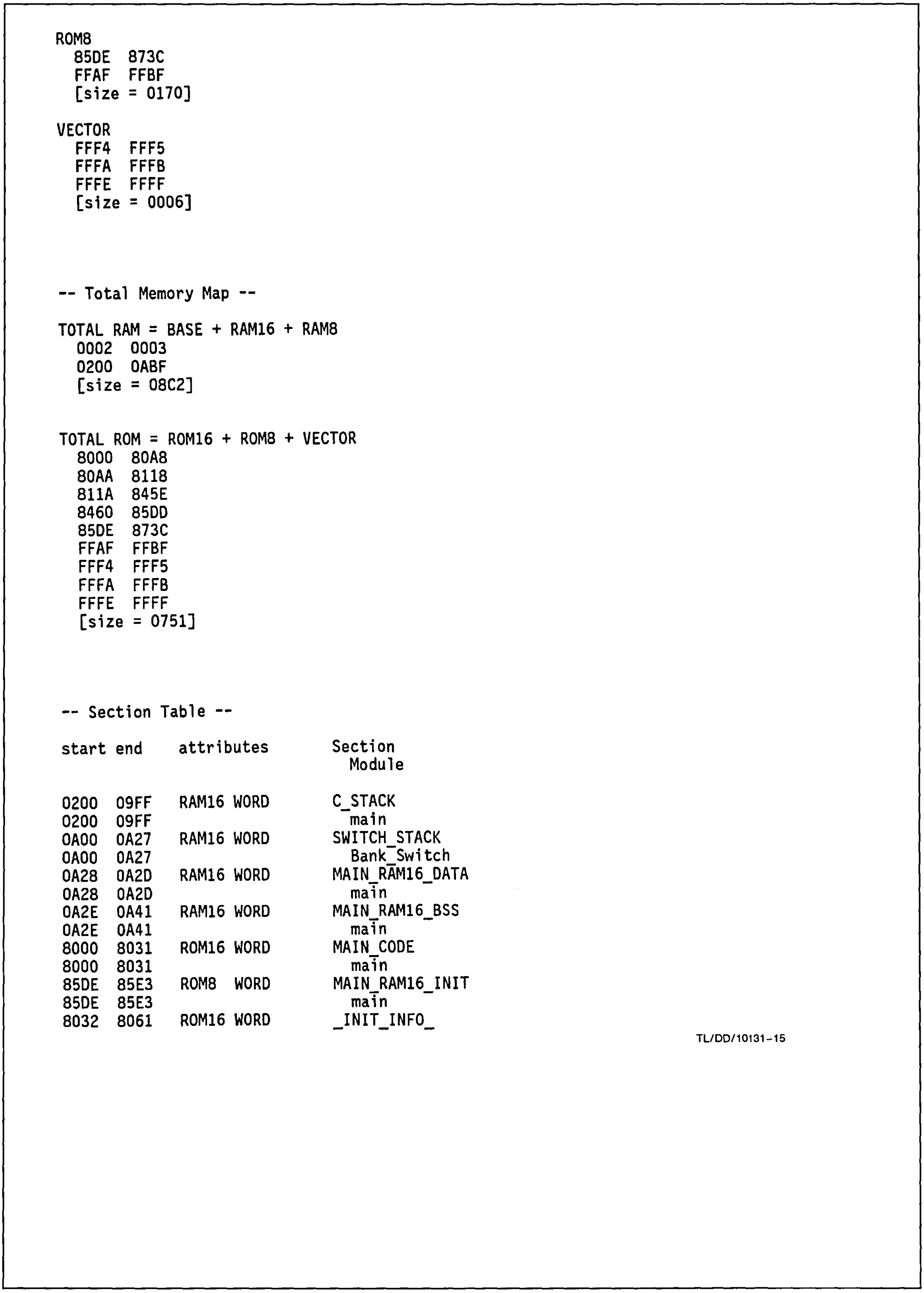
```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=shared_1.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
/Sect=c_stack=0x0200:0x0FFF
/Sect=switch_stack=c_stack
main,
timers,
uart,
crtfirst,
bankswit, shared_1
/NoOutput
```

Note that we include the files BANKSWIT.ASM and SHARED_1.ASM. BANKSWIT.ASM includes BANKLINK.INC file in which we have made the switch_to_function macro calls for the inter bank function refernces. SHARED_1.ASM contains the init_dummy macro call to create continuous space for _INIT_INFO_ section in shared memory. Also it contains the force_library macro call to force PUTCHAR and PRINTF in shared address space.

TL/DD/10131-13

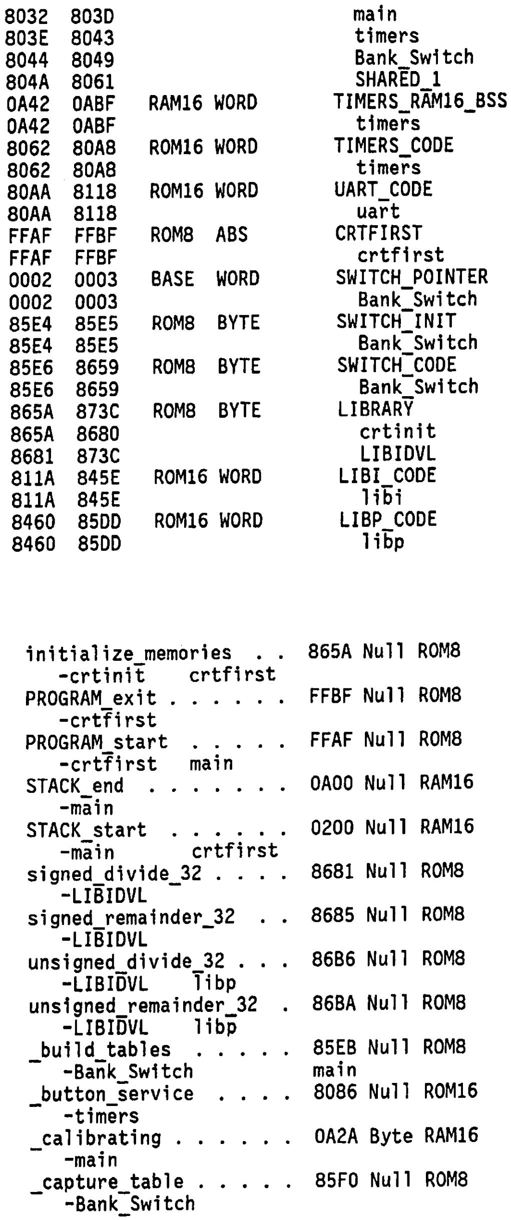
```
Contents of the Linker output file SHARED_1.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)                09-May-88 08:37


Reset Vector:  FFAF

-- Range Definitions --

BASE    0002:00BF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

-- Memory Order Map --

0002  0003   BASE
0200  0ABF   RAM16
8000  80A8   ROM16
80AA  8118   ROM16
811A  845E   ROM16
8460  85DD   ROM16
85DE  873C   ROM8
FFAF  FFBF   ROM8
FFF4  FFF5   VECTOR
FFFA  FFFB   VECTOR
FFFE  FFFF   VECTOR



-- Memory Type Map --

BASE
  0002  0003
  [size = 0002]

RAM16
  0200  0ABF
  [size = 08C0]

RAM8
  [size = 0000]

ROM16
  8000  80A8
  80AA  8118
  811A  845E
  8460  85DD
  [size = 05DB]
```

TL/DD/10131-14



```
ROM8
  85DE  873C
  FFAF  FFBF
  [size = 0170]

VECTOR
  FFF4  FFF5
  FFFA  FFFB
  FFFE  FFFF
  [size = 0006]



-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
  0002  0003
  0200  0ABF
  [size = 08C2]


TOTAL ROM = ROM16 + ROM8 + VECTOR
  8000  80A8
  80AA  8118
  811A  845E
  8460  85DD
  85DE  873C
  FFAF  FFBF
  FFF4  FFF5
  FFFA  FFFB
  FFFE  FFFF
  [size = 0751]



-- Section Table --

start end    attributes       Section
                                Module

0200  09FF   RAM16 WORD       C_STACK
0200  09FF                      main
0A00  0A27   RAM16 WORD       SWITCH_STACK
0A00  0A27                      Bank_Switch
0A28  0A2D   RAM16 WORD       MAIN_RAM16_DATA
0A28  0A2D                      main
0A2E  0A41   RAM16 WORD       MAIN_RAM16_BSS
0A2E  0A41                      main
8000  8031   ROM16 WORD       MAIN_CODE
8000  8031                      main
85DE  85E3   ROM8  WORD       MAIN_RAM16_INIT
85DE  85E3                      main
8032  8061   ROM16 WORD       _INIT_INFO_
```

TL/DD/10131-15



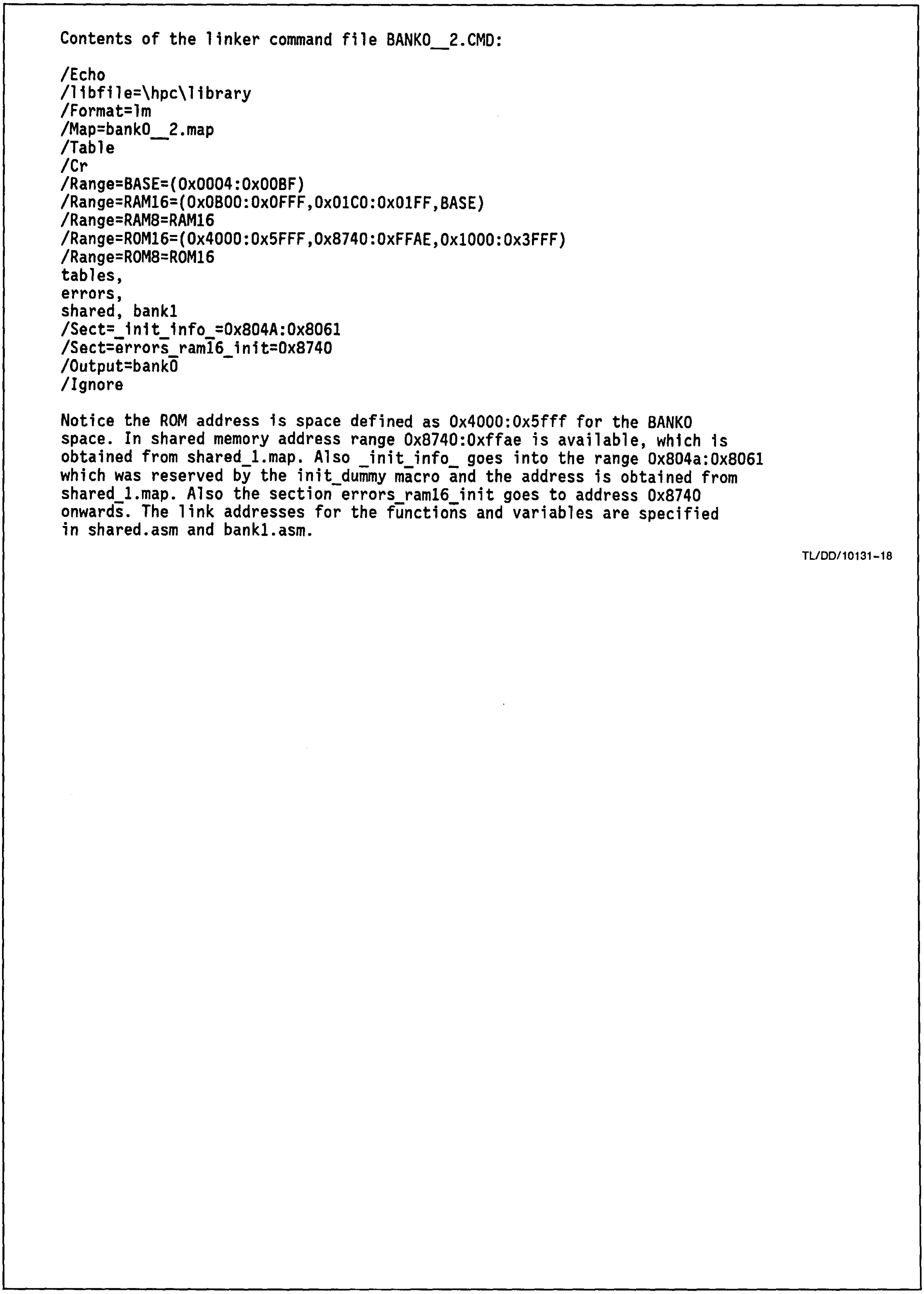
```
8032  803D                        main
803E  8043                        timers
8044  8049                        Bank_Switch
804A  8061                        SHARED_1
0A42  0ABF   RAM16 WORD         TIMERS_RAM16_BSS
0A42  0ABF                        timers
8062  80A8   ROM16 WORD         TIMERS_CODE
8062  80A8                        timers
80AA  8118   ROM16 WORD         UART_CODE
80AA  8118                        uart
FFAF  FFBF   ROM8  ABS          CRTFIRST
FFAF  FFBF                        crtfirst
0002  0003   BASE  WORD         SWITCH_POINTER
0002  0003                        Bank_Switch
85E4  85E5   ROM8  BYTE         SWITCH_INIT
85E4  85E5                        Bank_Switch
85E6  8659   ROM8  BYTE         SWITCH_CODE
85E6  8659                        Bank_Switch
865A  873C   ROM8  BYTE         LIBRARY
865A  8680                        crtinit
8681  873C                        LIBIDVL
811A  845E   ROM16 WORD         LIBI_CODE
811A  845E                        libi
8460  85DD   ROM16 WORD         LIBP_CODE
8460  85DD                        libp


  initialize_memories  . .  865A Null ROM8
     -crtinit    crtfirst
  PROGRAM_exit . . . . . .  FFBF Null ROM8
     -crtfirst
  PROGRAM_start  . . . . .  FFAF Null ROM8
     -crtfirst   main
  STACK_end  . . . . . . .  0A00 Null RAM16
     -main
  STACK_start  . . . . . .  0200 Null RAM16
     -main         crtfirst
  signed_divide_32 . . . .  8681 Null ROM8
     -LIBIDVL
  signed_remainder_32  . .  8685 Null ROM8
     -LIBIDVL
  unsigned_divide_32 . . .  86B6 Null ROM8
     -LIBIDVL    libp
  unsigned_remainder_32  .  86BA Null ROM8
     -LIBIDVL    libp
  _build_tables  . . . . .  85EB Null ROM8
      -Bank_Switch          main
  _button_service  . . . .  8086 Null ROM16
      -timers
  _calibrating . . . . . .  0A2A Byte RAM16
      -main
  _capture_table . . . . .  85F0 Null ROM8
      -Bank_Switch
```

TL/DD/10131-16

```
 _coefficients . . . . .  0A2E Byte RAM16
    -main
 _compute_coefficients  .  85F5 Null ROM8
    -Bank_Switch           main
 _d_printf . . . . . . .  8492 Null ROM16
    -libp       libi
 _error . . . . . . . . .  85FF Null ROM8
    -Bank_Switch
 _fatal_error . . . . . .  8604 Null ROM8
    -Bank_Switch
 _initialize_inputs . . .  8062 Null ROM16
    -timers     main
 _initialize_outputs  . .  80AA Null ROM16
    -uart       main
 _initialize_table_memory  85E6 Null ROM8
    -Bank_Switch           main
 _live  . . . . . . . . .  0A42 Byte RAM16
    -timers
 _main  . . . . . . . . .  8000 Null ROM16
    -main       crtfirst
 _monitor . . . . . . . .  85FA Null ROM8
    -Bank_Switch           main
 _operational . . . . . .  0A28 Byte RAM16
    -main
 _predicting  . . . . . .  0A2C Byte RAM16
    -main
 _printf  . . . . . . . .  811A Null ROM16
    -libi       SHARED_1
 _put_uart  . . . . . . .  810E Null ROM16
    -uart
 _putchar . . . . . . . .  80AB Null ROM16
    -uart       libi       libp
 _s_printf  . . . . . . .  8460 Null ROM16
    -libp       libi
 _timer_service . . . . .  8063 Null ROM16
    -timers
 _u_printf  . . . . . . .  84F0 Null ROM16
    -libp       libi
```

Notice that there is entry for each function that is actully placed in switchable bank being called from other banks. These entries are used as link addresses for the respective functions when linking the banks individually. Refer the files shared.asm, bank0.asm and bank1.asm.

TL/DD/10131-17



Contents of the linker command file BANK0__2.CMD:

```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank0__2.map
/Table
/Cr
/Range=BASE=(0x0004:0x00BF)
/Range=RAM16=(0x0B00:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x4000:0x5FFF,0x8740:0xFFAE,0x1000:0x3FFF)
/Range=ROM8=ROM16
tables,
errors,
shared, bank1
/Sect=_init_info_=0x804A:0x8061
/Sect=errors_ram16_init=0x8740
/Output=bank0
/Ignore
```

Notice the ROM address is space defined as 0x4000:0x5fff for the BANK0 space. In shared memory address range 0x8740:0xffae is available, which is obtained from shared_1.map. Also _init_info_ goes into the range 0x804a:0x8061 which was reserved by the init_dummy macro and the address is obtained from shared_1.map. Also the section errors_ram16_init goes to address 0x8740 onwards. The link addresses for the functions and variables are specified in shared.asm and bank1.asm.

TL/DD/10131-18

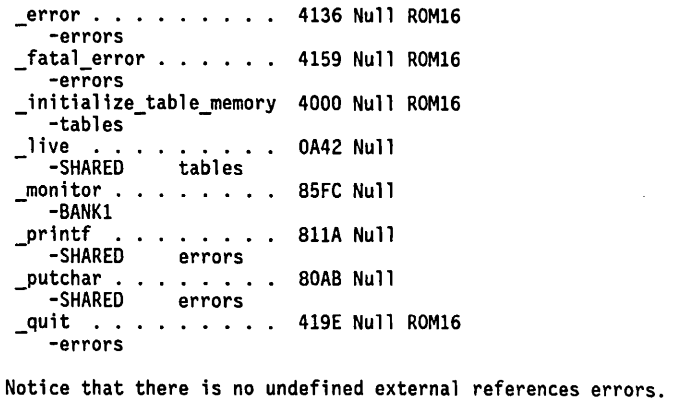
Contents of the Linker output file BANK0__2.MAP:

```
NSC LNHPC, Version E2 (Nov 02 15:46 1987)                    09-May-88 08:38


Reset Vector:  0000

-- Range Definitions --

BASE     0004:00BF
ROM16    4000:5FFF
ROM16    8740:FFAE
ROM16    1000:3FFF
RAM16    0B00:0FFF
RAM16    01C0:01FF
RAM16    BASE
ROM8     ROM16
RAM8     RAM16

-- Memory Order Map --

0B00  0B11   RAM16
4000  41B1   ROM16
41B2  422B   ROM8
804A  805B   ROM16
8740  8741   ROM8



-- Memory Type Map --

BASE
  [size = 0000]

RAM16
  0B00  0B11
  [size = 0012]

RAM8
  [size = 0000]

ROM16
  4000  41B1
  804A  805B
  [size = 01C4]

ROM8
  41B2  422B
  8740  8741
  [size = 007C]

VECTOR
  [size = 0000]
```

TL/DD/10131-19

```
-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
  0B00  0B11
  [size = 0012]


TOTAL ROM = ROM16 + ROM8 + VECTOR
  4000  41B1
  41B2  422B
  804A  805B
  8740  8741
  [size = 0240]



-- Section Table --

start end    attributes       Section
                                Module

804A  805B   ROM16 WORD       _INIT_INFO_
804A  804F                      tables
8050  805B                      errors
8740  8741   ROM8  WORD       ERRORS_RAM16_INIT
8740  8741                      errors
0B00  0B0D   RAM16 WORD       TABLES_RAM16_BSS
0B00  0B0D                      tables
4000  4135   ROM16 WORD       TABLES_CODE
4000  4135                      tables
0B0E  0B0F   RAM16 WORD       ERRORS_RAM16_DATA
0B0E  0B0F                      errors
0B10  0B11   RAM16 WORD       ERRORS_RAM16_BSS
0B10  0B11                      errors
4136  41B1   ROM16 WORD       ERRORS_CODE
4136  41B1                      errors
41B2  422B   ROM8  BYTE       ERRORS_ROM8_STRDATA
41B2  422B                      errors

Error:   No End Address has been specified


 _build_tables . . . . .  404B Null ROM16
    -tables
 _capture_table . . . . . 404E Null ROM16
    -tables
 _coefficients  . . . . . 0A2E Null
    -SHARED
 _compute_coefficients  . 40AB Null ROM16
    -tables
```

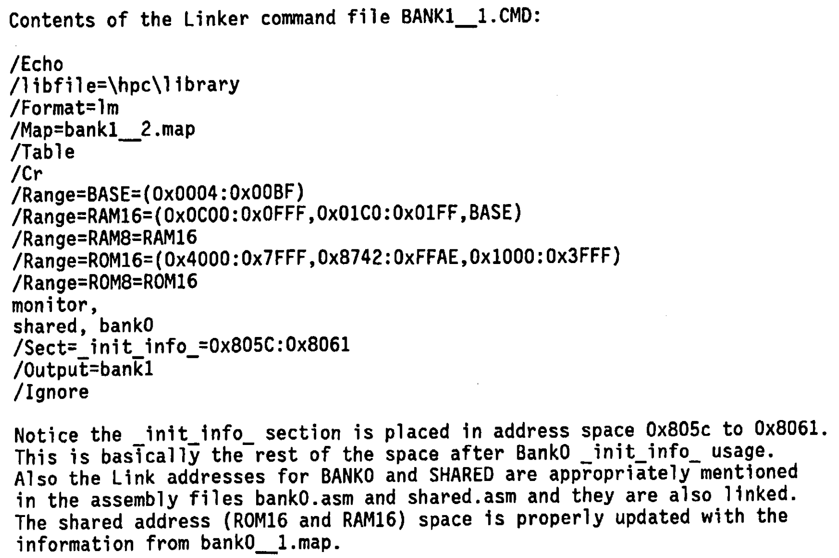
TL/DD/10131-20



```
    _error . . . . . . . . .  4136 Null ROM16
       -errors
    _fatal_error . . . . . .  4159 Null ROM16
       -errors
    _initialize_table_memory  4000 Null ROM16
       -tables
    _live  . . . . . . . . .  0A42 Null
       -SHARED     tables
    _monitor . . . . . . . .  85FC Null
       -BANK1
    _printf  . . . . . . . .  811A Null
       -SHARED     errors
    _putchar . . . . . . . .  80AB Null
       -SHARED     errors
    _quit  . . . . . . . . .  419E Null ROM16
       -errors

   Notice that there is no undefined external references errors.
```

TL/DD/10131-21

Since the function Main is not defined in this bank there is no reset vector address defined and hence the Linker gives the 'no end address specified' error message, which can be ignored.

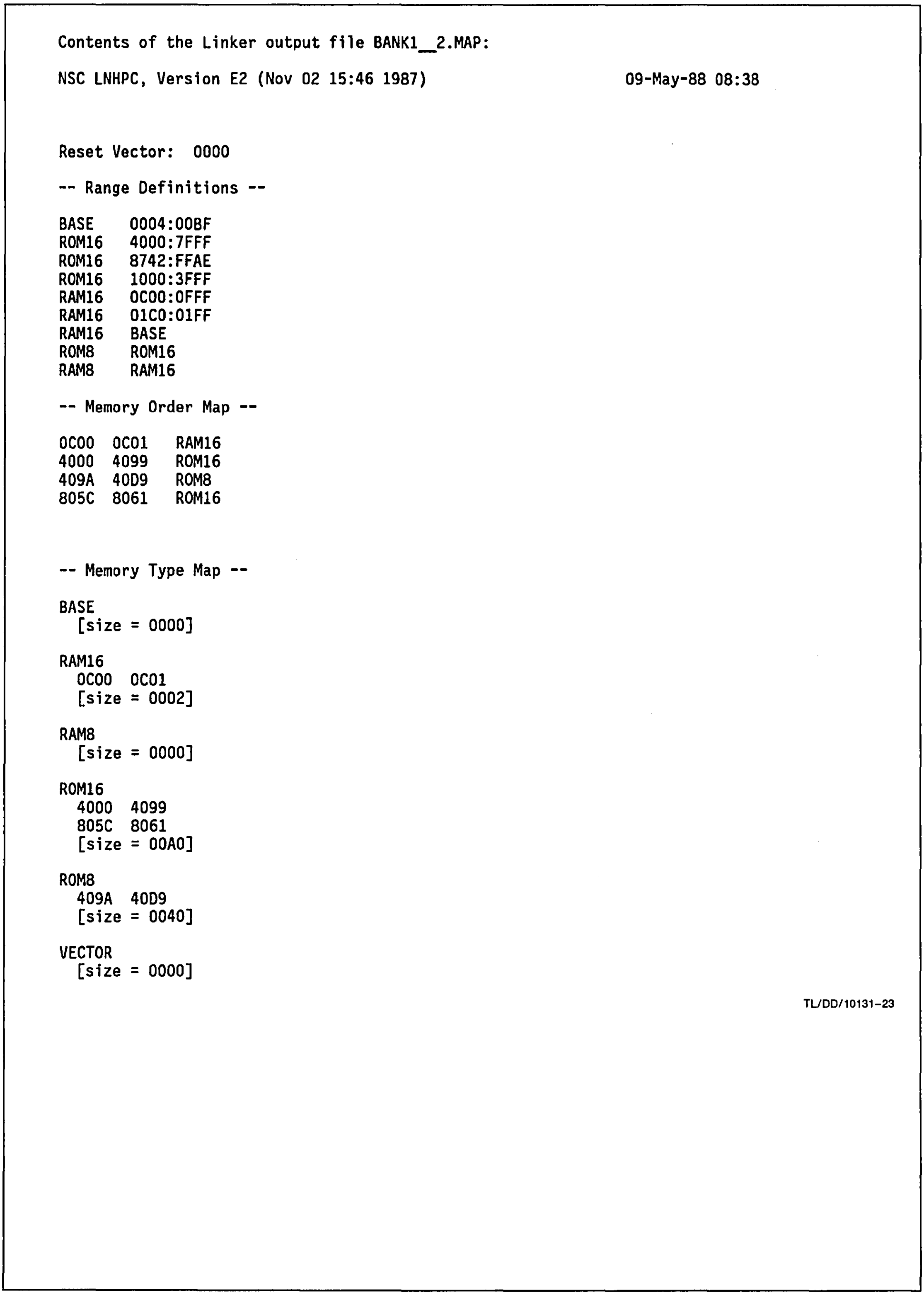


Contents of the Linker command file BANK1__1.CMD:

```
/Echo
/libfile=\hpc\library
/Format=lm
/Map=bank1__2.map
/Table
/Cr
/Range=BASE=(0x0004:0x00BF)
/Range=RAM16=(0x0C00:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x4000:0x7FFF,0x8742:0xFFAE,0x1000:0x3FFF)
/Range=ROM8=ROM16
monitor,
shared, bank0
/Sect=_init_info_=0x805C:0x8061
/Output=bank1
/Ignore
```

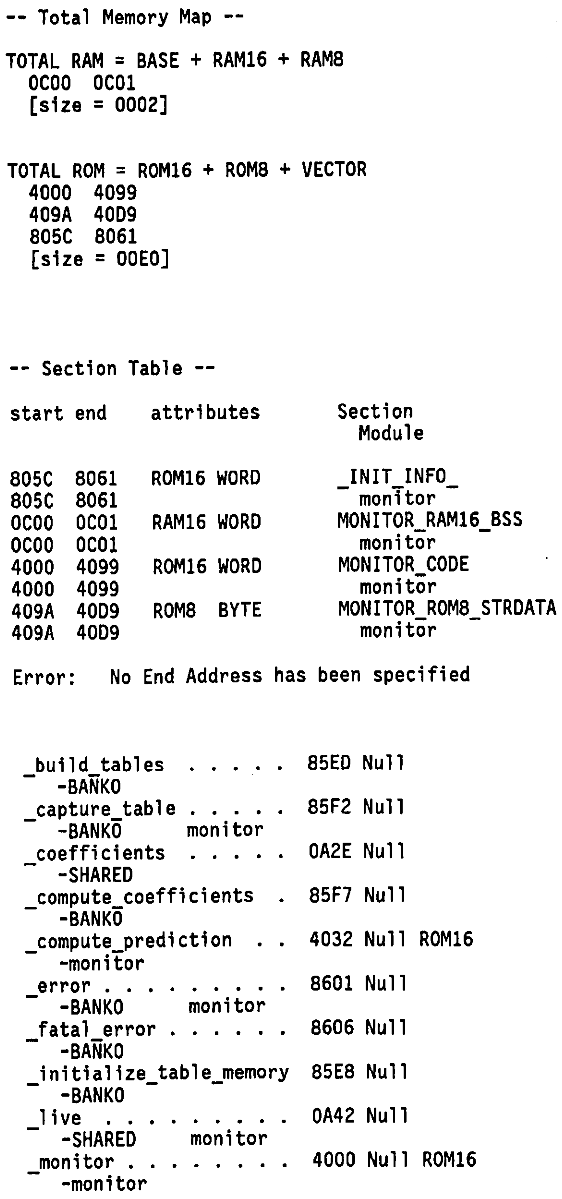
Notice the _init_info_ section is placed in address space 0x805c to 0x8061.
This is basically the rest of the space after Bank0 _init_info_ usage.
Also the Link addresses for BANK0 and SHARED are appropriately mentioned
in the assembly files bank0.asm and shared.asm and they are also linked.
The shared address (ROM16 and RAM16) space is properly updated with the
information from bank0__1.map.

TL/DD/10131-22



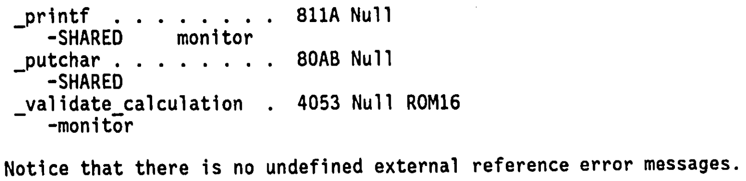
```
Contents of the Linker output file BANK1__2.MAP:

NSC LNHPC, Version E2 (Nov 02 15:46 1987)                    09-May-88 08:38


Reset Vector:  0000

-- Range Definitions --

BASE    0004:00BF
ROM16   4000:7FFF
ROM16   8742:FFAE
ROM16   1000:3FFF
RAM16   0C00:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

-- Memory Order Map --

0C00  0C01   RAM16
4000  4099   ROM16
409A  40D9   ROM8
805C  8061   ROM16



-- Memory Type Map --

BASE
  [size = 0000]

RAM16
  0C00  0C01
  [size = 0002]

RAM8
  [size = 0000]

ROM16
  4000  4099
  805C  8061
  [size = 00A0]

ROM8
  409A  40D9
  [size = 0040]

VECTOR
  [size = 0000]
```

TL/DD/10131-23



```
-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
  0C00  0C01
  [size = 0002]


TOTAL ROM = ROM16 + ROM8 + VECTOR
  4000  4099
  409A  40D9
  805C  8061
  [size = 00E0]



-- Section Table --

start end    attributes       Section
                                Module

805C  8061   ROM16 WORD       _INIT_INFO_
805C  8061                      monitor
0C00  0C01   RAM16 WORD       MONITOR_RAM16_BSS
0C00  0C01                      monitor
4000  4099   ROM16 WORD       MONITOR_CODE
4000  4099                      monitor
409A  40D9   ROM8  BYTE       MONITOR_ROM8_STRDATA
409A  40D9                      monitor

Error:   No End Address has been specified


 _build_tables  . . . . .  85ED Null
   -BANK0
 _capture_table . . . . .  85F2 Null
   -BANK0      monitor
 _coefficients  . . . . .  0A2E Null
   -SHARED
 _compute_coefficients  .  85F7 Null
   -BANK0
 _compute_prediction  . .  4032 Null ROM16
   -monitor
 _error . . . . . . . . .  8601 Null
   -BANK0      monitor
 _fatal_error . . . . . .  8606 Null
   -BANK0
 _initialize_table_memory  85E8 Null
   -BANK0
 _live  . . . . . . . . .  0A42 Null
   -SHARED     monitor
 _monitor . . . . . . . .  4000 Null ROM16
   -monitor
```

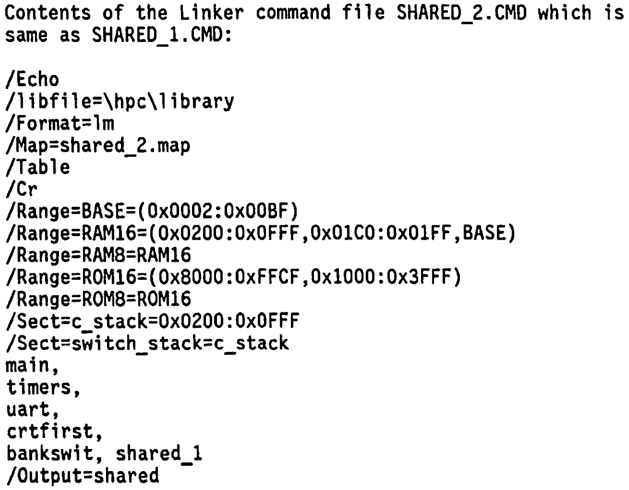
TL/DD/10131-24



```
   _printf . . . . . . . .  811A Null
      -SHARED     monitor
   _putchar . . . . . . . . 80AB Null
      -SHARED
   _validate_calculation  . 4053 Null ROM16
      -monitor

  Notice that there is no undefined external reference error messages.
```

TL/DD/10131-25

Since the function Main is not defined in this bank there is no reset vector address defined and hence the Linker gives the 'no end address specified' error message, which can be ignored.



```
Contents of the Linker command file SHARED_2.CMD which is
same as SHARED_1.CMD:

/Echo
/libfile=\hpc\library
/Format=lm
/Map=shared_2.map
/Table
/Cr
/Range=BASE=(0x0002:0x00BF)
/Range=RAM16=(0x0200:0x0FFF,0x01C0:0x01FF,BASE)
/Range=RAM8=RAM16
/Range=ROM16=(0x8000:0xFFCF,0x1000:0x3FFF)
/Range=ROM8=ROM16
/Sect=c_stack=0x0200:0x0FFF
/Sect=switch_stack=c_stack
main,
timers,
uart,
crtfirst,
bankswit, shared_1
/Output=shared
```

TL/DD/10131-26

Contents of the Linker output file SHARED_2.MAP which should be identical to SHARED_1.MAP:

```
NSC LNHPC, Version E2 (Nov 02 15:46 1987)                      09-May-88 08:38



Reset Vector:  FFAF

-- Range Definitions --

BASE    0002:00BF
ROM16   8000:FFCF
ROM16   1000:3FFF
RAM16   0200:0FFF
RAM16   01C0:01FF
RAM16   BASE
ROM8    ROM16
RAM8    RAM16

-- Memory Order Map --

0002  0003   BASE
0200  0ABF   RAM16
8000  80A8   ROM16
80AA  8118   ROM16
811A  845E   ROM16
8460  85DD   ROM16
85DE  873C   ROM8
FFAF  FFBF   ROM8
FFF4  FFF5   VECTOR
FFFA  FFFB   VECTOR
FFFE  FFFF   VECTOR



-- Memory Type Map --

BASE
  0002  0003
  [size = 0002]

RAM16
  0200  0ABF
  [size = 08C0]

RAM8
  [size = 0000]

ROM16
  8000  80A8
  80AA  8118
  811A  845E
  8460  85DD
  [size = 05DB]
```

TL/DD/10131-27

```
ROM8
  85DE  873C
  FFAF  FFBF
  [size = 0170]

VECTOR
  FFF4  FFF5
  FFFA  FFFB
  FFFE  FFFF
  [size = 0006]



-- Total Memory Map --

TOTAL RAM = BASE + RAM16 + RAM8
  0002  0003
  0200  0ABF
  [size = 08C2]


TOTAL ROM = ROM16 + ROM8 + VECTOR
  8000  80A8
  80AA  8118
  811A  845E
  8460  85DD
  85DE  873C
  FFAF  FFBF
  FFF4  FFF5
  FFFA  FFFB
  FFFE  FFFF
  [size = 0751]



-- Section Table --

start end    attributes       Section
                                Module

0200  09FF   RAM16 WORD       C_STACK
0200  09FF                      main
0A00  0A27   RAM16 WORD       SWITCH_STACK
0A00  0A27                      Bank_Switch
0A28  0A2D   RAM16 WORD       MAIN_RAM16_DATA
0A28  0A2D                      main
0A2E  0A41   RAM16 WORD       MAIN_RAM16_BSS
0A2E  0A41                      main
8000  8031   ROM16 WORD       MAIN_CODE
8000  8031                      main
85DE  85E3   ROM8  WORD       MAIN_RAM16_INIT
85DE  85E3                      main
```

TL/DD/10131-28

```
8032  8061   ROM16 WORD       _INIT_INFO_
8032  803D                      main
803E  8043                      timers
8044  8049                      Bank_Switch
804A  8061                      SHARED_1
0A42  0ABF   RAM16 WORD       TIMERS_RAM16_BSS
0A42  0ABF                      timers
8062  80A8   ROM16 WORD       TIMERS_CODE
8062  80A8                      timers
80AA  8118   ROM16 WORD       UART_CODE
80AA  8118                      uart
FFAF  FFBF   ROM8  ABS        CRTFIRST
FFAF  FFBF                      crtfirst
0002  0003   BASE  WORD       SWITCH_POINTER
0002  0003                      Bank_Switch
85E4  85E5   ROM8  BYTE       SWITCH_INIT
85E4  85E5                      Bank_Switch
85E6  8659   ROM8  BYTE       SWITCH_CODE
85E6  8659                      Bank_Switch
865A  873C   ROM8  BYTE       LIBRARY
865A  8680                      crtinit
8681  873C                      LIBIDVL
811A  845E   ROM16 WORD       LIBI_CODE
811A  845E                      libi
8460  85DD   ROM16 WORD       LIBP_CODE
8460  85DD                      libp
```

```
initialize_memories  . .  865A Null ROM8
   -crtinit    crtfirst
PROGRAM_exit . . . . . .  FFBF Null ROM8
   -crtfirst
PROGRAM_start  . . . . .  FFAF Null ROM8
   -crtfirst   main
STACK_end  . . . . . . .  0A00 Null RAM16
   -main
STACK_start  . . . . . .  0200 Null RAM16
   -main       crtfirst
signed_divide_32 . . . .  8681 Null ROM8
   -LIBIDVL
signed_remainder_32  . .  8685 Null ROM8
   -LIBIDVL
unsigned_divide_32 . . .  86B6 Null ROM8
   -LIBIDVL    libp
unsigned_remainder_32  .  86BA Null ROM8
   -LIBIDVL    libp
_build_tables  . . . . .  85EB Null ROM8
   -Bank_Switch           main
_button_service  . . . .  8086 Null ROM16
   -timers
_calibrating . . . . . .  0A2A Byte RAM16
   -main
_capture_table . . . . .  85F0 Null ROM8
```

TL/DD/10131-29

```
   -Bank_Switch
_coefficients . . . . .  0A2E Byte RAM16
   -main
_compute_coefficients  .  85F5 Null ROM8
   -Bank_Switch           main
_d_printf . . . . . . .  8492 Null ROM16
   -libp      libi
_error . . . . . . . . .  85FF Null ROM8
   -Bank_Switch
_fatal_error . . . . . .  8604 Null ROM8
   -Bank_Switch
_initialize_inputs . . .  8062 Null ROM16
   -timers    main
_initialize_outputs  . .  80AA Null ROM16
   -uart      main
_initialize_table_memory  85E6 Null ROM8
   -Bank_Switch           main
_live  . . . . . . . . .  0A42 Byte RAM16
   -timers
_main  . . . . . . . . .  8000 Null ROM16
   -main      crtfirst
_monitor . . . . . . . .  85FA Null ROM8
   -Bank_Switch           main
_operational . . . . . .  0A28 Byte RAM16
   -main
_predicting  . . . . . .  0A2C Byte RAM16
   -main
_printf  . . . . . . . .  811A Null ROM16
   -libi      SHARED_1
_put_uart  . . . . . . .  810E Null ROM16
   -uart
_putchar . . . . . . . .  80AB Null ROM16
   -uart      libi        libp
_s_printf  . . . . . . .  8460 Null ROM16
   -libp      libi
_timer_service . . . . .  8063 Null ROM16
   -timers
_u_printf  . . . . . . .  84F0 Null ROM16
   -libp      libi
```

After final linkages the shared bank address space in the map files BANK0__2.MAP, BANK1__2.MAP and SHARED_2.MAP should be verified for no memory overlap.

TL/DD/10131-30

```
;       *********************************************************
;       *                                                       *
;       *       National Semiconductor MicroController Group    *
;       *                                                       *
;       *       HPC C Compiler Support and Library Routines     *
;       *       C Run Time Initialization User Tunable Code     *
;       *       CRTFIRST.ASM -  C run time initialization       *
;       *********************************************************

;Copyright (c) 1987, National Semiconductor, Santa Clara Ca 95051

;See CRTFIRST.INC source code for explanation of macros and usage

;Code origin
        .sect   crtfirst,rom8,abs=0xffaf

        ld      0x00f3.b,#0xff          ;output pins for upper Port B
        ld      0x00e3.b,#0x00          ;select bank 0


        jp

        .incld  crtfirst.inc

        .end    PROGRAM_start
```

TL/DD/10131-31

```
; SHARED_1.ASM - Bank switch support function
; To force library functions onto shared bank and to
; allocate continuous space for _init_info_ section on the
; shared bank.
;
.incld  bankdefs.inc

force_library   printf

init_dummy      18, 6

.end

; BANK0.ASM - Link address for functions actually defined
; in bank0

.incld  bankdefs.inc

link_address    initialize_table_memory, 0x85e8
link_address    build_tables, 0x85ed
link_address    capture_table, 0x85f2
link_address    compute_coefficients, 0x85f7
link_address    error, 0x8601
link_address    fatal_error, 0x8606

.end


; BANK1.ASM - Link address for the function actually defined
; in bank1.

.incld  bankdefs.inc

link_address    monitor, 0x85fc

.end


; SHARED.ASM - Link address for the functions and variables
; defined in shared address space.

.incld  bankdefs.inc

link_address    printf, 0x811a
link_address    putchar, 0x80ab
link_address    live, 0x0a42
link_address    coefficients, 0x0a2e

.end
```

TL/DD/10131-41

```
        .title  crtfirst, 'C Run Time Initialization'
;       *******************************************************
;       *                                                     *
;       *       National Semiconductor MicroController Group  *
;       *                                                     *
;       *       HPC C Compiler Support and Library Routines   *
;       *       CRTFIRST.INC - C Run Time Initialization      *
;       *                                                     *
;       *******************************************************

;Copyright (c) 1987, National Semiconductor, Santa Clara Ca 95051

;Edit History
; 12/15/86 DKL  Create from CCHPC startup output
; 2/6/87 DKL    Convert to new Assembler Syntax
; 2/9/87 DKL    Seperate out Tunable Code
; 3/4/87 RPG    Modify to suit new compiler
; 3/10/87 DKL   Changes to DKL arrangement, initialize memory
; 3/20/87 DKL   Stack out, efficient list order in
; 5/6/87 DKL    Make this the included, not includer, file
; 7/27/87 DKL   Move Initialization of RAM to separate subroutine
;

        .public PROGRAM_start, PROGRAM_exit
        .extrn  _main
     .ifndef memories_8bit
        .extrn  initialize_memories
     .else
        .extrn  initialize_memories_8bit
     .endif
        .extrn  STACK_start

        .form
;This routine provides the standard C RunTime Routine for starting a
;compiled and linked program.  It initializes the stack pointer and
;RAM memories, and enters the compiler generated code in function
;"main()" with no arguments.

;Four macros are used to allow the end user to have control of the start
;process at key moments, before the C code begins execution.  The macros
;used are ORIGIN, START, READY, and HALT, in the following fashion:
;
;       ORIGIN
;PROGRAM_start:
;       ld      sp,<stack>
;       START
;       jsrl    initialize_memories
;       READY
;       jsrl    _main
;PROGRAM_exit:
;       HALT
;
;Code size is tested to ensure that the code does not overwrite any
;dedicated addresses (e.g., subroutine jump table), and optionally to
```

TL/DD/10131-32

```
;ensure that no space is wasted between the end and the dedicated area.
;The dedicated address is defined as ADDRESS_limit, and the check for
;waste space is controlled by ORIGIN_check being non-zero.  Either of
;these may be redefined by the user in the ORIGIN macro.

;ORIGIN macro
;Must declare the section and set the absolute origin for the startup
;code.  Code must end before any dedicated addresses (ADDRESS_limit),
;and should not waste any space.  If any of the other macros here are
;lengthened, this must be adjusted.  Might optionally redefine values
;of ADDRESS_limit or ORIGIN_check.
;

;START macro
;Code to execute after the stack pointer is initialized, and before the
;memories are initialized.  Must enable the appropriate configuration
;options for the chip, so that memories can be accessed.  Since all
;memories can be accessed, the list of RAM memories can be accessed
;where ever it may be.
;

;READY macro
;Code to execute after memory is initialized, but before the C code is
;entered.
;

;HALT macro
;Code to execute when the C code terminates.
;

;Limit address of code for this routine (first dedicated address)

;Whether to check that the origin provided is exactly correct

       .form
;C RunTime Initialization Startup Code
       ORIGIN  ;declares absolute section and defines address

PROGRAM_start:
       ld      sp,#STACK_start        ;Initialize stack
       START                          ;User code option
 .ifndef memories_8bit
       jsrl    initialize_memories
 .else
       jsrl    initialize_memories_8bit
 .endif
       READY
       jsrl    _main
PROGRAM_exit:
       HALT

origin= ADDRESS_limit - . + PROGRAM_start
 .if . > ADDRESS_limit
       .ERROR  'Startup Routine overlaps Subroutine Jump Table'
 .else
```

TL/DD/10131-33

```
 .if . < ADDRESS_limit & ORIGIN_check
      .ERROR  'Startup Routine not contiguous to Subroutine Jump Table'
 .endif
.endif
```

TL/DD/10131-34

```
        .Title  Bank_Switch, 'Bank Switch Function for Function Calls'
;       *********************************************************
;       *                                                       *
;       *       National Semiconductor MicroController Group    *
;       *                                                       *
;       *       HPC Code to Support Inter-Bank Function Calls   *
;       *       BANKSWIT.ASM -  Bank switch support functions   *
;       *********************************************************
;

;Copyright (c) 1988, National Semiconductor, Santa Clara Ca 95051

;Edit History
; 3/10/88 DKL   Create for Memo/Apps note
; 3/15/88 DKL   Add direct support for
;               C function names, assembler special
;

;This is the main switching function to allow inter-bank function calls
;transparent to the compiler and assembler.

;Requires compilation with the value SWITCH_STACK_DEPTH defined, for the
;number of levels of inter-bank function call nesting to be allowed.  The
;value should take into account any interrupt nesting from any interrupt
;service routines which may switch banks.

;Is called with stack as
;       SP -----> Next free location
;       SP-2 ---> Intermediate Switch Function Return Address
;       SP-4 ---> Destination's Return Address
;       SP-6 ---> Destination's Argument 1
;       ...       Destination's Argument Space
;       old sp -> Destination's Argument n
;       ...       Caller's Local Variable Space
;       FP -----> Caller's First Local Variable
;       FP-2 ---> Caller's Parent's Frame Pointer
;       FP-4 ---> Caller's Return Address
;       FP-6 ---> Caller's Argument 1
;       ...       Caller's Argument Space
;and must call Destination Function with stack in same form, but the
;Destination's Return Address must cause return to the switcher function.

;An additional stack is necessary to store the additional information so
;the main stack is not polluted.  This also requires an additional stack
;pointer.

        .form
 .macro  switch_to      function, bank, address
        .public _^function
_^function:
        jsr     function_call_switcher
 .if @ > 1
        .byte   low(address)
        .byte   high(address)
        .byte   bank
```

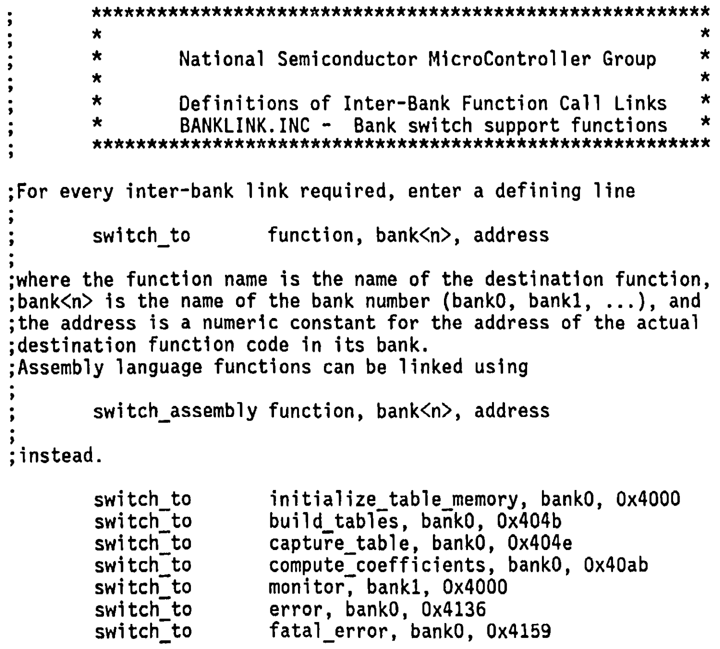
TL/DD/10131-35

```
 .else
        .byte   0,0,0           ;temporary place holders
 .endif
 .endm   ;switch_to

 .macro  switch_assembly function, bank, address
        .public function
function:
        jsr     function_call_switcher
 .if @ > 1
        .byte   low(address)
        .byte   high(address)
        .byte   bank
 .else
        .byte   0,0,0           ;temporary place holders
 .endif
 .endm   ;switch_assembly

        .form
;Bank Switching Control Port
bank_switch_port= 0x00e3:b   ;must not touch low byte of Port B

;Values for Bank Switching Control Port
bank0   = 0x00
bank1   = 0x01
bank2   = 0x02
bank3   = 0x03
bank4   = 0x20
bank5   = 0x21
bank6   = 0x22
bank7   = 0x23
bank8   = 0x40
bank9   = 0x41
bank10  = 0x42
bank11  = 0x43
bank12  = 0x60
bank13  = 0x61
bank14  = 0x62
bank15  = 0x63

;Switch stack
        .sect   switch_stack, ram16, rel
        .dsw    SWITCH_STACK_DEPTH * 2
growth  = 4
        .endsect

;Switch stack pointer
        .sect   switch_pointer, base, rel
switch_stack_pointer:   .dsw    1
        .endsect

;Initialization value for switch stack pointer
        .sect   switch_init, rom8, rel
        .byte   low(e_sect(switch_stack))
        .byte   high(e_sect(switch_stack))
```

TL/DD/10131-36

```
        .endsect

;Initialization control for switch stack pointer
        .sect   _init_info_, rom16, rel
        .word   b_sect(switch_pointer)
        .word   e_sect(switch_pointer) -1
        .word   b_sect(switch_init)
        .endsect

        .sect   switch_code, rom8, rel
;Linkages
        .incld  banklink.inc

;Switch from caller's bank to destination bank, transparently
;All registers must be preserved
;
function_call_switcher:
        push    a               ;free up registers
        push    x
        add     switch_stack_pointer,#-growth ;get switch stack room
        ld      x,switch_stack_pointer
        ld      a,bank_switch_port ;put caller bank on switch stack
        x       a,[x+].w
        ld      a,-8[sp].w      ;put caller return on switch stack
        x       a,[x+].w
        ld      a,-6[sp].w      ;access destination information
        st      a,x
        ld      a,[x+].b        ;get destination address onto stack
        st      a,-6[sp].b
        ld      a,[x+].b        ;(as bytes because no alignment)
        st      a,-5[sp].b
        ld      a,[x+].b        ;put destination bank in port
        st      a,bank_switch_port
        ld      a,#function_call_returner ;put switcher return on stack
        st      a,-8[sp].w
        pop     x
        pop     a
        ret                     ;transfer to destination in new bank
;
;Return to caller's bank from destination bank, transparently
;All registers must be preserved
;
function_call_returner:
        push    a               ;space for return address
        push    a               ;free up register
        ld      a,[switch_stack_pointer].w ;restore caller bank
        st      a,bank_switch_port
        ld      a,2[switch_stack_pointer].w ;restore caller return
        st      a,-4[sp].w
        add     switch_stack_pointer,#growth ;give up switch stack room
        pop     a
        ret                     ;return to caller in original bank
;
        .endsect
        .end
```

TL/DD/10131-37



```
;       *********************************************************
;       *                                                       *
;       *       National Semiconductor MicroController Group    *
;       *                                                       *
;       *       Definitions of Inter-Bank Function Call Links   *
;       *       BANKLINK.INC -  Bank switch support functions   *
;       *********************************************************
;

;For every inter-bank link required, enter a defining line
;
;       switch_to       function, bank<n>, address
;
;where the function name is the name of the destination function,
;bank<n> is the name of the bank number (bank0, bank1, ...), and
;the address is a numeric constant for the address of the actual
;destination function code in its bank.
;Assembly language functions can be linked using
;
;       switch_assembly function, bank<n>, address
;
;instead.

        switch_to       initialize_table_memory, bank0, 0x4000
        switch_to       build_tables, bank0, 0x404b
        switch_to       capture_table, bank0, 0x404e
        switch_to       compute_coefficients, bank0, 0x40ab
        switch_to       monitor, bank1, 0x4000
        switch_to       error, bank0, 0x4136
        switch_to       fatal_error, bank0, 0x4159
```

TL/DD/10131-38

```
;       *******************************************************
;       *                                                     *
;       *       National Semiconductor MicroController Group  *
;       *                                                     *
;       *       Macros to Assist Bank Switching Linkages      *
;       *       BANKDEFS.INC -  Bankswitch support functions  *
;       *******************************************************

;For every inter-bank link into a module, substitute definitions
;are needed using the values of the inter-bank link in shared
;memory.  These macros make it easier.
;
;       link_address    function, address
;       link_assembly   function, address
;
;where function is the name of the linked function and address is the
;address of the link code in the shared bank.

 .macro  link_address    function, address
        .public _~function
_~function = address
 .endm

 .macro  link_assembly   function, address
        .public function
function = address
 .endm

;For forcing a library routine to be linked, even though not accessed.
;
;       force_library   routine, routine, routine, ...
;       force_assembly  routine, routine, routine, ...
;
;Multiple lines may be used.

 .macro  force_library   list
 .set $count, 0
 .do @
 .set $count, $count + 1
        .extrn  _~@$count
 .enddo
 .endm   ;force_library

 .macro  force_assembly  list
 .set $count, 0
 .do @
 .set $count, $count + 1
        .extrn  @$count
 .enddo
 .endm   ;force_assembly

;To create the dummy place holders for the initialization information
;sections.
```

TL/DD/10131-39

```
;
;       init_dummy      size, size, size, ...
;
;Multiple lines may be used.

 .macro  init_dummy      list
        .sect   _init_info_, rom16, rel
 .set $count, 0
 .do @
 .set $count, $count + 1
        .dsb    @$count
 .enddo
        .endsect
 .endm   ;init_dummy
```

TL/DD/10131-40

```
/*
        tables.c        Placed in Bank0.
*/

#include "tables.h"

extern int coefficients[10];
extern struct table_entry live;

#define table_memory        (* ((struct table_entry *) 0x6000))
#define table_memory_end    (* ((struct table_entry *) 0x8000))

static int table_entries, table_values;
static struct table_entry * first_table;

/* this initializes special RAM memory in the bank for tables */

NOLOCAL
initialize_table_memory()
{
    static struct table_entry * p;

    /* initialize memory as an array of structure */
    for( p = &table_memory, table_entries = 0;
        p < &table_memory_end;
        p++, table_entries++ )
    {
        p->spins = 0;
        p->rolls = 0;
        p->result = 0;
    }
    /* record initial state */
    first_table = &table_memory;
    table_values = 0;
}

/* builds a series of table entries in the RAM memory from inputs */

NOLOCAL
build_tables()
{
    /*
    ...
    decide when table is ready
    ...
    */
    capture_table();
}

NOLOCAL
capture_table()
{
    static struct table_entry * next;

    if( table_values < table_entries )
```

TL/DD/10131–42

```
    {
        /* table not full, locate next and add one */
        next = first_table + table_values;
    }
    else
    {
        /* table full, advance one as ring */
        next = first_table;
        if( ++first_table >= &table_memory_end )
        {
            first_table = &table_memory;
        }
    }
    *next = live;
}

/* data reduction on table */

NOLOCAL
compute_coefficients()
{
    static int i;
    static struct table_entry * p;

    for( i = 0, p = first_table; i < table_values; i++ )
    {
        /*
        ...
        code to do data reduction on available data
        ...
        */
            recursive_spin_reduction(p, 0);
        if( ++p >= &table_memory_end )
        {
            p = &table_memory;
        }
    }
}

/* reduction on each entry */

static
recursive_spin_reduction(entry, item)
struct table_entry * entry;
int item;
{
    /* ... */
    if( item < entry->spins )
    {
        recursive_spin_reduction(entry, item + 1);
        /* ... */
    }
    /* ... */
}
```

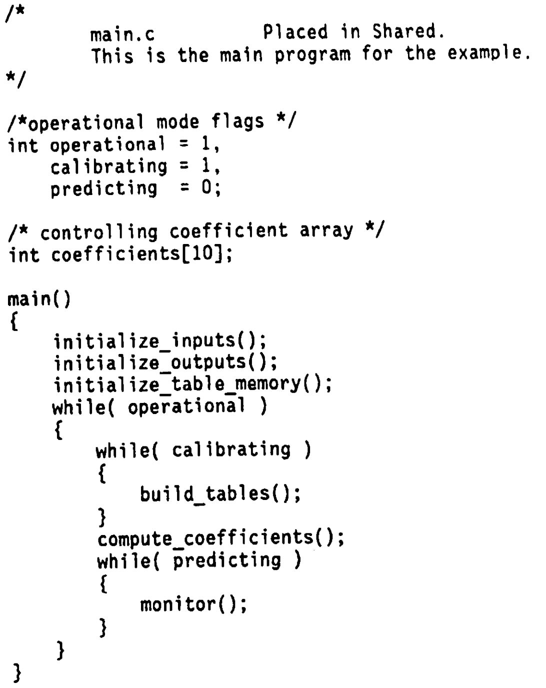
TL/DD/10131-43

```
/*
        errors.c        Placed in Bank0.
*/

static int error_count = 0;

NOLOCAL
error(code)
int code;
{
    printf("Error number %i - continuing\n", code);
    error_count++;
}

NOLOCAL
fatal_error(code)
int code;
{
    static int i;

    for( i = 0; i < 15; i++ )
    {
        putchar(0x07);
    }
    printf("\n\nFATAL ERROR number %i - ABORTING PROCESSING\n\n",
        code);
    quit();
}

NOLOCAL
quit()
{
    printf("Program terminated.  %i recoverable errors\n",
        error_count);
}
```

TL/DD/10131-44

```
/*
        monitor.c       Placed in Bank1.
*/

#include "tables.h"

extern struct table_entry live;

NOLOCAL
monitor()
{
    static int predictable;

    /*
    ...
    system monitoring
    ...
    */
    while( live.spins < 3
        || live.rolls < 5 ) ;
    while( !live.result )
    {
        compute_prediction();
    }
    validate_calculation();
    capture_table();
}

compute_prediction()
{
    int i;

    /*
    ...
    complex calculations to give a SWAG
    ...
    */
    printf("Prediction: %i\n", i);
}

validate_calculation()
{
    int i, j, k;

    /*
    ...
    match latest result to what we would predict
    ...
    */
    printf("Final prediction: %i, actual: %i, accuracy: %i\n", i, j, k);
    if( k < 10 )
    {
        error(1);
    }
}
```

TL/DD/10131-45



```
/*
        main.c          Placed in Shared.
        This is the main program for the example.
*/

/*operational mode flags */
int operational = 1,
    calibrating = 1,
    predicting  = 0;

/* controlling coefficient array */
int coefficients[10];

main()
{
    initialize_inputs();
    initialize_outputs();
    initialize_table_memory();
    while( operational )
    {
        while( calibrating )
        {
            build_tables();
        }
        compute_coefficients();
        while( predicting )
        {
            monitor();
        }
    }
}
```

TL/DD/10131-46

AN-579

MICROWIRE/PLUS™ Serial Interface for COP800 Family

National Semiconductor
Application Note 579
Ramesh Sivakolundu
Sunder Velamuri

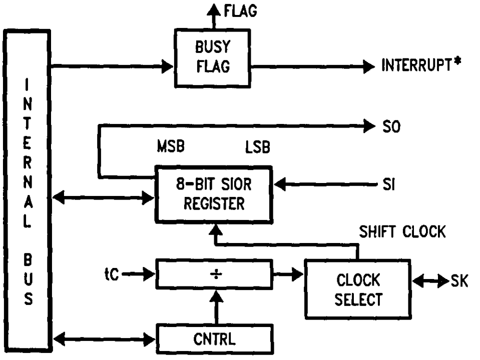
INTRODUCTION

National Semiconductor's COP800 family of full-feature, cost-effective microcontrollers use a new 8-bit single chip core architecture fabricated with M²CMOS process technology. These high performance microcontrollers provide efficient system solutions with a versatile instruction set and high functionality.

The COP800 family of microcontrollers feature the MICROWIRE/PLUS mode of serial communication. MICROWIRE/PLUS is an enhancement of the MICROWIRE™ synchronous serial communications scheme, originally implemented on the COP400 family of microcontrollers. The MICROWIRE/PLUS interface on the COP800 family of microcontrollers enables easy I/O expansion and interfacing to several COPS peripheral devices (A/D converters, EEPROMs, Display drivers etc.), and interfacing with other microcontrollers which support MICROWIRE/PLUS or SPI* modes of serial interface.

MICROWIRE/PLUS DEFINITION

MICROWIRE/PLUS is a versatile three wire, SI (serial input), SO (serial output), and SK (serial clock), bidirectional serial synchronous communication scheme where the COP800 is either the Master providing the Shift Clock (SK) or a slave accepting an external Shift Clock (SK). The COP800 MICROWIRE/PLUS system block diagram is shown in *Figure 1*. The MICROWIRE/PLUS serial interface utilizes an 8-bit memory mapped MICROWIRE/PLUS serial shift register, SIOR, clocked by the SK signal. As the name suggests, the SIOR register serves as the shift register for serial transfers. SI, the serial input line to the COP800 microcontroller, is the shift register input. SO, the shift register output, is the serial output to external devices. SK is the serial synchronous clock. Data is clocked into and out of the peripheral devices with the SK clock. The SO, SK and SI are mapped as alternate functions on pins 4, 5, and 6 respectively of the 8-bit bidirectional G Port.



TL/DD/10252-1

*only in COP888XX series

FIGURE 1. MICROWIRE/PLUS Block Diagram

MICROWIRE/PLUS OPERATION

In MICROWIRE/PLUS serial interface, the input data on the SI pin is shifted high order first into the Least Significant Bit (LSB) of the 8-bit SIOR shift register. The output data is shifted out high order first from the Most Significant Bit (MSB) of the shift register onto the SO pin. The SIOR register is clocked on the falling edge of the SK clock signal. The input data on the SI pin is shifted into the LSB of the SIOR register on the rising edge of the SK clock. The MSB of the SIOR register is shifted out to the SO pin on the falling edge of the SK clock signal. The SK clock signal is generated internally by the COP800 for the master mode of MICROWIRE/PLUS operation. In the slave mode, the SK clock is generated by an external device (which acts as the master) and is input to the COP800.

The MSEL (MICROWIRE Select) flag in the CNTRL register is used to enable MICROWIRE/PLUS operation. Setting the MSEL flag enables the gating of the MICROWIRE/PLUS interface signals through the G port. Pins G4, G5, and G6 of the G port are used for the signals SO, SK and SI, respectively. It should be noted that the G port configuration register must be set up appropriately for MICROWIRE/PLUS operation. Table I illustrates the G-port configurations. In the master mode of MICROWIRE/PLUS operation, G4 and G5 need to be selected as outputs for SO and SK signals. Alternatively, in the slave mode of operation, G5 needs to be configured as an input for the external SK. The SI signal is a dedicated input on G6 and therefore no further setup is required.

TABLE I. G Port Configurations

G4 (SO) Config. Bit	G5 (SK) Config Bit.	G4 Fun.	G5 Fun.	Operation
1	1	SO	Int. SK	MICROWIRE Master
0	1	TRI-STATE	Int. SK	MICROWIRE Master
1	0	SO	Ext. SK	MICROWIRE Slave
0	0	TRI-STATE	Ext. SK	MICROWIRE Slave

The SL1 and SL0 (S1 and S0 in COP820C and COP840C) bits of the CNTRL register are used to select the clock division factor (2, 4, or 8) for SK clock generation in MICROWIRE/PLUS master mode operation. A clock select table for these bits of the CNTRL register along with the CNTRL register is shown in Table II. The counter associated with the master mode clock division factor is cleared when the MICROWIRE/PLUS BUSY flag is low. The clock division factor is relative to the instruction cycle frequency. For example, if the COP800 is operating with an internal clock of 1 MHz, the SK clock rate would be 500 kHz, 250 kHz, or 125 kHz for SL1 and SL0 values of 00, 01 and 10 (or 11) respectively.

TABLE II

CNTRL Register (Address X'00EE)

The Timer1 (T1) and MICROWIRE control register contains the following bits:

SL1 & SL0 Select the MICROWIRE clock divide by (00 = 2, 01 = 4, 1X = 8)

IEDG External Interrupt Edge Polarity Select (0 = Rising Edge, 1 = Falling Edge)

MSEL Selects G5 and G4 as MICROWIRE Signals SK and SO Respectively

T1C0 Timer T1 Start/Stop Control in Timer Modes 1 and 2
Timer T1 Underflow Interrupt Pending Flag in Timer Mode 3

T1C1 Timer T1 Mode Control Bit

T1C2 Timer T1 Mode Control Bit

T1C3 Timer T1 Mode Control Bit

T1C3	T1C2	T1C1	T1C0	MSEL	IEDG	SL1	SL0
Bit 7							Bit 0

SL1	SL0	SK
0	0	$2 \times t_C$
0	1	$4 \times t_C$
1	x	$8 \times t_C$

Where t_C is the instruction cycle clock

MICROWIRE/PLUS MASTER MODE OPERATION

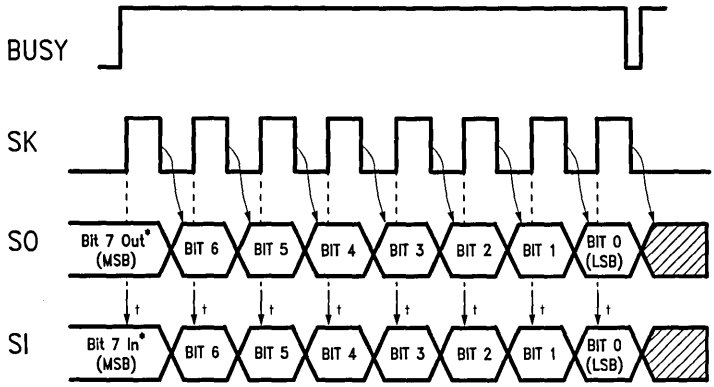
In the MICROWIRE/PLUS master mode, the BUSY flag of PSW (Processor Status Word) is used to control the shifting of the MICROWIRE/PLUS 8-bit shift register. Setting the BUSY flag causes the SIOR register to shift out 8 bits of data from SO at the high order end of the shift register. During the same time, 8 new bits of data from SI are shifted into the low order end of the SIOR register. The BUSY flag is automatically reset after the 8 bits of data have been shifted *(Figure 2)*. The COP888XX series of microcontrollers provide a vectored maskable interrupt when the BUSY goes low indicating the end of an 8-bit shift. Input data is clocked into the SIOR register from the SI pin with the rising edge of the SK clock, while the MSB of the SIOR is shifted onto the SO pin with the falling edge of the SK clock. The user may reset the BUSY bit by software to allow less than 8 bits to shift. However, the user should ensure that the software BUSY resets only occurs when the SK clock is low, in order to avoid a narrow SK terminal clock.

MICROWIRE/PLUS SLAVE MODE OPERATION

In the MICROWIRE/PLUS Slave mode of operation the SK clock is generated by an external source. Setting the MSEL bit in the CNTRL register enables the SO and SK functions onto the G Port. The SK pin must be configured as an input and the SO pin configured as an output by resetting and setting the appropriate bits in the Port G configuration register. The user must set the BUSY flag immediately upon entering the Slave mode. After eight clock pulses the Busy flag will be cleared and the sequence may be repeated. However, in the Slave mode the COP888 series does not shift data if the BUSY flag is reset, whereas the COP820C and COP840C continues to shift regardless of the BUSY flag, if the SK clock is active.

MICROWIRE/PLUS ALTERNATE SK MODE

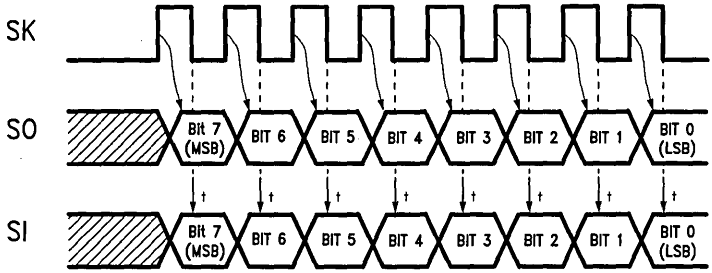
The COP888XX series of microcontrollers also allow an additional Alternate SK Phase Operation. In the normal mode data is shifted in on the rising edge of the SK clock and data is shifted out on the falling edge of the SK clock *(Figure 2)*. The SIOR register is shifted on each falling edge of the SK clock. In the alternate SK phase operation, data is shifted in on the falling edge of the SK clock and data is shifted out on the rising edge of the SK clock *(Figure 3)*.



*This bit becomes valid immediately after loading the SIOR register of the transmitting device.

†Arrows indicate points at which SI is sampled.

FIGURE 2. MICROWIRE/PLUS Timing



TL/DD/10252-3

↑ Arrows indicates points at which SI is sampled.

FIGURE 3. Alternate Phase SK Clock Timing

A control flag, SKSEL, allows either the normal SK clock or alternate SK clock to be selected. Resetting SKSEL selects the normal SK clock and setting SKSEL selects the alternate SK clock for the MICROWIRE/PLUS logic. The SKSEL flag is mapped into the G6 configuration bit. The SKSEL flag is reset after power up, selecting the normal SK clock signal. The alternate mode facilitates the usage of the MICROWIRE/PLUS protocol for serial data transfer between peripheral devices which are not compatible with the normal SK clock operation, i.e., shifting data out on the falling edge of the SK clock and shifting in data on the rising edge of the SK clock.

MICROWIRE/PLUS SAMPLE PROTOCOL

This section gives a sample MICROWIRE/PLUS protocol using a COP888CL and COP840C. The slave mode operating procedure for this sample protocol is explained, and a timing illustration of the protocol is provided.

1. The MSEL bit in the CNTRL register is set to enable MICROWIRE; G0 ($\overline{CS}$) and G5 (SK) are configured as inputs and G4 (SO) as an output. G6 (SI) is always an input.
2. Chip Select line ($\overline{CS}$) from master device is connected to G0 of the slave device. An active-low level on $\overline{CS}$ line causes the slave to interrupt.
3. From the high-to-low transistion on the $\overline{CS}$ line, there is no data transfer on the MICROWIRE until time "T" (See *Figure 4*).
4. The master initiates data transfer on the MICROWIRE by turning on the SK clock.
5. A series of data transfers take place between the master and slave devices.
6. The master pulls the $\overline{CS}$ line high to end the MICROWIRE operation. The slave device returns to normal mode of operation.

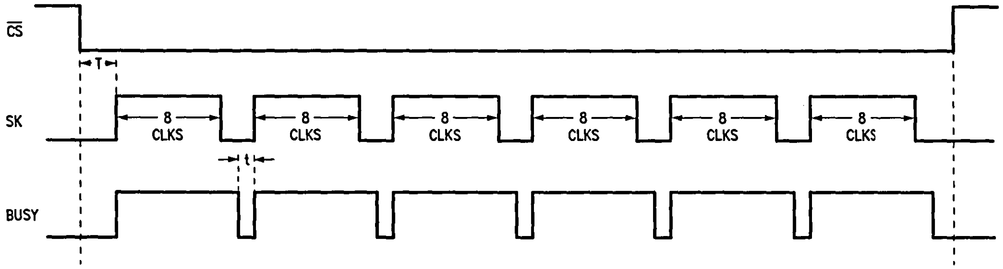
SLAVE MODE OPERATING PROCEDURE

1. The MSEL bit in the CNTRL register is set to enable MICROWIRE; G0 ($\overline{CS}$) and G5 (SK) are configured as inputs and G4 (SO) as an output. G6 (SI) is always an input.
2. Normal mode of operation until interrupted by $\overline{CS}$ going low.
3. Set the BUSY flag and load SIOR register with the data to be sent out on SO. (The shift register shifts 8 bits of data from SO at the high order end of the shift register. During the same time, 8 new bits of data from SI are loaded into the low order end of the shift register.)
4. Wait for the BUSY flag to reset. (The BUSY flag is automatically reset after 8 bits of data have been shifted).
5. If data is being read in, the user should save contents of the SIOR register.
6. The prearranged set of data transfers are performed.
7. Repeat steps 3 through 6. The user must ensure steps 3 through 6 are performed in time "t" (See *Figure 4*) as agreed upon in the protocol.

DIFFERENCES BETWEEN COP888 AND COP820/COP840

The COP888 series MICROWIRE/PLUS feature differs from that of the COP820/COP840 in some respects. The COP888 series can be configured to interrupt the processor after the completion of a MICROWIRE/PLUS operation indicated by the BUSY flag going low. The COP888 series supports a vectored interrupt scheme. Two bytes of program memory space are reserved for each interrupt source. The user would do any required context switching and then program a VIS (Vector Interrupt Select) instruction in order to branch to the interrupt service routine of the highest priority interrupt enabled and pending at the time of the VIS instruction. The addresses of the different interrupt service routines are chosen by the user and stored in ROM in a table starting at 0yE0 where "y" depends on the 256 byte block (0y00 to 0yFF) in which the VIS instruction is located. The vector address for the MICROWIRE/PLUS interrupt is 0yF2-0yF3.

Secondly, the COP888 series supports the alternate SK phase mode of MICROWIRE/PLUS operation. This feature facilitates the usage of the MICROWIRE/PLUS protocol for serial data transfer between peripheral devices which are not compatible with the normal SK clock operation, i.e., shifting data out on the falling edge of SK clock and shifting in data on the rising edge of the SK clock.



TL/DD/10252–4

FIGURE 4. MICROWIRE/PLUS Sample Protocol Timing Diagram

INTERFACE CONSIDERATIONS

To preserve the integrity of data exchange using MICROWIRE/PLUS, two aspects have to be considered:

1. Serial data exchange timing.
2. Fan-out/fan-in requirements.

Theoretically, infinite devices can access the same interface and be uniquely enabled sequentially in time. In practice, however, the actual number of devices that can access the same serial interface depends on the following: System data transfer rate, system supply requirement, capacitive loading on SK and SO outputs, the fan-in requirements of the logic families or discrete devices to be interfaced.

HARDWARE INTERFACE

For proper data transfer to occur the output should be able to switch between a HIGH level and a LOW level in a predetermined amount of time. The transfer is strictly synchronous and the timing is related to the MICROWIRE/PLUS system clock (SK). For example, if a COPS controller outputs a value at the falling edge of the clock and is latched in by the peripheral device at the rising edge, then the following relationship has to be satisifed:

$$t_{DELAY} + t_{SETUP} \leq t_{CK}$$

where t_{CK} is the time from data output starts to switch to data being latched into the peripheral chip, t_{SETUP} is the setup time for the peripheral device where the data has to be at a valid level, and t_{DELAY} is the time for the output to read the valid level. t_{CK} is related to the system clock provided by the SK pin of the COPS controller and can be increased by increasing the COPS instruction cycle time.

Besides the timing requirements, system supply and fan-out/fan-in requirements also have to be considered when interfacing with MICROWIRE/PLUS. To drive multi-devices on the same MICROWIRE/PLUS, the output drivers of the controller need to source and sink the total maximum leakage current of all the inputs connected to it and keep the signal level within the valid logic "1" and "0" input voltage levels. Thus, if devices of different types are connected to the same serial interface, output driver of the controller must satisfy all the input requirements of each device. Similarly, devices with TRI-STATE® outputs, when connected to the SI input, must satisfy the minimum valid input level of the controller and the maximum TRI-STATE® leakage current of all outputs.

So, for devices that have incompatible input levels or source/sink requirements, external pull-up resistors or buffers are necessary to provide level-shifting or driving.

TABLE III

Features		Part Number							
		DS890XX	MM545X	COP470	COP472	ADC83X (COP430)	COP498/499	COP452L	NMC9306 (COP494)
GENERAL									
Chip Function		AM/PM PLL	LED Display Driver	VF Display Driver	LCD Display Driver	A/D	RAM & Timer	Frequency Generator	E^{2}PROM
Process		ECL	NMOS	PMOS	CMOS	CMOS	CMOS	NMOS	NMOS
V_{CC} Range		4.75V–5.25V	4.5V–11V	−9.5V to −4.5V	3.0V–5.5V	4.5V–0.3V	2.4V–5.5V	4.5V–6.3V	4.5V–5.5V
Pinout		20	40	20	20	8/14/20	14/8	14	14
HARDWARE INTERFACE									
Min V_{IH}/Max V_{IL}		2.1V/0.7V	2.2V/0.8V	−1.5V/−4.0V	0.7 V_{CC}/0.8V	2.0V/0.8V	0.8 V_{CC}/0.4 V_{CC}	2.0V/0.8V	2.0V/0.8V
SK Clock Range		0–625 kHz	0–500 kHz	0–250 kHz	4–250 kHz	10–200 kHz	4–250 kHz	25–250 kHz	0–250 kHz
Write Data DI	Setup Min	0.3 μs	0.3 μs	1.0 μs	1.0 μs	0.2 μs	0.4 μs	800 ns	0.4 μs
	Hold Min	0.8 μs	(Note 3)	50 ns	100 ns (Note 1)	0.2 μs	0.4 μs	1.0 μs	0.4 μs
Read Data Prop Delay		(Note 4)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	2 μs (Note 2)	1 μs (Note 2)	2.0 μs
Chip Enable	Setup	0.275 μs	0.4 μs	1.0 μs Min	1 μs (Note 1)	0.2 μs	0.2 μs (Note 1)	(Note 3)	0.2 μs
	HOLD	0.300 μs	(Note 3)	1.0 μs Min	1 μs (Note 2)	0.2 μs	0 (Note 2)	(Note 3)	0
Max Frequency Range	AM	8 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
	FM	120 MHz	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)	(Note 3)
Max Osc. Freq.		(Note 3)	(Note 3)	250 kHz	(Note 3)	(Note 3)	2.1 MHz (−21) 32 kHz (−15)	256–2100 kHz (−4) 64–525 kHz (−2)	(Note 3)
SOFT									
Serial I/O Protocol		11D1–D20	1D1–D35	8 Bits At a Time	b1–b40	1xxx	1yyxxD6–D0 Start Bit	1yxxxx	1AA–DD
Instruction/ Address Word		None	None	None	None	(Note 4)	(Note 4)	(Note 4)	(Note 4)

Note 1: Reference to SK rising edge.

Note 2: Reference to SK falling edge.

Note 3: Not defined.

Note 4: See data sheet for different modes of operation.

TYPICAL APPLICATIONS

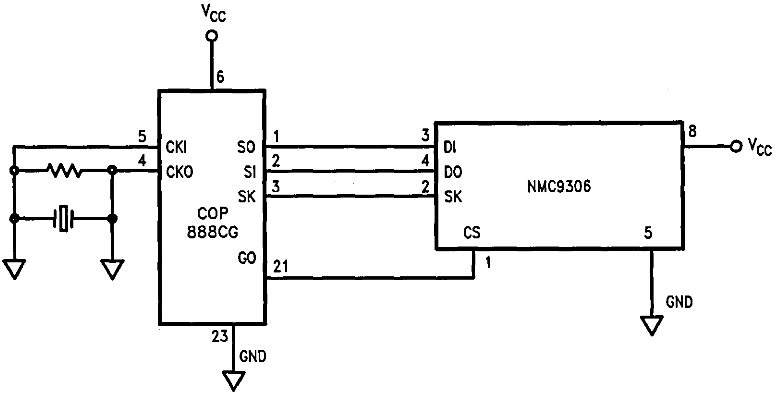
A whole family of off-the shelf devices exist that are directly compatible with MICROWIRE/PLUS protocol. This allows direct interface with the COP800 family of microcontrollers. Table III provides a summary of the existing devices, their function and specification.

NMC9306-COP888CG INTERFACE

The pin connection involved in interfacing an NMC9306 (COP494), a 256 bit E²PROM, with the COP888CG microcontroller is shown in *Figure 5*. Some notes on the NMC9306 interface requirements are:

1. The SK clock frequency should be in the 0 kHz–250 kHz range.
2. $\overline{CS}$ low period following an Erase/Write instruction must not exceed 30 ms maximum. It should be set at typical or minimum specification of 10 ms.
3. The start bit on DI must be set by a "0" to "1" transition following a $\overline{CS}$ enable ("0" to "1") when executing any instruction. One $\overline{CS}$ enable transition can only execute one instruction.
4. In the read mode, following an instruction and data train, the DI can be a "don't care", while the data is being outputted, i.e., for the next 17 bits or clocks. The same is true for other instructions after the instrution and data has been fed in.
5. The data out train starts with a dummy bit 0 and is terminated by chip deselect. Any extra SK cycle after 16 bits is not essential.

   If $\overline{CS}$ is held on after all 16 of the data bits have been outputed, the DO will output the state of DI until another $\overline{CS}$ LO to HI transition starts a new instruction cycle.
6. After a read cycle, the $\overline{CS}$ must be brought low for one SK clock cycle before another instruction cycle starts.



TL/DD/10252-5

FIGURE 5. NMC9306-COP888CG Interface

Instruction Set

Commands	Start Bit	Opcode	Address	Comments
READ	1	0000	A3A2A1A0	Read Register 0–15
WRITE	1	1000	A3A2A1A0	Write Register 0–15
ERASE	1	0100	A3A2A1A0	Erase Register 0–15
EWEN	1	1100	00 01	Write/Erase Enable
ENDS	1	1100	00 10	Write/Erase Disable
***WRAL	1	1100	01 00	Write All Registers
ERAL	1	1100	01 01	Read All Registers

Where A3A2A1A0 corresponds to one of the sixteen 16-bit registers.

All commands, data in, and data out are shifted in/out on the rising edge of the SK clock.

Write/Erase is then done by pulsing $\overline{CS}$ low for 10 ms.

All instructions are initiated by a LO-HI transition on $\overline{CS}$ followed by a LO-HI transition on DI.

READ— After read command is shifted in DI becomes don't care and data can be read out on data out, starting with dummy bit zero.

WRITE— Write command shifted in followed by data in (16 bits) the$\overline{CS}$ pulsed low for 10 ms minimum.

ERASE/ERASE ALL— Command shifted in followed by $\overline{CS}$ low.

WRITE ALL— Pulsing $\overline{CS}$ low for 10 ms.

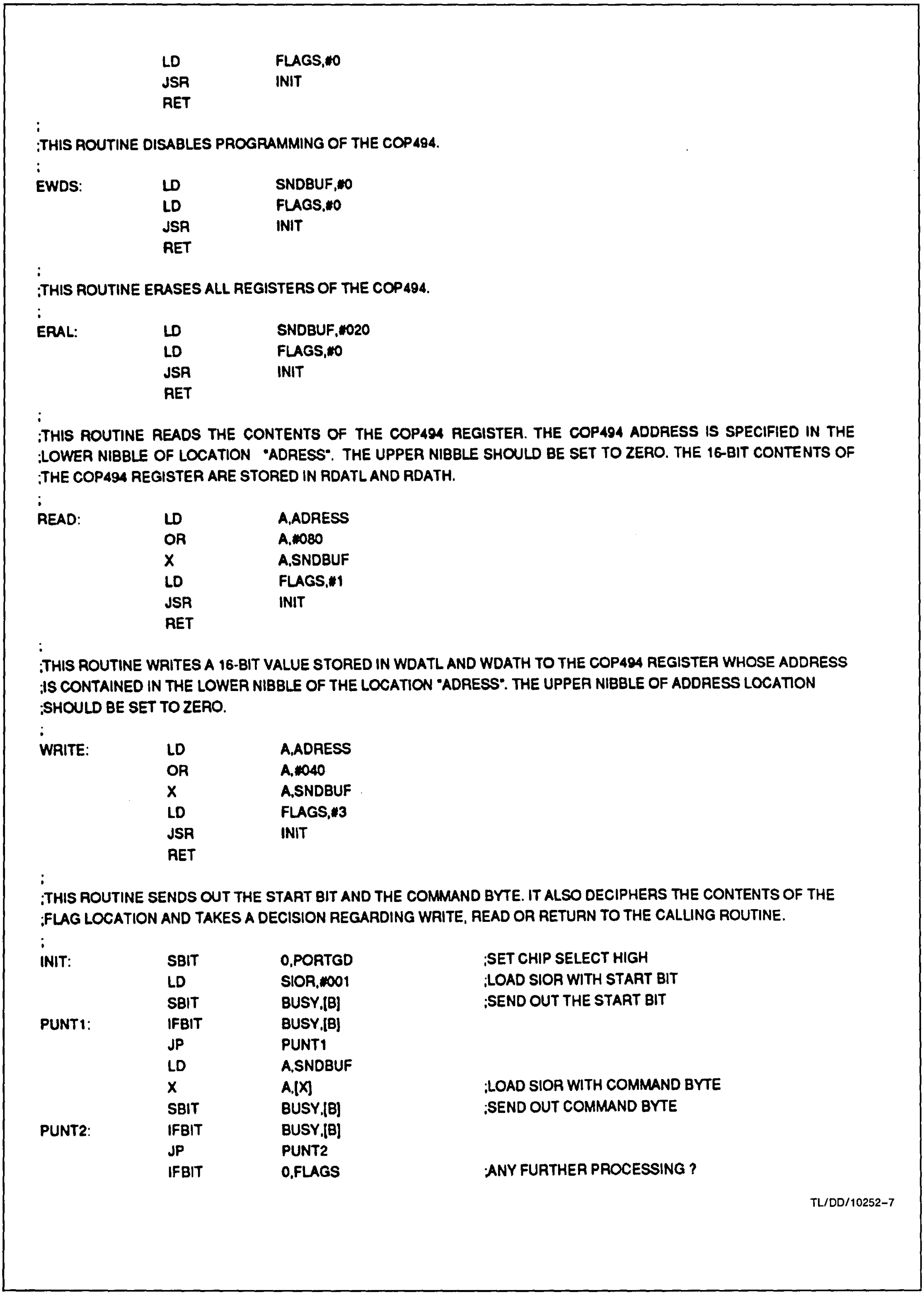
ENABLE/DISABLE— Command shifted in.

A detailed explanation of the E²PROM timing diagrams, instruction set and the various considerations could be found in the NMC9306 data sheet. A source listing of the software to interface the NMC9306 with the COP888CG is provided.

SOURCE LISTING

```
.INCLD COP888.INC
;
;This program provides in the form of subroutines, the ability to erase,enable, disable, read and write to the COP494 EEPROM.
;
;
SNDBUF = 0                     ;CONTAINS THE COMMAND BYTE TO BE WRITTEN TO COP494
RDATL  = 1                     ;LOWER BYTE OF THE COP494 REGISTER DATA READ
RDATH  = 2                     ;UPPER BYTE OF THE COP494 REGISTER DATA READ
WDATL  = 3                     ;LOWER BYTE OF THE DATA TO BE WRITTEN TO COP494
                               ;REGISTER
WDATH  = 4                     ;UPPER BYTE OF THE DATA TO BE WRITTEN TO COP494
                               ;REGISTER
ADRESS = 5                     ;THE LOWER 4-BITS OF THIS LOCATION CONTAIN THE
                               ;ADDRESS
                               ;OF THE COP494 REGISTER TO BE READ/WRITTEN
FLAGS  = 6                     ;USED FOR SETTING UP FLAGS
                               ;
                               ; FLAG VALUE     ACTION
                               ;-------         ----
                               ;  00        ERASE,ENABLE,DISABLE,ERASE ALL
                               ;  01        READ CONTENTS OF COP494 REGISTER
                               ;  03        WRITE TO COP494 REGISTER
                               ;  OTHERS       ILLEGAL COMBINATION
DLYH   = 0F0
DLYL   = 0F1
;
;THE INTERFACE BETWEEN THE COP888CG AND THE COP494 (256-BIT EEPROM) CONSISTS OF FOUR LINES. THE
;G0 (CHIP SELECT LINE), G4 (SERIAL OUT SO), G5 (SERIAL CLOCK SK) ;AND G6 (SERIAL IN SI).
;
;    INITIALIZATION
;
            LD          PORTGC,#031          ;Setup G0,G4,G5 as outputs
            LD          PORTGD,#00           ;Initialize G data reg to zero
            LD          CNTROL,#08           ;Enable MSEL, select MW rate of 2tc
            LD          B,#PSW
            LD          X,#SIOR
;
;THIS ROUTINE ERASES THE MEMORY LOCATION POINTED TO BY THE ADDRESS CONTAINED IN THE LOCATION
;"ADRESS". THE LOWER NIBBLE OF "ADRESS" CONTAINS THE COP494 REGISTER ADDRESS AND THE UPPER NIBBLE
;SHOULD BE SET TO ZERO.
;
ERASE:      LD          A,ADRESS
            OR          A,#0C0
            X           A,SNDBUF
            LD          FLAGS,#0
            JSR         INIT
            RET
;
;THIS ROUTINE ENABLES PROGRAMMING OF THE COP494. PROGRAMMING MUST BE PRECEDED ONCE BY A
;PROGRAMMING ENABLE (EWEN).
;
EWEN:       LD          SNDBUF,#030
```

TL/DD/10252-6



```
                LD          FLAGS,#0
                JSR         INIT
                RET
;
;THIS ROUTINE DISABLES PROGRAMMING OF THE COP494.
;
EWDS:           LD          SNDBUF,#0
                LD          FLAGS,#0
                JSR         INIT
                RET
;
;THIS ROUTINE ERASES ALL REGISTERS OF THE COP494.
;
ERAL:           LD          SNDBUF,#020
                LD          FLAGS,#0
                JSR         INIT
                RET
;
;THIS ROUTINE READS THE CONTENTS OF THE COP494 REGISTER. THE COP494 ADDRESS IS SPECIFIED IN THE
;LOWER NIBBLE OF LOCATION  "ADRESS". THE UPPER NIBBLE SHOULD BE SET TO ZERO. THE 16-BIT CONTENTS OF
;THE COP494 REGISTER ARE STORED IN RDATL AND RDATH.
;
READ:           LD          A,ADRESS
                OR          A,#080
                X           A,SNDBUF
                LD          FLAGS,#1
                JSR         INIT
                RET
;
;THIS ROUTINE WRITES A 16-BIT VALUE STORED IN WDATL AND WDATH TO THE COP494 REGISTER WHOSE ADDRESS
;IS CONTAINED IN THE LOWER NIBBLE OF THE LOCATION "ADRESS". THE UPPER NIBBLE OF ADDRESS LOCATION
;SHOULD BE SET TO ZERO.
;
WRITE:          LD          A,ADRESS
                OR          A,#040
                X           A,SNDBUF
                LD          FLAGS,#3
                JSR         INIT
                RET
;
;THIS ROUTINE SENDS OUT THE START BIT AND THE COMMAND BYTE. IT ALSO DECIPHERS THE CONTENTS OF THE
;FLAG LOCATION AND TAKES A DECISION REGARDING WRITE, READ OR RETURN TO THE CALLING ROUTINE.
;
INIT:           SBIT        0,PORTGD                ;SET CHIP SELECT HIGH
                LD          SIOR,#001               ;LOAD SIOR WITH START BIT
                SBIT        BUSY,[B]                ;SEND OUT THE START BIT
PUNT1:          IFBIT       BUSY,[B]
                JP          PUNT1
                LD          A,SNDBUF
                X           A,[X]                   ;LOAD SIOR WITH COMMAND BYTE
                SBIT        BUSY,[B]                ;SEND OUT COMMAND BYTE
PUNT2:          IFBIT       BUSY,[B]
                JP          PUNT2
                IFBIT       0,FLAGS                 ;ANY FURTHER PROCESSING ?
```

TL/DD/10252-7

```
            JP          NOTDON                  ;YES
            RBIT        0,PORTGD                ;NO, RESET CS AND RETURN
            RET
;
NOTDON:     IFBIT       1,FLAGS                 ;READ OR WRITE?
            JP          WR494                   ;JUMP TO WRITE ROUTINE
            LD          SIOR,#000               ;NO, READ COP494
            SBIT        BUSY,PSW                ;DUMMY CLOCK TO READ ZERO
            RBIT        BUSY,[B]
            SBIT        BUSY,[B]
PUNT3:      IFBIT       BUSY,[B]
            JP          PUNT3
            X           A,[X]
            SBIT        BUSY,[B]
            X           A,RDATH
PUNT4:      IFBIT       BUSY,[B]
            JP          PUNT4
            LD          A,[X]
            X           A,RDATL
            RBIT        0,PORTGD
            RET
;
WR494:      LD          A,WDATH
            X           A,[X]
            SBIT        BUSY,[B]
PUNT5:      IFBIT       BUSY,[B]
            JP          PUNT5
            LD          A,WDATL
            X           A,[X]
            SBIT        BUSY,[B]
PUNT6:      IFBIT       BUSY,[B]
            JP          PUNT6
            RBIT        0,PORTGD
            JSR         TOUT
            RET
;
;ROUTINE TO GENERATE DELAY FOR WRITE
;
TOUT:       LD          DLYH,#00A
WAIT:       LD          DLYL,#0FF
WAIT1:      DRSZ        DLYL
            JP          WAIT1
            DRSZ        DLYH
            JP          WAIT
            RET
            .END
```

TL/DD/10252–8

COP472-COP820 Interface

The pin connection required for interfacing COP472-3 Liquid Crystal Display (LCD) Controller with COP820C microcontroller is shown in *Figure 6.* The COP472-3 drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. One COP472-3 can drive 36 segments and two or more COP472-3's can be cascaded to drive additional segments as long as the output loading capacitance does not exceed specifications.

The COP472-3 requires 40 information bits: 36 data and 4 control. The function of each control bit is described briefly. Data is loaded in serially, in sets of eight bits. Each set of segment data is in the following format:

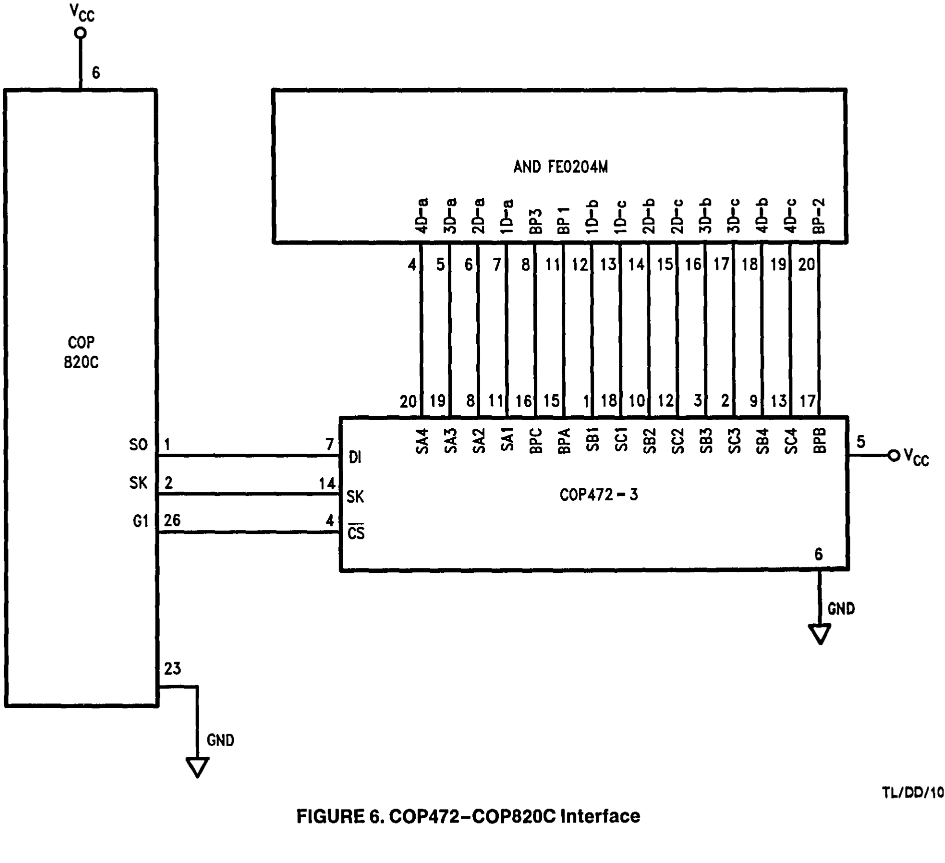
SA	SB	SC	SD	SE	SF	SG	SH

Data is shifted into an eight bit shift register. The first bit of data is for segment H, digit 1, and the eight bit is for segment A, digit 1. A set of eight bits are shifted in and then loaded into the digit one latches. The second, third, and fourth set is then loaded sequentially. The fifth set of data bits contain special segment data and control data in the following format:

SYNC	Q7	Q6	X	SP4	SP3	SP2	SP1

The first four bits shifted in contain the special character segment data. The fifth bit is not used. The sixth and seventh bits program the COP472-3 as a stand alone LCD driver or as a master or slave for cascading COP472-3's. The Table IV summarizes the function of bits six and seven.

The eight bit is used to synchronize two COP472-3's to drive an 8½ digit display. A detailed explanation of the various timing diagrams, loading sequence and segment/backplane multiplex scheme can be found in the data sheets of COP472-3. The source listing of the software used in the interface is provided.



TL/DD/10252-12

FIGURE 6. COP472-COP820C Interface

SOURCE LISTING

```
;THIS PROGRAM DISPLAYS FOUR DIGITS OF THE RAM SPECIFIED BY; THE ADDRESS POINTER "HEAD" ON A 4 DIGIT 3
;DECIMAL POINT (MULTIPLEXED) LCD DISPLAY. THE DATA STREAM IS SENT OUT SERIALLY THROUGH THE
;MICROWIRE/PLUS INTERFACE TO THE COP472 LCD DISPLAY DRIVER. NOTE: THE RAM CONTENTS SHOULD BE
;BETWEEN "0" AND "F".
;
              .TITLE       LCD
;
'
              .CHIP        820
;
;
              PORTGD       = 0D4                ;PORT G DATA REGISTER
              PORTGC       = 0D5                ;PORT G CONFIGURATION
;
;
              SIO          = 0E9                ;MICROWIRE SHIFT REGISTER
;
;
              PSW          = 0EF                ;PSW REGISTER
              CNTRL        = 0EE                ;CNTRL REGISTER
;
;
              CONTRL       = 04                 ;MEMORY LOCATION FOR THE
;                                               ;COP472 CONTROL WORD
              HEAD         = 00                 ;STARTING MEMORY LOC FOR
                                                ;DATA TO BE DISPLAYED
              MEMSTR       = 05                 ;STARTING MEMORY LOC FOR
                                                ;STORING SEGMENT DATA
              MEMEND       = 08                 ;MEMORY LOC FOR LAST
                                                ;SEGMENT DATA
;
;
START:        LD           CNTRL,#08            ;SET MSEL BIT IN CNTRL
              LD           PORTGC,#032          ;SET G5,G4& G1 AS OUTPUTS
              LD           CONTRL,#0FC          ;SET COP472 IN STAND ALONE MODE
;                                               ;
;                                               ;
;
;THIS ROUTINE  GETS THE SEGMENT DATA FOR RAM DIGITS POINTED BY B REGISTER AND STORES IN RAM MEMORY
;POINTED BY X REGISTER
;
;
AGAIN:        LD           B,#HEAD              ;POINTER TO START ADDRESS
              LD           X,#MEMSTR            ;POINTER TO STORE ADDRESS
NEXDIG:       LD           A,[B+]               ;LOAD A WITH RAM DIGIT AND
                                                ;INCREMENT B POINTER
              ADD          A,#0F0               ;ADD OFFSET TO THE DIGIT
              LAID                              ;LOOKUP SEGMENT DATA TO A
              X            A,[X+]               ;STORE IN MEMORY
              IFBNE        #04                  ;CHECK FOR END OF FOUR
                                                ;DIGITS AND REPEAT
              JP           NEXDIG               ;IF NECESSARY
;
;
; THIS ROUTINE DISPLAYS THE CONTENTS OF FOUR MEMORY LOCATION
; ON THE LCD DISPLAY.
;
;
DSP:          LD           B,#MEMEND            ;LOAD THE START ADDRESS
              RBIT         1,PORTGD             ;BIT G1 IS USED TO SELECT
                                                ;COP472 (PIN 4)
```

TL/DD/10252-9

TL/DD/10252-10

```
REPEAT:    LD        A,[B-]            ;SEGMENT DATA TO A
           X         A,SIO             ;LOAD THE SIO REGISTER
           SBIT      #2,PSW            ;SET BUSY BIT IN PSW
WAIT:      IFBIT     #2,PSW            ;WAIT TILL SHIFTING IS
           JP        WAIT              ;COMPLETE
           IFBNE     #04               ;CHECK FOR END OF FOUR
           JP        REPEAT            ;DIGITS AND REPEAT
           SBIT      1,PORTGD          ;DESELECT COP472
LOOP:      JP        LOOP              ;DONE DISPLAYING
;
;
; STORE THE LOOKUP TABLE FOR SEGMENT DATA IN ROM LOCATION 0F0
;
;
           .=0F0
;
           .BYTE     03F,006,05B,04F   ;DATA FOR 0,1,2,3
           .BYTE     066,06D,07D,07    ;DATA FOR 4,5,6,7
           .BYTE     07F,067,077,07C   ;DATA FOR 8,9,A,B
           .BYTE     039,05E,079,071   ;DATA FOR C,D,E,F
;
;
           .END
```

TL/DD/10252-11

The code listed in this App Note is available on Dial-A-Helper.

Dial-A-Helper is a service provided by the Microcontroller Applications Group. The Dial-A-Helper system provides access to an automated information storage and retrieval system that may be accessed over standard dial-up telephone lines 24 hours a day. The system capabilities include a MESSAGE SECTION (electronic mail) for communicating to and from the Microcontroller Applications Group and a FILE SECTION mode that can be used to search out and retrieve application data about NSC Microcontrollers. The minimum system requirement is a dumb terminal, 300 or 1200 baud modem, and a telephone.

With a communications package and a PC, the code detailed in this App Note can be downloaded from the FILE SECTION to disk for later use. The Dial-A-Helper telephone lines are:

Modem (408) 739-1162
Voice (408) 721-5582

For Additional Information, Please Contact Factory

High Performance Controller in Information Control Applications

National Semiconductor
Application Note 585
Steve McRobert

ABSTRACT

This paper describes National Semiconductor's HPC™ family of High Performance microControllers. Included are two examples showing how the devices are used in actual Information Control applications.

The architecture, technology, and instruction set of the HPC family are presented, with emphasis on how these features are appropriate for use in microcontroller based information control systems. Two example applications are given, the first being the use of a single chip mode HPC as an I/O processor and interrupt handler in a laser beam printer. In this case the HPC acts as a slave to the main 32-bit CPU in the printer, freeing it from the many tasks which require fast interrupt response and thus improves system throughput. The second example shows the HPC used in expanded mode as the sole microprocessor in an ESDI to SCSI bridge adapter card. The operations performed by the HPC in this application are used as an example of how the instruction set and addressing modes work together to achieve high throughput. The paper concludes with a brief discussion of the future of the HPC family of devices.

INTRODUCTION

The HPC (High Performance Controller) family of microcontrollers was designed by National Semiconductor as the first of a new generation of 16-bit CMOS microcontrollers.

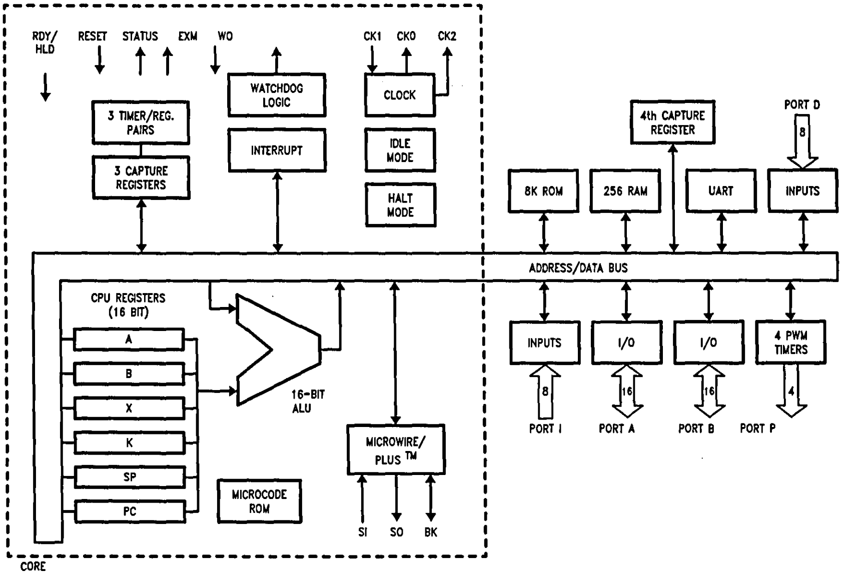
The intention was to start afresh, using the experience gained from earlier device families and, without software compatibility constraints, to create an architecture sufficiently advanced to be competitive for 10 years or more. Other design goals were to minimize device complexity, thus allowing for dependable, economical, high volume production, and to make HPC easy to understand so that system designers could readily convert designs to use the new family's advanced features.

These goals have been met, and, since the first device was sampled in early 1986, the HPC family has developed into a well proven solution to many design problems.

ARCHITECTURE

The HPC family is based on a core concept. All devices share a common core including the CPU and a base set of peripherals such as timer/counters etc. *Figure 1* shows a block diagram of the HPC16083 with the core emphasized at left. HPC uses a memory-mapped Von Neuman architecture, in which all registers, I/O ports, peripherals etc. are assigned memory locations in one uniform address space.

This includes the CPU registers *(Figure 1)*, allowing all HPC instructions to operate on every register in the programmer's model. Such uniformity simplifies the work of the assembly language programmer and the writer of the C compiler, making the HPC a particularly efficient microcontroller for running programs written in "C".



TL/DD/10346-1

FIGURE 1. HPC16083 Block Diagram

The core is connected to peripherals and on-chip memory by a 16-bit address/data bus, which is multiplexed to reduce die size. This bus is brought out on the A port when the device is used in expanded and/or ROMless modes, allowing off-chip devices to be accessed in exactly the same fashion as on-chip memory or peripherals.

When writing assembly language or C instructions the programmer perceives no difference between on-chip and off-chip memories, but both assembler and compiler take account of two key differences. When the HPC is run at high oscillator frequencies (up to 30 MHz on current production devices) a wait state must be applied for accesses to external memories or peripherals, but are never applied to on-chip RAM or registers. The other difference is that accesses to on-chip locations with addresses below 100 hexadecimal (called basepage accesses) require only a one byte address, so are thus shorter and faster than accesses to non-basepage locations *(Figure 2)*.

FFF:FFF0	INTERRUPT VECTORS	HPC16083 ON-CHIP ROM SPACE
FFEF:FFD0	JSRP VECTORS	
FFCF:E000	GENERAL PURPOSE ROM	
DFFF:0200	EXPANDED MODE ADDRESS SPACE	EXTERNAL USER MEMORY
0IFF:0IC0	ON-CHIP RAM	ON-CHIP RAM AND REGISTERS
0IBF:00C0	ON-CHIP REGISTERS	
00BF:0000	ON-CHIP RAM	

FIGURE 2

The programmer must choose which variables to put into on chip RAM or the basepage to achieve maximum performance and code efficiency.

Basepage RAM, because it is very fast and efficient to use, provides many of the benefits of the register file architecture used on some other microcontrollers. The HPC is different, however, in that it has a small set of registers: Accumulator, B pointer, X pointer and K (or limit) register. These registers all have addresses and can be used as general purpose memory locations, but are best used for their special functions. Many HPC instructions have two operands, the source and the destination. If the Accumulator (A) register is used as the destination, this is implied in the opcode and the address of A need not be included in the instruction, thus making it shorter and faster than instructions using another memory location as the destination. If the address of the source is contained in the B register then this too can be implied from the opcode and the whole instruction becomes one byte long.

Most HPC instructions thus have a single-byte form, using the B or X register as a pointer to the memory location being accessed.

The use of the K register will be discussed in the next section.

The primary objective when designing the architecture and instruction set of HPC was to minimize code size, an approach which can reduce throughput if unlimited bus bandwidth is available. In typical microcontroller applications the use of external memory is undesirable for board space and cost reasons. If the code is too large for mask ROM, the best solution in terms of space and cost is a single, relatively slow, EPROM.

In this situation of low bus bandwidth, the high byte efficiency of the HPC goes hand-in-hand with good performance.

ADDRESSING MODES

In keeping up with the HPC philosophy of being simple and quick to understand, the HPC instruction set *(Figure 3)* has relatively few mnemonics. This is because for those instructions with one or two addressable operands the same mnemonic is used regardless of the addressing mode, operand size (byte or word) or address size (depending upon whether each operand is in the basepage or not). Each individual memory location may be addressed using one of the following addressing modes:

Direct: The 8- or 16-bit address is included in the series of bytes that make up the instruction.

Indirect: The 8-bit address of a word in the base page is included in the instruction. The contents of this word are used as a pointer to the variable to be accessed.

Mnemonic	Description	Action	
ARITHMETIC INSTRUCTIONS			
ADD	Add	MA+Meml → MA	carry → C
ADC	Add with carry	MA+Meml+C → MA	carry → C
ADDS	Add short imm8	MA+imm8 → MA	carry → C
DADC	Decimal add with carry	MA+Meml+C → MA (Decimal)	carry → C
SUBC	Subtract with carry	MA−Meml+C → MA	carry → C
DSUBC	Decimal subtract w/carry	MA−Meml+C → MA (Decimal)	carry → C
MULT	Multiply (unsigned)	MA*Meml → MA & X, 0 → K, 0 → C	
DIV	Divide (unsigned)	MA/Meml → MA, rem. → X, 0 → K, 0 → C	
DIVD	Divide Double Word (unsigned)	(X & MA)/Meml → MA, rem → X, 0 → K, carry → C	
IFEQ	If equal	Compare MA & Meml, Do next if equal	
IFGT	If greater than	Compare MA & Meml, Do next if MA > Meml	
AND	Logical and	MA and Meml → MA	
OR	Logical or	MA or Meml → MA	
XOR	Logical exclusive-or	MA xor Meml → MA	
MEMORY MODIFY INSTRUCTIONS			
INC	Increment	Mem + 1 → Mem	
DECSZ	Decrement, skip if 0	Mem −1 → Mem, Skip next if Mem = 0	

FIGURE 3. HPC Instruction Set Description

Mnemonic	Description	Action
BIT INSTRUCTIONS		
SBIT	Set bit	1 → Mem.bit
RBIT	Reset bit	0 → Mem.bit
IFBIT	If bit	If Mem.bit is true, do next instr.
MEMORY TRANSFER INSTRUCTIONS		
LD	Load	MemI → MA
	Load, incr/decr X	Mem(X) → A, X ±1 (or 2) → X
ST	Store to Memory	A → Mem
X	Exchange	A ←→ Mem
	Exchange, incr/decr X	A ←→ Mem(X), X ±1 (or 2) → X
PUSH	Push Memory to Stack	W → W(SP), SP+2 → SP
POP	Pop Stack to Memory	SP−2 → SP, W(SP) → W
LDS	Load A, incr/decr B, Skip on condition	Mem(B) → A, B ±1 (or 2) → B, Skip next if B greater/less than K
XS	Exchange, incr/decr B, Skip on condition	Mem(B) ←→ A,B±1 (or 2) → B, Skip next if B greater/less than K
REGISTER LOAD IMMEDIATE INSTRUCTIONS		
LD B	Load B immediate	imm → B
LD K	Load K immediate	imm → K
LD X	Load X immediate	imm → X
LD BK	Load B and K immediate	imm → B,imm → K
ACCUMULATOR AND C INSTRUCTIONS		
CLR A	Clear A	0 → A
INC A	Increment A	A + 1 → A
DEC A	Decrement A	A − 1 → A
COMP A	Complement A	1's complement of A → A
SWAP A	Swap nibbles of A	A15:12 ← A11:8 ← A7:4 ←→ A3:0
RRC A	Rotate A right thru C	C → A15 → ... → A0 → C
RLC A	Rotate A left thru C	C ← A15 ← ... ← A0 ← C
SHR A	Shift A right	0 → A15 → ... → A0 → C
SHL A	Shift A left	C ← A15 ← ... ← A0 ← 0
SC	Set C	1 → C
RC	Reset C	0 → C
IFC	IF C	Do next if C = 1
IFNC	IF not C	Do next if C = 0
TRANSFER OF CONTROL INSTRUCTIONS		
JSRP	Jump subroutine from table	PC → [SP],SP+2 → SP W(table#) → PC
JSR	Jump subroutine relative	PC → [SP],SP+2 → SP,PC+ # → PC (#is +1025 to −1023)
JSRL	Jump subroutine long	PC → [SP],SP+2 → SP,PC+ # → PC
JP	Jump relative short	PC+ # → PC(# is +32 to −31)
JMP	Jump relative	PC+ # → PC(#is +257 to −255)
JMPL	Jump relative long	PC+ # → PC
JID	Jump indirect at PC + A	PC+A+1 → PC
JIDW		then Mem(PC)+PC → PC
NOP	No Operation	PC + 1 → PC
RET	Return	SP−2 → SP,[SP] → PC
RETSK	Return then skip next	SP−2 → SP,[SP] → PC, & skip
RETI	Return from interrupt	SP−2 → SP,[SP] → PC, interrupt re-enabled

Note: W is 16-bit word of memory
MA is Accumulator A or direct memory (8 or 16-bit)
Mem is 8-bit byte or 16-bit word of memory
MemI is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data
imm8 is 8-bit immediate data only

FIGURE 3. HPC Instruction Set Description

```
      LD  X,#0100              ; Point to beginning of source code
      LD  BK,# 0400, #0600;P   ; Point to beginning & end of target
LOOP: LD  A, [X+].W            ; Get word from source block
      XS  A, [B+].W            ; Store it at target
      JP  LOOP
```

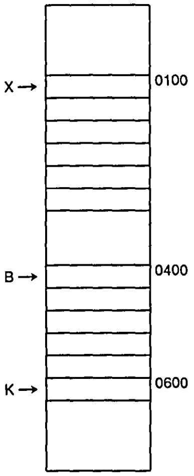


FIGURE 4. Word Block Move

Indexed: As Indirect, but with an 8- or 16-bit immediate offset added to the pointer.

Register Indirect: As indirect, but the B or X registers are used as pointers, with their addresses implied in the opcode.

Immediate: Only for the source in two-operand instructions. An 8- or 16-bit immediate value is included in the instruction.

The first four addressing modes are used both for single operand instructions e.g. bit set, bit clear, bit test, increment, decrement, and two operand instructions such as ADD and LD.

Direct and immediate modes can be used in combination, allowing operations to be performed directly on memory or registers without using the accumulator.

Two variables, each byte or word, each located anywhere in memory, can be compared, added, divided or have any of the other two-address instructions performed on them. This improves the byte-efficiency of the HPC, and enhances the power of the instruction set in that it takes less lines of assembly code to perform a given function than it would for earlier, completely accumulator-based CPUs.

An important benefit provided by the indirect and indexed modes is that any of the 96 words of RAM or the basepage registers, such as port A or the accumulator, may be used as pointers.

There are two special addressing modes which are used only with the LD and X (exchange) instructions. These modes are called auto increment/decrement and auto increment/decrement with conditional skip, and their use is illustrated by the example shown in *Figure 4.*

This example uses the B pointer, the X pointer and the K register to move a block of data one word at a time. Some points to note are that the LD BK instruction initializes both registers with one instruction, and that both the LD and XS instructions increment the pointer by two because two bytes (one word) are moved. The S in XS signifies the conditional skip. After A has been exchanged with the word pointed to by B, B is incremented, then compared with K. If B is greater than K (or, for an XS A, [B−] instruction, if B is less than K) the next statement is skipped over, thus terminating the loop. This example epitomizes the approach taken in designing the HPC family.

String operations are built up from simple data movement instructions, allowing them to be interrupted at any time with no need for complex re-start or recovery schemes.

INSTRUCTION SET

The HPC instruction set is noticeably different from other 16-bit controllers, in that many of its instructions are single byte. How this is achieved can be seen by looking at the opcode map *(Figure 5).*

Instructions such as bit manipulation operations and single byte jumps (JP) use many opcodes for the same mnemonic. This is because information, such as the jump length for JP, is coded into the opcode.

This makes these instructions very efficient, and enhances the performance of the HPC in information control applications, where decision making and bit manipulation operations tend to be important.

All of the arithmetic, comparison, logical and data movement instructions have a single byte form using register indirect addressing mode. The opcode space "used up" by having many opcodes for a few instructions is restored by using addressing mode prefixes for the less commonly used addressing modes. These make instructions using these modes one byte longer, but the use of these prefixes allows all of the two address instructions to use all of the addressing modes. Without the prefixes the HPC would run out of opcode space and restrictions would have to be placed on some instructions, making the assembly language much harder to use and the C compiler harder to write. Examples are given in *Figure 6* of several combinations of instructions and addressing modes, with execution times for systems using low cost external memories.

C.13 HPC OPCODE MAP

LSB/MSB →

	0	1	2	3	4	5	6	7
0	CLR A	IFBIT 0	JSRP 0	JSR +	JP +1*	JP +17	JP 0	JP −16
1	COMP A	IFBIT 1	JSRP 1	JSR +	JP +2	JP +18	JP −1	JP −17
2	SC	IFBIT 2	JSRP 2	JSR +	JP +3	JP +19	JP −2	JP −18
3	RC	IFBIT 3	JSRP 3	JSR +	JP +4	JP +20	JP −3	JP −19
4	INC A	IFBIT 4	JSRP 4	JSR −	JP +5	JP +21	JP −4	JP −20
5	DEC A	IFBIT 5	JSRP 5	JSR −	JP +6	JP +22	JP −5	JP −21
6	IFNC	IFBIT 6	JSRP 6	JSR −	JP +7	JP +23	JP −6	JP −22
7	IFC	IFBIT 7	JSRP 7	JSR −	JP +8	JP +24	JP −7	JP −23
8	SBIT 0	RBIT 0	JSRP 8	RBIT X	JP +9	JP +25	JP −8	JP −24
9	SBIT 1	RBIT 1	JSRP 9	SBIT X	JP +10	JP +26	JP −9	JP −25
A	SBIT 2	RBIT 2	JSRP 10	IFBIT X	JP +11	JP +27	JP −10	JP −26
B	SBIT 3	RBIT 3	JSRP 11	SWAP A	JP +12	JP +28	JP −11	JP −27
C	SBIT 4	RBIT 4	JSRP 12	RET	JP +13	JP +29	JP −12	JP −28
D	SBIT 5	RBIT 5	JSRP 13	RETSK	JP +14	JP +30	JP −13	JP −29
E	SBIT 6	RBIT 6	JSRP 14	RETI	JP +15	JP +31	JP −14	JP −30
F	SBIT 7	RBIT 7	JSRP 15	POP	JP +16	JP +32	JP −15	JP −31
	8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	Dir−Dir	LD A,i	Dir−Dir	LD A,ii	LDS [B+].b	LD [X+],b	LDS [B+].w	LD [X+].w
1	Dir−Dir	LD K,i	Dir−Dir	LD K,ii	XS [B+].b	X [X+],b	XS [B+].w	X [X+].w
2	Imm−Dir	LD B,i	Index	LD B,ii	LDS [B−].b	LD [X−],b	LDS [B−].w	LD [X−].w
3	Imm−Dir	LD X,i	—	LD X,ii	XS [B−].b	X M[X−],b	XS [B−].w	X [X−].w
4	Dir−Dir	JMP+	Dir−Dir	JMPL	LD [B].b	LD [X].b	LD [B].w	LD [X].w
5	Dir−Dir	JMP−	Dir−Dir	JSRL	X [B].b	X [X].b	X [B].w	X [X].w
6	Imm−Dir	Direct	Index	Direct	ST [B].b	ST [X].b	ST [B].w	ST [X].w
7	Imm−Dir	LD bd,i	LD BK,ii	LD wd,ii	SHR A	RRC A	SHL A	RLC A
8	LD A,bd	ADD A,i	LD A,wd	ADD A,ii	ADC A,b	ADD A,b	ADC A,w	ADD A,w
9	INC bd	AND A,i	INC wd	AND A,ii	DADC A,b	AND A,b	DADC A,w	AND A,w
A	DECSZ bd	OR A,i	DECSZ wd	OR A,ii	DSUBC A,b	OR A,b	DSUB A,w	OR A,w
B	ST A,bd†	XOR A,i	ST A,wd†	XOR A,ii	SUBC A,b	XOR A,b	SUBC A,w	XOR A,w
C	LD bd,bd	IFEQ A,i	LD wd,wd	IFEQ A,ii	JID	IFEQ A,b	JIDW	IFEQ A,w
D	LD BK,i	IFGT A,i	Indirect	IFGT A,ii	—	IFGT A,b	—	IFGT A,w
E	X A,bd	MULT A,i	X A,wd	MULTA,ii	—	MULT A,b	—	MULT A,w
F	XIndirect	DIV A,i	PUSH	DIV A,ii	DIVD A,b	DIV A,b	DIVD A,w	DIV A,w

— = opcode is reserved for future use.
b = byte of memory
bd = direct byte of memory
i = 8-bit immediate value
w = word of memory
wd = direct word of memory
ii = 16-bit immediate value
Dir−Dir, Imm−Dir, Index, Direct, Indirect and XIndirect are all Addressing Mode directives.

Notes:

*NOP is the same as JP + 1 and has the same opcode.

†These opcodes are LD if prefixed by Dir−Dir or Imm−Dir directive.

FIGURE 5

		20 MHz	30 MHz
CLR	A	300 ns	200 ns
RRC	A	400 ns	267 ns
LD	B, H'3CF2	600 ns	400 ns
IFBIT	7,[B].B	800 ns	533 ns
ST	A,38.W	900 ns	600 ns
JSR		1.10 µs	733 µs
JSRL		1.40 µs	933 µs
ADC	[H'10].W, [H'20].W	1.70 µs	1.13 µs
DSUBC	[H'A0].W, [H'B0].W	2.00 µs	1.33 µs
MULT	A, [B].W	5.90 µs	3.93 µs
DIVD	A,[X].W	6.40 µs	4.27 µs

Times Calculated with 1 Wait State Inserted

FIGURE 6. Typical Execution Times

There are many more powerful features of the HPC instruction set, but space does not permit describing them here. For more information see the documents listed in the references section.

TECHNOLOGY

The HPC family and nearly all other new National Semiconductor analog and digital VLSI devices are fabricated in an advanced double metal process called M2CMOS. This is a very high speed process, as shown by the current production two micron (drawn) HPC46083, which is available as a 30 MHz version.

The HPC family has been migrated to a 1.5 micron (drawn) process for the first part with an analog to digital converter on chip, the HPC46164.

National Semiconductor already manufactures the NS32532 microprocessor in 1.25 micron M2CMOS, and will shrink this process still further in the future. The HPC devices will be migrated to these smaller geometries and will benefit from other process developments such as on chip EPROM.

INFORMATION CONTROL APPLICATIONS

Laser Beam Printer Front End Processor

This section describes a customer's application for an HPC46083 used in single chip mode. It makes use of the Universal Peripheral interface (UPI) port which is a feature of all HPC devices with on-chip mask ROM.

The UPI port allows an HPC device to be used as a peripheral to a host processor, connected to the host via its data bus. The HPC in UPI mode appears to the host to be a peripheral device such as a UART, but provides additional processing power, relieving the host of interrupt-intensive tasks and thus improving the host's performance.

The UPI port of the HPC provides status signals to both the HPC CPU and that of the host which ensure that no data is lost when the CPUs communicate.

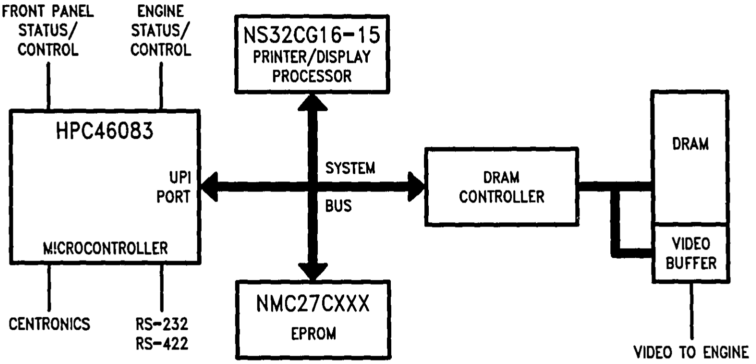
In the laser beam pointer (LBP) application *(Figure 7)*, the HPC handles the serial and Centronics interfaces of the printer, buffering received characters and interrupting the host CPU when a block of up to 128 characters has been received. When the host CPU (a National Semiconductor NS32CG16 printer/display controller) is interrupted it then transfers the whole block of data into its own memory very rapidly.

This approach reduces the number of interrupts received by the 32CG16 by a factor of over 100 compared to a solution using a conventional UART while being simpler, cheaper and offering higher system performance than using a DMA approach. These overhead reductions are very important in LBP systems, because the main CPU must keep up with the paper movement, otherwise image data will be lost.

In addition to improving printer performance, the HPC reduces the system cost by providing functions that would otherwise need extra devices. The HPC acts as the interrupt controller for the 32CG16, generating an interrupt signal to it and then placing the interrupt vector on the UPI port when the 32CG16 acknowledges the interrupt. Another function provided by the HPC is an intelligent interface to the printer front panel displays and push buttons controlling such functions as LCD contrast. Finally, the HPC implements a serial interface to the electronic subsystem of the printer engine itself, providing diagnostic capability to the 32CG16. For all of these functions, the HPC performs first-level error checking, further relieving the main CPU of minor tasks.

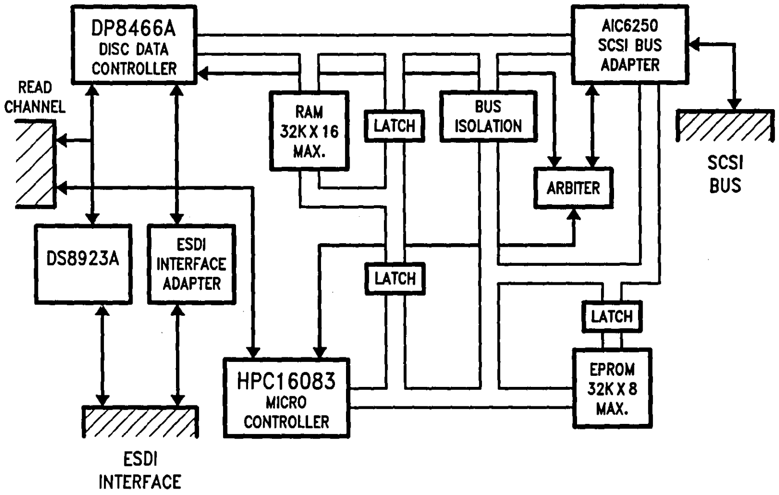
The LBP is at one extreme of the range of HPC applications, where the HPC uses virtually nothing but its on-chip peripherals and memories.

The next section deals with an application towards the other end of the range.



TL/DD/10346-2

FIGURE 7



TL/DD/10346-3

FIGURE 8

SCSI Bridge Adapter

The fast growing usage of Winchester disk drives in the Small Computer System Interface (SCSI) environment has provided another important market for the HPC family.

The HPC architecture is well suited for use in embedded SCSI systems, as the peripherals such as the SCSI interface device may be memory mapped into the HPC address space, allowing bit and byte manipulation operations to be performed directly on the registers of the peripheral using single assembly language instructions. Many SCSI interface devices are relatively unintelligent, requiring the CPU to perform many bit test, set, and clear operations to set up a data transfer operation. Most other microcontrollers need up to three instructions to set a bit in one of these peripherals, thus reducing drive performance.

National Semiconductor has produced an ESDI-to-SCSI bridge adapter board, which demonstrates the use of the HPC46003 and the DP8466A disk data controller in a real synchronous SCSI system. A software package has been written in HPC assembly language which implements the SCSI common command set and is available in source code form to companies wishing to use the HPC in embedded SCSI or host adapter designs.

The code was written in HPC assembly language because for very high volume, cost sensitive designs, like a disk drive, the extra development cost of writing in assembler is outweighed by the advantages of reduced code size and improved performance.

The adapter board design *(Figure 8)* uses the HPC46003 running in 8-bit mode with a single EPROM providing program memory. Data memory is provided by the 256 bytes of on-chip RAM which provides fast scratch pad and stack space.

One important function in embedded SCSI disk drives is logical to physical address conversion, in which a logical address (typically 24 bits) is divided twice by constants, the result and the two remainders being the head, cylinder and sector numbers.

The HPC is capable of dividing a 32-bit number by a 16-bit number in under four microseconds, thus providing a dramatic improvement in logical to physical address conversion time compared to earlier 8-bit microcontroller solutions. As a final point in this necessarily brief discussion, the HPC uses very little power due to its advanced CMOS manufacturing technology. This is important in disk drive applications, where low power consumption is an important performance parameter for the end product.

CONCLUSION AND FUTURE DEVELOPMENTS

This paper has discussed the design of the HPC family and described two actual applications in important market areas. The development work performed for these and other projects has shown that the HPC architecture provides very high performance in embedded control applications.

The plans for future products are to take the high performance core and add various combinations of peripherals, thus allowing the family to reach a wide range of markets. *Figure 9* shows some of the current and future devices.

HPC16083	8K ROM, 256 RAM
HPC16003	ROMless, 256 RAM
HPC16400	256 RAM, 2 HDLC + 4 DMA Channels
HPC16164	16K ROM, 512 RAM, 8 Channel ADC
HPC16064	16K ROM, 512 RAM
HPC16104	ROMless, 8 Channel ADC
HPC16004	ROMless, 512 RAM
HPC167164	16K EPROM, 512 RAM, 8 Channel ADC

FIGURE 9. HPC Family Devices Principal Features

REFERENCES

National Semiconductor Application Note AN-510: *Assembly Language programming for the HPC.*

National Semiconductor Publication Number 424410897-001A July 1987: *HPC16083/HPC16043/HPC16003 User's Manual.*

Pulse Width Modulation Using HPC

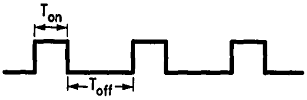
National Semiconductor
Application Note 586
Alvin Chan, Bill Miller,
Bob Moeckel, Bob Hanrahan

As the use of MicroControllers in embedded control applications grows in popularity, we find more use of width modulated pulse trains. Typical applications that use Pulse Width Modulation are automotive engine control, motor speed control, display intensity control, and sound generation.

PWM DEFINITION

Pulse width modulation is simply a method of communicating information to a device. It can be viewed as an analog signal provided in digital form. *Figure 1* shows a typical timing diagram of a PWM signal. The duty cycle is expressed as the duration of T_{on} over the sum of T_{on} and T_{off}. A signal has a constant duty cycle if T_{on} and T_{off} are uniform. If T_{on} is equal to T_{off}, the signal has a 50% duty cycle.

$$\text{Duty Cycle} = \frac{T_{on}}{T_{on} + T_{off}}$$

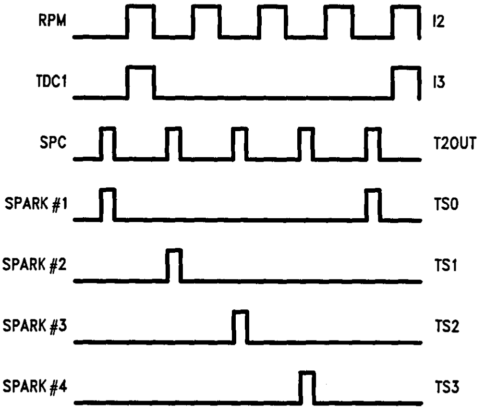


TL/DD/10347–1

FIGURE 1. A Typical PWM Signal

TYPICAL APPLICATIONS THAT REQUIRE PWM

One element of an automotive engine control system is the spark ignition. In a distributorless ignition system, spark control signals are required to appear in sequence, with a time delay between each of them. Typical signals for a four spark plug system are shown in *Figure 2*. The generation of these signals will be explained further in the timer synchronous output section.

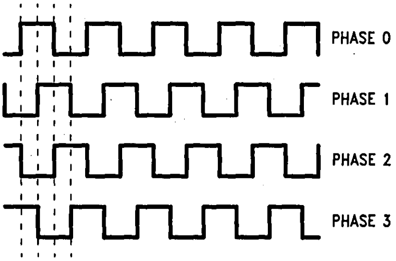


TL/DD/10347–2

FIGURE 2. HPC Based Spark Ignition Control

Another element of an automotive system is the carburetion and idle speed control. When no pressure is applied to the accelerator pedal, the throttle is completely shut off. The idle speed control utilizes a stepping motor to operate an auxilliary fuel valve. *Figure 3* shows the control signals that have to be generated for a four phase stepper motor. Each of the PWM signals should have a phase lag of one quarter of a cycle from the previous one.

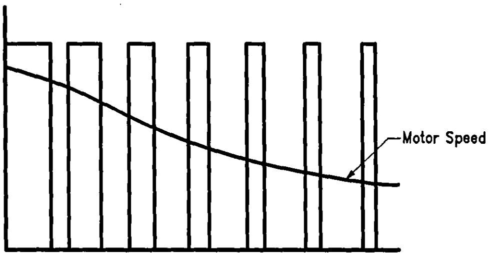
PWM is applied to motor speed control. The speed of a dc motor is directly proportional to the voltage applied.



TL/DD/10347–3

FIGURE 3. Stepper Motor Control Signals

PWM is used selectively to switch full supply power on and off to the motor at some frequency and duty cycle. The bigger the duty cycle, the more power is supplied to the motor. Hence the speed is higher. Motor speed can be controlled by adjusting the ON time of the signal. *Figure 4* depicts the relationship of motor speed and the applied signal.



TL/DD/10347–4

FIGURE 4. Using PWM to Control Motor Speed

The same manipulation also applies to controlling the intensity of light emitting diodes. The brightness of the LED can be varied by using different duty cycles.

Sound synthesis can be achieved by uniting the process of sinusoidal signal generation and envelope generation.

A sinusoidal signal can be generated by a variety of methods. A common technique is to use Walsh functions. Walsh functions are the digital equivalent of Fourier Series. They are essentially pulse signals with varying duty cycles. The individual Walsh components are generated by the microcontroller and combined with the proper weighting factors to form the sinusoids.

Envelope generation can be done by using PWM to build a D/A converter. The envelope will give the composite sinusoidal signal the characteristic sharp attack followed by slow decay. The amplitude of the envelope function is altered by changing the duty cycle of the PWM input to the D/A converter. This function is performed by another timer.

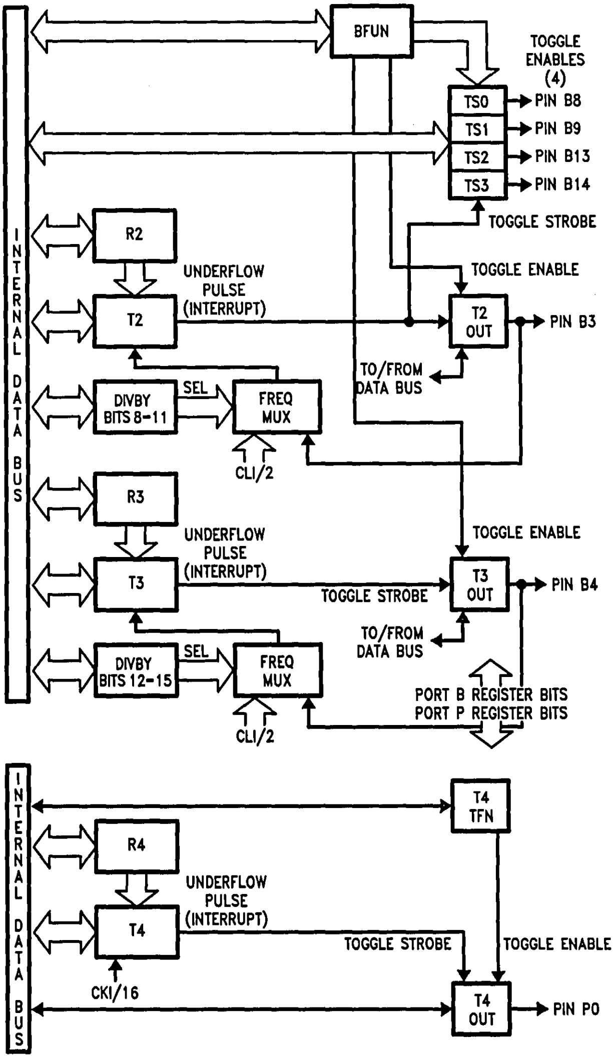
HPC Implementation

National Semiconductor's HPC, High Performance MicroController, provides a simple method for generating width modulated pulse trains, with little or no software overhead, by use of the device's 9 on-chip timers, T0 through T8.

SETTING UP HPC TO DO PWM

PWM Outputs In the HPC

Timers T1 through T7 are down-counters with associated input registers R1 through R7. The value in the registers is loaded automatically into the timers when the timers underflow. Timers T2 through T7 have individual output signals which toggle when the timers underflow. Interrupts are generated at the time of underflow *Figure 5* shows the structure of these timers.



TL/DD/10347-5

Note: Only Time 4 is Shown. T5, T6, and T7 are identical.

FIGURE 5. HPC Timers T2–T7

Timers T2 through T7 can be separated into 2 groups. Different procedures and registers are used to set up the two groups of timers. In one group is timers T4 through T7, which are dedicated to PWM applications. They count down at a constant rate of 1/16 of the input clock (CKI/16) while enabled to do so. In the other group are the more versatile timers, T2 and T3. The clock input to timers T2 and T3 may be independently selected as coming from one of 14 available prescaled versions of the CKI clock, or from an external pin, as specified in the DIVBY register. Timer T2 can also be specified to be clocked on underflows from timer T3 by appropriate selection in the DIVBY register; the pair then form, in effect, a single 32-bit counter.

With timers T4 through T7, the maximum PWM frequency that can be achieved is half of CKI/16. The associated register provides a 16-bit resolution for the duration of the pulse width.

To use T2 and T3 as PWM timers, the clock must come from an internal source. By configuring the DIVBY register and selecting a value for the counter, the maximum frequency that can be achieved is half CKI/16 and the minimum frequency is half (CKI/131072)/65536.

50% Duty Cycle PWM

On underflow of the timers T2 through T7, the value in the corresponding input register is automatically reloaded into the counters. Therefore a 50% duty cycle PWM can be generated without software intervention once the timer is set up.

Listings 1 and 2 illustrate the use of T4 and T2 in generating PWM outputs. The PWM frequency to be generated is 20 kHz. By using a 16 MHz crystal and CKI/16 as the input clock, the counter value to be loaded into the registers is 24 so that an underflow occurs and the output toggles every 25 µs.

Generating Non 50% Duty Cycle PWM without Listing 1 Use of T4 to Generate 50% Duty Cycle

```
        .TITLE  'T4 PWM 50% DC'
        .SECT   CODE,ROM16,REL
TMMODE  =       0190:W
DIVBY   =       018E:W
T4      =       0140:W
R4      =       0142:W
T5      =       0144:W
R5      =       0146:W
PWMODE  =       0150:W
PORTP   =       0152:W
PWMSTR: LD      SP,#STKS      ;initialize stack pointer
        LD      PWMODE,#0x4   ;stop timer T4
        NOP                   ;delay to provide 8
                              ;CK2 cycles
        NOP                   ;to make sure timer
                              ;is updated
        LD      PWMODE,#0xC      ;clear T4
                              ;interrupt pending bit
        LD      T4,#24        ;load T4 with counter
                ;value to obtain a 20 kHz PWM
                ;frequency the counter should
                ;underflow on a 40 kHz frequency,
                ;therefore by using a 16 MHz crystal
                ;and CKI/16 input to the timer, the
                ;counter value should be 24
                ;16 MHz/16/25 = 40 kHz
        LD      R4,#24        ;load auto-reload
                              ;register
        SBIT    0,PORTP       ;set initial value of
                              ;output pin for T4 to 0
        SBIT    3,PORTP       ;enable toggling of
                              ;pin on underflow
        RBIT    2,PWMODE      ;start timer
STOP:
        JP      STOP
        .ENDSECT

        .SECT   STACK,BASE
STKS:   DSW     10
        .ENDSECT
        .END    PWMSTR
```

Non 50% Duty Cycle PWM (Software/Interrupts)

Timers T1 through T7 will generate an interrupt on underflow. For non-50% duty cycle PWM, software has to be involved in controlling the duty cycle. The same software for the 50% duty cycle is used to set up the timers for counting down. On interrupt from the timers, the interrupt service routine loads the other half of the cycle time into the timer register.

On each interrupt from the timer the user software alternately loads T_{on} and T_{off} into the register. The result is a constant duty cycle output. Examples of programming the interrupts are shown in listings 3 and 4.

TIMER SYNCHRONOUS OUTPUTS OF TIMER T2

Timer T2 has in addition to the normal output pin, four output pins which can be independently selected. These pins are referred to collectively as the "Timer Synchronous" outputs. *Figure 2* shows the synchronous output being applied to engine control in spark ignition. The signals TS0 to TS3 are synchronous outputs derived from timer T2. By enabling each pin in sequence, the spark control signals SP1 to SP4 can be generated.

SOFTWARE INTERVENTION

Another problem facing the designer of a MicroController based system is that software overhead must be kept to a minimum. Interrupt latency and changing input registers can use a significant portion of the time which would otherwise be available for processing of sensor data.

The conventional way of generating non-50% duty cycle was discussed earlier. That involves software changing the value of the auto-reload register every time the timer counts down and interrupts. Two timers can be used to generate two synchronized and offset 50% duty cycle pulses. By EXCLUSIVE-ORing them, a non-50% duty cycle PWM is generated.

Listing 2 Use of T2 to Generate 50% Duty Cycle

```
          .TITLE 'T2 PWM 50% DC'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
T2        =      0188:W
R2        =      0186:W
PWMSTR:   LD     SP,#STKS     ;initialize stack pointer
          LD     TMMODE,#0x400  ;stop timer T2
          NOP                 ;delay to provide 8
          NOP                 ;CK2 cycles to make
                              ;sure timer is updated
          LD     TMMODE,#0xC00  ;clear T2
                              ;interrupt pending bit
          SBIT   3,BFUN       ;set pin 3 of port B
                              ;as timer 2 output
          SBIT   3,DIRB       ;set output direction
                              ;on port B pin 3
          LD     DIVBY,#0x200;set clock as CKI/16
                              ;for T2
          LD     T2,#24       ;load T2 with counter
                 ;value to obtain a 20 kHz PWM
                 ;frequency the counter should
                 ;underflow on a 40 kHz frequency
                 ;therefore by using a 16 MHz crystal
                 ;and CKI/16 input to the timer, the
                 ;counter value should be 24
                 ;16 MHz/16/25 = 40 kHz
          LD     R2,#24       ;load auto-reload
                              ;register
          RBIT   3,PORTB      ;initialize output pin
                              ;value to 0
          RBIT   3,TMMODE     ;start timer
STOP:
          JP     STOP
          .ENDSECT

          .SECT  STACK,BASE
STKS:     DSW    10
          .ENDSECT
          .END   PWMSTR
```

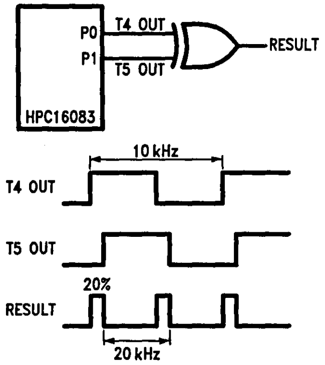
Listing 3 Use of T4 to Generate Non-50% Duty Cycle with Interrupts

```
          .TITLE 'T4 NON-50% DC'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
T4        =      0140:W
R4        =      0142:W
T5        =      0144:W
R5        =      0146:W
PWMODE    =      0150:W
PORTP     =      0152:W
ENIR      =      00D0:B
IRPD      =      00D2:B
PWMSTR:   LD     SP,#STKS   ;initialize stack pointer
          LD     PWMODE,#0x4;stop timer T4
          NOP               ;delay to provide 8
          NOP               ;CK2 cycles to make
                            ;sure timer is updated
          LD     PWMODE,#0xC   ;clear T4
                            ;interrupt pending bit
          LD     ENIR,#00   ;disable interrupts
          LD     IRPD,#00   ;clear interrupt
                            ;pending bits
          LD     T4,#9      ;load T4 with counter
                ;value to obtain a 20 kHz PWM
                ;frequency with 20% duty cycle
                ;using a 16 MHz crystal and CKI/16
                ;input to the timer, the counter
                ;value should be 9
          LD    R4,#39           ;load auto-reload
                                 ;register with count
                                 ;of 80%
          LD    TCYCLE,#48       ;set total cycle time
                                 ;count −2
          SBIT  0,PORTP          ;set initial value of
                                 ;output pin for T4 to 0
          SBIT  3,PORTP          ;enable toggling of
                                 ;pin on underflow
          LD    ENIR,#0x20       ;enable timer interrupt
          RBIT  2,PWMODE         ;start timer
STOP:
          JP    STOP
          .ENDSECT

          .SECT STACK,BASE
STKS:     DSW   10
          .ENDSECT

          .IPT  5,INTRPT5

          .SECT DATA,BASE,REL
TCYCLE:   .DSW  1                ;total cycle time
          .ENDSECT

          .SECT SUBR,ROM 16,REL
INTRPT5:
          LD    A,TCYCLE         ;get total cycle time
          SC
          SUBC  A,R4             ;subtract current
                                 ;counter
          ST    A,R4             ;to get alternate cycle
                                 ;time and store to
                                 ;auto-reload reg
          RETI
          .ENDSECT
          .END  PWMSTR
```

Listing 4 Use of T2 to Generate Non-50% Duty Cycle with Interrupts

```
          .TITLE 'T2 NON-50% DC'
          .SECT  CODE,ROM16,REL
TMMODE    =      0190:W
DIVBY     =      018E:W
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
T2        =      0188:W
R2        =      0186:W
ENIR      =      00D0:B
IRPD      =      00D2:B
PWMSTR:   LD     SP,#STKS     ;initialize stack pointer
          LD     TMMODE,#0x400  ;stop timer T2
          NOP    ;delay to provide 8 CK2 cycles
          NOP    ;to make sure timer is updated
          LD     TMMODE,#0xC00  ;clear T2
          LD     ENIR,#00     ;interrupt pending bit
                              ;disable interrupts
          LD     IRPD,#00     ;clear interrupt
                              ;pending bits
          SBIT   3,BFUN       ;set pin 3 of port B
                              ;as timer 2 output
          SBIT   3,DIRB       ;set output direction
                              ;on port B pin 3
          LD     DIVBY,#0x200;set clock as CKI/16
                              ;for T2
          LD     T2,#9        ;load T2 with counter
                 ;value to obtain a 20 kHz PWM
                 ;frequency using a 16 MHz crystal
                 ;and CKI/16 input to the timer, the
                 ;counter value should be 9
          LD     R2,#39       ;load auto-reload
                              ;register with count
                              ;of 80%
          LD     TCYCLE,#48   ;set total cycle time
                              ;count -2
          RBIT   3,PORTB      ;initialize output pin
                              ;value to 0
          LD     ENIR,#0x20   ;enable timer interrupt
          RBIT   3,TMMODE     ;start timer
STOP:
          JP     STOP
          .ENDSECT

          .SECT STACK,BASE
STKS:     .DSW   10
          .ENDSECT

          .IPT   5,INTRPT5

          .SECT  DATA,BASE,REL
TCYCLE:   .DSW   1            ;total cycle time
          .ENDSECT

          .SECT  SUBR,ROM16,REL
INTRPT5:
          LD     A,TCYCLE     ;get total cycle time
          SC
          SUBC   A,R2         ;subtract current
                              ;counter
          ST     A,R2         ;to get alternate cycle
                              ;time and store to
                              ;auto-reload reg
          RETI
          .ENDSECT
          .END   PWMSTR
```

Figure 6 shows the result of EXCLUSIVE-ORing the two timers. The duty cycle depends only on the phase shift between the timer outputs. In can be seen that the resulting frequency is actually twice the frequency of the original timers. Therefore, in order to generate a 20 kHz result, two 10 kHz timers must be used. The code is shown in listing 5. By varying the initial delay in the second timer, different duty cycles can be chosen. In the example given, a one digit difference in the counter value results in a 2% difference in the duty cycle.



TL/DD/10347-6

FIGURE 6. Using 2 Timers to Generate Non 50% Duty Cycle

Listing 5 Use of T4, T5 to Generate Non-50% Duty Cycle without Interrupts

```
         .TITLE 'NON 50% PWM (T4,T5)'
         .SECT  CODE,ROM16,REL
TMMODE   =      0190:W
DIVBY    =      018E:W
T4       =      0140:W
R4       =      0142:W
T5       =      0144:W
R5       =      0146:W
PWMODE   =      0150:W
PORTP    =      0152:W
FREQ:    .DW    49          ;counter value for the timers
                            ;this generates 10 kHz PWM
DC:      .DW    20          ;duty cycle = 20%
PWMSTR:  LD     SP,#STKS
         LD     PWMODE,#0X44    ;stop T4 and
                                ;T5
         NOP
         NOP
         LD     PWMODE,#0XCC    ;clear T4, T5
                            ;int pending bits
         LD     T4,FREQ     ;load T4, R4, R5
         LD     R4,FREQ     ;with counter value
         LD     R5,FREQ
         LD     A,DC        ;calculate delay for
         MULT   A,FREQ          ;T5
         DIV    A,#100
         ST     A,T5        ;store in T5
         LD     PORTP,#0X10 ;set output pins, T4
                            ;low T5 high
         SBIT   3,PORTP     ;enable toggling of
         SBIT   7,PORTP     ;pins on underflow
         AND    PWMODE,#0XFFBB  ;start T4 and
                                ;T5
STOP:
         JP     STOP
         .ENDSECT

         .SECT  STACK,BASE
STKS:    .DSW   10
         .ENDSECT
         .END   PWMSTR
```

Listing 6 Use of T2, T3 to Generate Non-50% Duty Cycle without Interrupts

```
          .TITLE 'NON 50% PWM (T2,T3)'
          .SECT  CODE,ROM16,REL
BFUN      =      00F4:W
DIRB      =      00F2:W
PORTB     =      00E2:W
TMMODE    =      0190:W
DIVBY     =      018E:W
T4        =      0140:W
R4        =      0142:W
T5        =      0144:W
R5        =      0146:W
PWMODE    =      0150:W
PORTP     =      0152:W
T2        =      0188:W
R2        =      0186:W
R3        =      018A:W
FREQ:     .DW    49   ;counter value for the timers
                      ;this generates 10 kHz PWM
DC:       .DW    20   ;duty cycle = 20%
PWMSTR:   LD     SP,#STKS
          LD     TMMODE,#0x4400    ;stop T2,T3
          NOP
          NOP
          LD     TMMODE,#0xCCC8    ;clear T2, T3
                                   ;int pending bits
          LD     PORTB,#0x10       ;set output pins, T2
                                   ;low T3 high
          LD     DIRB,#0xFFFF      ;output on PORT B
          OR     BFUN,#0x0018      ;set T2,3 as timers
          OR     DIVBY,#0x2200     ;select CKI/16 clock
          LD     T2,FREQ           ;load T2 R2, R3 with
                                   ;counter value
          LD     R2,FREQ
          LD     R3,FREQ
          LD     A,DC              ;calculate delay for
          MULT   A,FREQ               ;T3
          DIV    A,#100
          ST     A,R3              ;store delay in T3
          AND    TMMODE,#0xBBFF       ;start T2,T3
          STOP:
          JP     STOP
          .ENDSECT

          .SECT  STACK,BASE
STKS:     .DSW   10
          .ENDSECT
          .END   PWMSTR
```

CONCLUSION

PWM is easily generated by the HPC 16083 with its abundant source of timers. With a 30 MHz crystal, the maximum PWM frequency that can be achieved is 937.5 kHz. The timers run by themselves once the proper setup is performed. A method of obtaining non-50% duty cycle PWM without software intervention was presented.

C in Embedded Systems and the Microcontroller World

National Semiconductor
Application Note 587
David LaVerne

ABSTRACT

C is becoming the higher-level language of choice for microcontroller programming. Traditional usage of C depends on assembly language for the intimate interface to the hardware. A few extensions to ANSI C allow embedded systems to connect directly and simply, using a single language and avoiding detailed knowledge of the compiler and hardware connections.

HIGHER-LEVEL LANGUAGE USAGE

The desires leading to the greater use of higher-level languages in microcontrollers include increased programmer productivity, more reliable programs, and portability across hardware. Few such languages have served well when required to manipulate hardware intimately because most have been for mathematical computation. The C language has always been close to machine level. Indeed Kernighan and Ritchie[1] refer to it as not really a higher-level language; one view of C is as a higher-level syntax expressing PDP-11 assembly language.

C has gained a great deal of its reputation and popularity associated with its use for operating systems, specifically UNIX®[2] and similar systems. Many languages will do well enough for the application and utility programs of such a system, but being appropriate for the kernel indicates C can probably do the job of hardware control in an effective manner.

The needs of an embedded system, however, are not identical to the environment from which C has come. This warrants looking at C as it is and comparing it to the needs of C for the microcontroller world.

Operating Systems vs Embedded Systems

In most non-embedded programs, it is the processing which is important, and the Input/Output is only to get the data and report the results. In embedded or realtime applications, it is the Input/Output which is vital, and the processing serves only to connect inputs with outputs.

Operating systems are actually not as closely tied to the hardware as they might appear initially, and those portions which are close are not very portable. Operating systems manipulate hardware registers primarily for memory management (to map tasks), task process switching (to activate tasks), interrupt response (to field requests), and device drivers (to service requests). Because memory management hardware is so different between systems; because task process changing is so contingent on processor operations and compiler implementations; because interrupt system behavior is so varied; and because device control is so dependent on architecture and busses, these particular aspects of the operating system are not concerned with portability. As a result, they are generally kept separate, use a less convenient form of C depending on constants, and frequently are implemented in assembly language. This is not a major problem, since they comprise only a small portion of the total system, and have to change anyway each time the system is ported.

Embedded systems, by their very nature, are closely tied to the hardware throughout the system. The system consists of manipulating the hardware registers, with varying amounts of calculation and data transformations interspersed with the manipulations. As the system gets larger, the calculations may get more complex and may become a larger share of the program, but it is still the hardware operations which are the purpose of the system. Because the system in which these hardware pieces reside consists mostly of these hardware pieces, it is reasonable to hope for portability across processors or controllers for an application or product. Attempting to isolate all of the hardware operations is often impractical; using inconvenient forms of C is troublesome throughout the system and throughout its life-cycle; and implementing them in assembly language defeats the advantages of higher-level language usage and eliminates portability for those (and related) portions. For embedded systems, conveniently accessing hardware registers while doing calculations is essential.

Computer Systems vs Embedded Systems

Computational systems generally can be down the cable, and thus down the hall, from where they are used and can be whatever size is necessary to get the performance; production quantities are measured in hundreds and thousands, so price is a price/performance issue. Embedded systems end up tucked away in some of the strangest and tiniest places, so size can be a success or failure issue; quantities are often tens of thousands to millions of units, so additional chips or costs are multiplied ferociously and become a bottom-line issue.

The computer systems for which C was originally developed were relatively small and not especially sophisticated. However, as systems have grown, C and its implementation has grown right along with them. Most computer systems for which C is used now involve high-speed processors with large memory caches to huge memory spaces, backed by virtual memory. Many have large register sets. Such linear memory with heuristic accelerators allow for very large programs and fast execution. A major effort in optimization is in the allocation and usage of the registers, which tend to be general purpose and orthogonally accessible. Such systems, processor chips, and compilers compete almost exclusively in the field of speed.

Embedded systems, and most especially microcontrollers, have a different nature. While some applications may add external devices and memories to the controller, many are meant to be fully self-contained on one chip or have at most a few I/O chips. Microcontroller systems are small, are often required to fit in a physically small space, and are usually fed small amounts of power. Even when the system is externally expanded, the memories provided on-chip are significantly faster than the external memories because of buss driving. The total addressing space is usually very limited (32k, 64k) with expansion not linear. The registers in microcontrollers are usually a limited number of special purpose registers, thus eliminating orthogonal usage. Speed is only one of many considerations in the microcontroller competition. Cost, package size, power consumption, memory size, number of timers, and I/O count are very important considerations.

Embedded Systems

Higher-level languages will achieve the goals of programmer productivity, program reliability, and application portability only if they fit the target environment well. If not, productivity will disappear into work-arounds and maintenance, reliability will be lost to kludges, and portability will not exist.

DESIRED TRAITS IN C FOR MICROCONTROLLERS

The environment in which C has developed is not the same as the embedded microcontroller world. What changes or extensions or implementations of C will provide the means to adapt the language? National Semiconductor Microcontroller Division has a compiler[3] developed for the 16-bit High Performance Controller (HPC™[4]) which has led to some exploration of these issues. The needs can be summarized as:

Compatibility
Direct Access to Hardware Addresses
Direct Connection to Interrupts
Optimization Considerations
Development Environment
Re-Entrancy

Compatibility

The first consideration for any such adaptation MUST be compatibility. Any attempt to create a different language, or another dialect of C, will create more problems than using C will solve. Dialects create problems in portability, maintenance, productivity, and possibly reliability. A programmer used to working in C will be tripped up by every little gotcha in a dialect; everyone will be tripped up by a different language.

Providing extensions to the language, while maintaining compatibility and not creating a new dialect, is accomplished by using the C Pre-Processor. By carefully choosing the extensions and their syntax, the use of the preprocessor's macro capability allows them to be discarded for normal C operation with non-extended compilers. By carefully choosing their semantics, the elimination of the extensions does not render the program invalid, just less effective.

Within these considerations there should be no unnecessary additions. An extension should not be made to avoid the optimizer's having to work hard. An extension should be made only to give the user an ability he would not have without it, or to tell the compiler something it cannot figure out by itself.

Direct Access to Hardware Addresses

Access to hardware addresses is improper in computation programs, is unusual in utility programs, is infrequent in operating systems, and is the raison d'etre of microcontrollers. The normal means of accessing hardware addresses in C is via constant pointers. This is adequate, if not great, when the accesses are minimal. For example

```
struct HDLC_registers
{
 ....
};
#define HDLC_1(*(struct HDLC_registers*)
  0x01a0)
```

allows reference to a structure of HDLC device registers at address 0x01a0, but never actually creates the entity of such a structure. If a debugger were asked about HDLC__1, it would not recognize the reference. If many registers and devices are involved, it becomes a problem to be handled by the programmer, not his tools. If the debugger tries to read the source for preprocessor statements, it adds significant complexity.

Another way of doing it is

```
struct HDLC_registers
{
....
};
extern struct HDLC_registers HDLC_1;
```

and providing an external file defining the address of HDLC__1, written in assembly language. This is clean, and does create the actual entity of a structure at the address, but has required an escape to assembly language for the system (although only at the system definition level). This was the first choice at National, and retains merit because the use of macros in the definition file allows the simple creation of a table exactly like the table in the hardware manual.

What is desirable, so that the user can do his own definitions without resorting to two languages, is a means to create the entities and define the addresses of those entities, a simple means of saying that this variable (or constant) is at a specific absolute address. The syntax

```
struct HDLC_registers HDLC_1 @ 0x01a0;
```

would be excellent as an official enhancement to the language, since the @ parses like the = for an initialization (and the program shouldn't initialize a hardware register this way like a variable). However, this violates the compatibility rule for an extension, since the preprocessor cannot throw away the address following the @ character. Therefore,

```
struct HDLC_registers HDLC_1 At (0x01a0);
```

is a much more practical form as an extension—and can be made to expand to the previous (or any other) form if it is ever added as an enhancement to the language. The resulting forms

```
volatile struct HDLC_registers HDLC_1 At
  (0x01a0);
volatile struct HDLC_registers HDLC_2 At
  (0x01b0);
volatile const int Input_Capture_3 At
  (0x0182);
```

are straightforward, simple, readable, and intuitively understandable, and provide the data item definitions as desired.

Direct Connection to Interrupts

Operating systems attach to interrupts in one centralized, controlled location and manage them all in that module. Embedded systems attach to varied interrupts for a variety of purposes, and frequently the different interrupt routines are in different modules with associated routines for each purpose. It is possible to do this with another escape to assembly language, but this requires that the system be maintained and enhanced in two languages.

The solution chosen for the National compiler is to provide an identifier for functions which are to service interrupts.

These functions obviously take no arguments and return no values, so they are worth considering as special. The syntax chosen was simply

```
INTERRUPT2 timer_interrupt( )
```

although a more desirable form as an official enhancement would be

```
INTERRUPT(type)
interrupt_service_routine( )
```

because the chosen syntax can be preprocessed into whatever might be the final form. The semantics of the interrupt function were more difficult to guarantee for the future—should an interrupt function be callable by the other functions? Prohibiting it allows eventually permitting it if necessary; for improved efficiency, the National compiler does not allow an interrupt function to serve as anything other than an interrupt service routine, although one function can be attached to several interrupts.

Because the functions are special purpose, the function entry and exit code can be dedicated to interrupt entry and exit, rather than having to hide it in a separate library module. The National compiler actually generates the interrupt vector to point directly to the interrupt function; the function saves and restores the registers which it may destroy. Latency is minimized.

Interrupt response speed (latency) and interrupt system performance are important characteristics of a microcontroller. It is one thing (inconvenient or embarrassing) for a multi-MIPS machine to choke on long 9600 baud transmissions and drop a character or two because of inefficient interrupt response. It is another thing entirely—lethal, a total failure—for an embedded system's interrupt response to be so poor as to miss even one critical interrupt.

Optimization Considerations

Computer systems compete on speed (or at least MIPS ratings); compilers for them must be speed demons. Microcontrollers compete on size and costs; compilers for them must be frugal. Embedded systems are limited in their memory and different memories frequently have significantly different behavior.

The major concern of optimization comes down to code size. In most controller systems, as generated code size decreases speed usually increases. The effort in the code generation and optimization should be directed towards reducing code size. Claims for exactly how close the generated code gets to hand-written assembly code depend on specific benchmarks and coding techniques. An acceptance criterion for the National HPC compiler was code size comparison on a set of test programs. A level slightly below 1.4 times larger than assembly was reached.

In addition to the implementation of the optimization, other concerns of microcontrollers affect the way code can be generated. An example is the different forms of memory. Many controllers have memories which can be accessed by faster or shorter code. Certain variables should be placed in these memories without all the variables of a module going there (which is a linker process). There is no possible way for the optimizer to guess which variables should go there, especially in a multiple module program, so it must be told. The syntax used is

```
static BASEPAGE int important_variable;
```

because the special memory in National's HPC is the first page of RAM memory. Several other possibilities offer themselves, including using for an official enhancement

```
static register int important_variable;
```

because currently static register variables are specifically prohibited. This cannot be an extension, because the register word could not be redefined to the preprocessor. If some variables need to be accessed by fast code, and some need to be accessed by short code, and if the two were mutually exclusive, it would be desirable to have two separate extension words. Since such hardware is unlikely, the single word BASEPAGE is probably sufficient.

Additional savings can be achieved by reconsidering string literals. The ANSI C requires that each string literal is a separate variable, but in actual usage they are usually constants and therefore need not be separate nor variables. The National compiler provides an invocation line switch to indicate that all string literals (but not string variables) can be kept in ROM rather than being copied to RAM on system start-up. Such strings can be merged in the ROM space to eliminate duplication of strings.

An extension to the language to identify functions which will not be used recursively is

```
NOLOCAL straight_forward_function( );
```

which causes all local variables to be converted to static variables, which are easier and faster to access and use. If the function has no arguments, the compiler can even eliminate the use and creation of the Frame Pointer for the function, saving additional code and time.

The particular processor, the HPC, has a special form of subroutine call. Since the optimizer cannot guess across modules which functions should be called with the special form, the extension

```
ACTIVE specially_called_function(arg);
```

was added. This may or may not be appropriate for other processors, but is a good example of why the language needs careful extensions to take advantage of different processors.

One command extension was added to the language because it allows the programmer to guarantee something the optimizer cannot usually determine. The form

```
switchf(value) {...}
```

provides for a switch/case statement without a default case. When speed and size become critical, the extra code required to validate the control value and process the default is highly undesirable when the user's code has already guaranteed a good value.

The National compiler has one extension which violates the issues stated under compatibility. It remains for historical reasons. It is a command

```
loop(number) {...};
```

which produces a shortened form of the for loop, without an accessible index. This does not provide the user with any new ability, it merely allows the compiler optimizer to know, without figuring out, that the index is not used inside nor outside the loop, and can therefore be a special counted form. The preprocessor cannot produce an exact semantic equivalent for the statement. This is a perfect example of a poor extension and will eventually be eliminated.

Development Environment

Languages developed for large or expensive systems can usually depend on large systems for development support, either self-hosted or with a large system host providing cross-development tools. Microcontrollers are often price sensitive, are frequently in the laboratory or the field, and are not always supported by a large system as a development host. Personal computers provide an excellent platform for the entire suite of development tools.

National Semiconductor currently provides its compiler and associated cross-development programs on the IBM PC and clone type of computer. The software is all very portable, and can be run under VAX/VMS, VAX/Ultrix, or VAX/BSD4.2, and on the NSC 32000-based Opus add-in board for the PC running UNIX V.3, and some other versions of UNIX. The demand has been for the PC version; the PC is a very good workstation environment for microcontrollers. Other environments may be desirable, but the PC is first.

Re-Entrancy

Even with all these other considerations handled, there is a time bomb lurking in C on microcontrollers. C is a single thread, synchronous language as it is usually implemented. Since most utilities are strictly single-thread and the UNIX kernel forces itself into a single-thread, this is not a big problem for them. Embedded systems involving controllers are inherently asynchronous; the language in which they are implemented must be multi-thread without special rules and exception cases.

The passing of arguments on the stack and the returning of values in registers allow for complete re-entrancy and thus asynchronous multi-threading, but this breaks down when structures are returned. Most implementations of C use a static structure to contain the returned value and actually return a pointer to it; the compiler generates the code to access the returned structure value as required. This cannot be used in a microcontroller environment, because if an interrupt occurs during the time the static structure is being used, it cannot re-enter the function. On an operating system level such conflicts can be managed with gates, semaphores, flags, or the like, but that solution is completely inappropriate on the language level. Turning the interrupts off is similarly not a language level concept, and is impossible on a system with a NonMaskable Interrupt. Telling users not to get themselves into that situation is crippling at best, impossible to enforce, and extremely difficult to track down and correct.

The solution should be at the language level, and should allow the return of a structure without hindering re-entrancy. The author's solution, developed with National, has been to have the code calling the function provide the address of a structure in which to build the return value. Since this is frequently on the caller's stack, and is never invisibly static, the program has no hidden re-entrancy flaws.

The HPC C Compiler

The HPC C Compiler (CCHPC) is a full and complete implementation of ANSI Draft Standard C (Feb 1986) for freestanding environment. Certain additions take advantage of special features of the HPC (for the specific needs of microcontrollers). The extensions include the support of two non-standard statement types (loop and switchf), non-standard storage class modifiers and the ability to include assembly code in-line. The compiler supports enumerated types, passing of structures by value, functions returning structures, function prototyping and argument checking.

Symbol Names, both internal and external, are 32 characters. Numerics are 16-bit for **short** or **int**, 32-bit for **long**, and 8-bit for **char**, all as either **signed** or **unsigned**; floating point are offered as **float** of **double**, both using 32-bit IEEE format.

All data types, storage classes and modifiers are supported. All operators are supported, and anachronisms have been eliminated (as per the standard). Structure assignment, structure arguments, and structure functions are supported. Forward reference functions and argument type checking are supported.

Assembly code may be embedded within C programs between special delimiters.

See Table I.

CCHPC SPECIFICATIONS

TABLE I

Note: Extensions are boldface

Name length	32 letters, 2 cases
Numbers	
Integer, Signed and Unsigned	16–32 Bits
Short and Long	16 bits and 32 bits
Floating, Single and Double	32 bits and 32 bits

Data Types

Arrays
Strings
Pointers
Structures

Preprocessor

#include
#define #define() #undef
#if #ifdef #ifndef #if defined #else #elif #endif

Declarations

auto register const volatile **BASEPAGE**
static static global static function **NOLOCAL INTERRUPTn ACTIVE**
extern extern global extern function
char short int long signed unsigned float double void
struct union bit field enum
pointer to array of function returning
type cast typedef initialization

Statments

;{...} expression; assignment; structure assignments;
while ()...; do...while (); for(;;;)...; **loop( )...;**
if ()...else...; switch ()...; case:...; default:...; **switchf ( )...;**
return; break; continue; goto...; ...:

Operators

```
primary:        function( ) array[] struct_union. struct_pointer ->
unary           * & + - ! ~ ++ -- sizeof (typecast)
arithmetic:     * / % + - << >>
relational:     < > <= >= == !=
boolean:        & ^ | && ||
assignment:     = += -= *= /= %= >>= <<= &= ^= |=
misc.:          ?: ,
```

Functions

arguments: Numbers, Pointers, Structures
return values: Numbers, Pointers, Structures
forward reference (argument checking)

Library Definition **Limited-Freestanding environment**

Embedded Assembly Code

CONCLUSIONS

With the right extensions, the right implementations, and the right development environment, National is providing its customers with a C compiler tool which allows effective higher-level language work within the restrictive requirements of embedded microcontrollers. Productivity increases do not have to come at the expense of larger programs and more memory chips. No strangeness has been added to the language to cause reliability problems. Portability has been retained. Assembly language code has been eliminated as the chewing gum and baling wire trying to hold it all together, further increasing reliability and portability.

FOOTNOTES

[1] Kernighan, Brian W. and Ritchie, Dennis M., "*The C Programming Language*", Prentice-Hall 1978, Pages ix and 1.

[2] UNIX® is a registered trademark of AT&T.

[3] Produced by Bit Slice Software, Waterloo, Ontario, Canada.

ADDITIONAL INFORMATION

Datasheet

HPC Software Support Package

User's Manual

HPC C Compiler Users Manual #424410883-001

AN-593

HPC16400 A Communication Microcontroller with HDLC Support

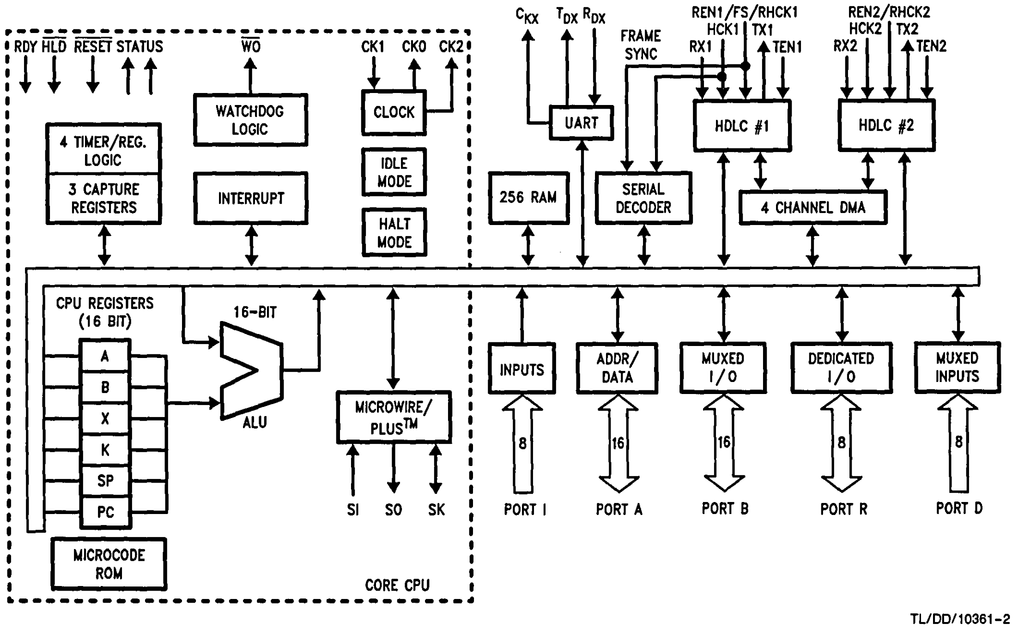
National Semiconductor
Application Note 593
Nick Burd

INTRODUCTION

The HPC16400 is a communications microcontroller for HDLC based applications and is the latest in the range of High Performance microcontrollers (HPC™) from National Semiconductor Corporation. HPC is a family of 16-bit CMOS microcontrollers which feature a common core to which are added peripherals for a specific application area. In the case of the HPC16400, these include dual HDLC channels and a four channel DMA controller which make the HPC16400 ideally suited to embedded protocol processing, such as X.25/LAPB. In addition, the HPC16400 also contains an on-chip serial decoder which allows the HDLC channels to be time multiplexed onto common transmit and receive lines as used by the ISDN (Integrated Services Digital Network) Basic Rate interface. This means that together with Nationals' ISDN line interface and COMBO™ circuits, and a software package which implements the generic ISDN protocols (Q.921 and Q.931) a complete system solution for ISDN Basic Rate applications is possible.

The HPC16400 is capable of running at a maximum clock frequency of 20 MHz, and each of its HDLC channels can operate up to a maximum 4.65 Mbps data rate. A photograph of the HPC16400 chip is shown in *Figure 1.*

This article describes the features of the HPC16400, and in particular the operation of the HDLC/DMA channels and the serial decoder. As an example of how the HPC16400 would be used in an ISDN application, an ISDN terminal is described together with the features of the ISDN software package which can be used to minimize the time and effort in developing such equipment.



TL/DD/10361-2

FIGURE 1. Block Diagram of the HPC16400

THE HPC CORE

Figure 1 shows the block diagram of the HPC16400 in which the functions within the dotted line form the HPC core which is common to all HPC family members. It can be seen that the core contains the CPU as well as several peripherals. Those functions outside the dotted line are the peripherals specific to the HPC16400.

The CPU contains a 16-bit ALU and a 16-bit accumulator which acts as the source and destination for most operations. Two 16-bit address pointer registers, B and X, are intended to be used for indirect addressing of data with auto increment and decrement of the register. The K register is used to set a limit for the B register when it is either incremented or decremented with successive execution within program loops. A specific feature of the instruction set of the HPC CPU is that conditional execution of an instruction is based on a skip structure instead of the traditional conditional branch or jump. This is best illustrated through an example using the B, K and X registers described above. The example listed in *Figure 2* swaps the contents of two areas of memory in the ranges 0x4000 to 0x4FFF and 0x5000 to 0x5FFF. A single instruction is used to load the B and K registers which define the boundaries of the lower memory area, and the X register is loaded to point to the beginning of the upper memory area. The first instruction within the loop loads the accumulator with the memory word pointed to by the X register, and the X register is then incremented. The fact that a word value has been specified here means that the X register will automatically be incremented by two. If a byte value had been specified, it would be incremented by one. The second instruction in the loop is an exchange with a conditional skip which exchanges the contents of the 16-bit accumulator with the memory word pointed to by the B register, and the B register is then incremented by two. If the new value of the B register now exceeds the value in the K register, the following jump instruction will be skipped and program execution will exit the loop. If the value of the B register is less than the K register, then the next instruction is executed and the loop is continued. Judicious encoding of the opcodes for the HPC instruction set has resulted in a very efficient implementation of common constructs such as the loop just described. The register indirect instructions are encoded as single-byte instructions as well as the short jump instruction where a six bit offset is included within the opcode. The loop described above therefore generates only three bytes of program code. In total, the HPC has 54 instructions and nine addressing modes.

```
        LD BK, #4000, #4FEE   ;load B and K with start and end of 1st. memory block.
        LD X, #5000           ;load X with start of second memory block.
LOOP:   LD A, [X+].W          ;get word from second block, increment X.
        XS A, [B+].W          ;exchange with word from first block, increment pointer,
                              ;skip if B>K.
        JP LOOP               ;do loop again

EXIT:    ↓    Continue program
```

FIGURE 2. An Example HPC Program to Swap Two Memory Areas

The HPC core contains several peripheral features. The MICROWIRE/PLUS™ is an inter-chip serial communication port which consists of an 8-bit shift register and a clock. Writing data to the microwire port when configured as a master causes the data to be loaded into the shift register and eight clock pulses generated to shift the data out. At the same time, these clock pulses can be used to clock data in from a microwire slave device such as the ADC0834 A/D converter or the NMC93C46 EEPROM.

The HPC core also contains a number of timers. A purpose of one of these timers, T0, is to provide a means for accurate time interval measurements, and when configured in this mode, it is associated with up to three capture registers which can be triggered by external interrupt inputs. Timer T1 provides a dual function as it can operate as a normal timer, or its registers can be used as two of the capture registers for T0. The timer T0 also drives the Watchdog™ logic which causes the Watchdog output to trigger whenever it is not serviced before a timeout of T0. The remaining two timers can be used to generate a variety of timing outputs.

Interrupt logic provides enabling circuitry for the numerous sources of interrupt on the HPC, and an interrupt pending register eases the processing of multiple interrupts. The HPC can be placed into one of two power saving modes by programming the Processor Status Word (PSW) register and the Halt Enable register. In the Halt mode, all processor activities, including the clock and timers, are stopped thereby reducing the power requirements of the HPC to a minimum. Recovery from the Halt Mode can either be from a Reset or from the NMI. In Idle mode, all processor activity apart from the on-board oscillator and timer T0 is stopped so that recovery from the Idle mode can be achieved with the timer T0 overflow as well as the reset or NMI functions as in the Halt mode (except that in the Halt mode recovery is not immediate as the oscillator will take tome to stabilize).

HPC MEMORY

All functions on the HPC chip are memory mapped. The on-chip peripherals, core registers, and on-chip user RAM (16-bit) occupy an address area between 0 and 0x1FF as shown in the memory map of *Figure 3a*. The area of user on-chip RAM in the range 0–0xBF is in the BASEPAGE (0–0xFF) of the address space, and in addition to being used as general purpose storage locations for variables, the indirect addressing mode of the HPC allows memory words in this area to be used as pointers containing the effective address of the operand. This allows many additional pointers to be created in addition to the B and X register and significantly eases the programming of many tasks.

The memory requirements in telecom applications are generally large for both program and data areas, and so the HPC16400 does not have a single-chip configuration with on-chip ROM. Instead, a 16-bit multiplexed address and data bus is brought external to the chip and is used to add program memory and additional data memory to the system in the address range 0x200 to 0xFFFF.

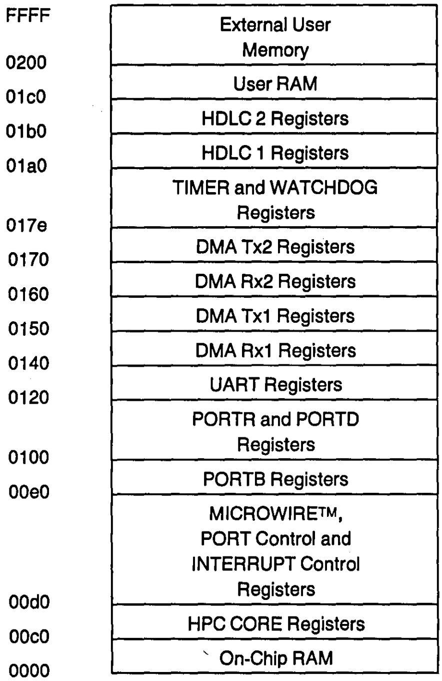
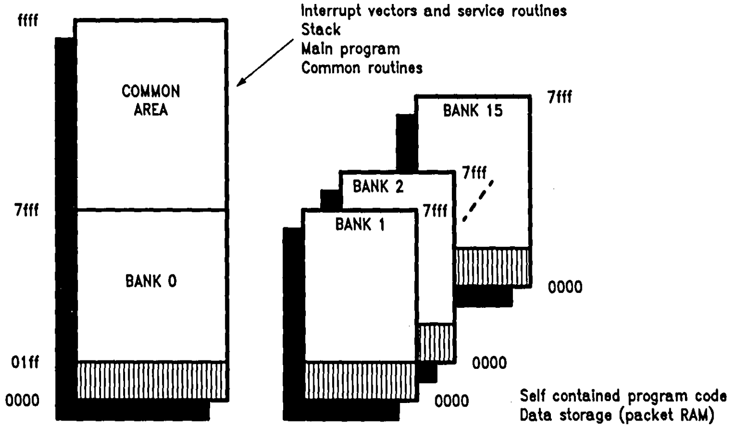


FIGURE 3a. The Basic 64k Memory Map of the HPC16400

The 64 kbyte address space can be expanded further by the use of bank switching. Four lines from Port B may be used to select one of sixteen banks of 32 kbytes in the address range 0–0x7FFF as shown in *Figure 3b*. In this way, the upper 32 kbytes of memory are common to all of the banks and allows a program in one bank to jump or call subroutines in other banks via this common area where the banks can be safely switched (see reference 1 for a more in-depth discussion of bank switching on the HPC). The common memory also provides storage area for global variables and stack locations when operating in a bank-switched environment. The total memory addressing capability of the HPC16400 amounts to just over 500 kbytes as the section of on-chip RAM in the range 0–0x1FF is common to all banks.



TL/DD/10361-3

FIGURE 3b. HPC Extended Memory Addressing by Bank Switching

When the HPC fetches program from memory, it does so one byte at a time because the opcode encoding is byte oriented. This allows the HPC to be configured with either 16-bit external memory, 8-bit external memory for more cost sensitive applications, or a mixture of both. When operating with 16-bit memory, the HPC can access both odd and even bytes, and words on an even boundary. In 8-bit mode the HPC makes only byte accesses to external memory, and although the registers in the BASEPAGE memory of the HPC are 16 bits, they may also be addressed individually as high and low bytes thereby enabling the 16-bit architecture of the CPU to be used.

Selection of 8- or 16-bit bus mode for the HPC is achieved on reset of the processor when the "high byte enable" control line is sampled by the CPU. If this line is detected in a high state, the HPC enters 8-bit mode. However, if the line is detected as high impedance, as a result of it being used as a control output to select low and high 8-bit memory banks, then the HPC enters 16-bit bus mode.

THE HDLC AND DMA CHANNELS

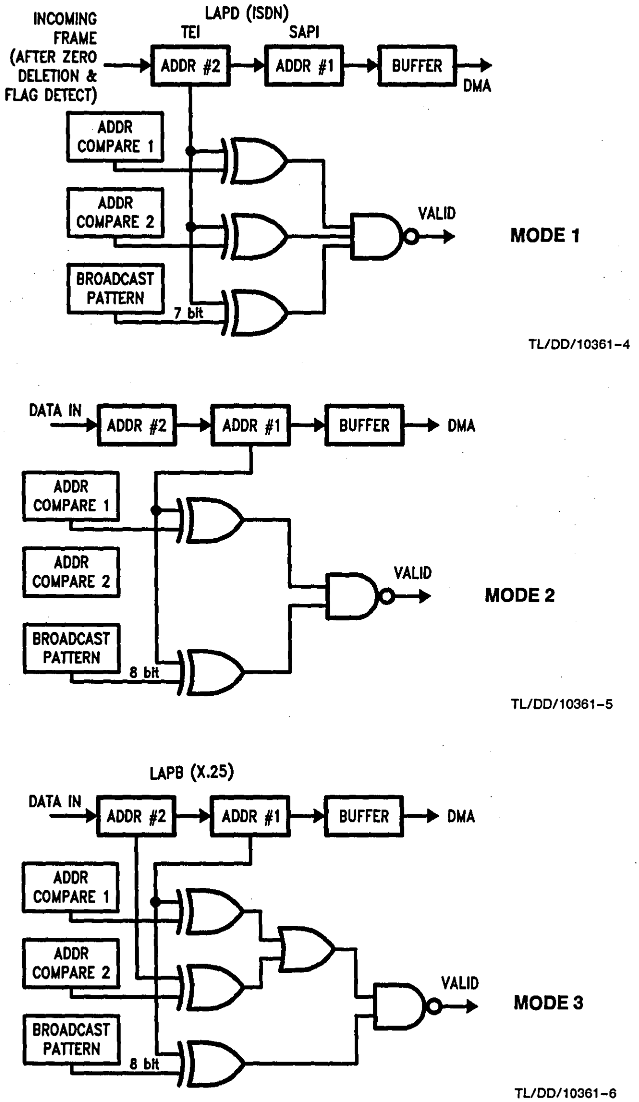
The HPC16400 contains two identical on-chip HDLC channels, each capable of transmitting and receiving HDLC frames transparently to the operation of the CPU. The format of an HDLC frame is shown in *Figure 4*. The frame is delimited by an identical opening and closing flag which is a unique bit pattern consisting of a zero followed by six consecutive ones and then a final zero. This pattern must not occur anywhere else within the frame and is guaranteed by a zero insertion mechanism which, after the transmission of five consecutive ones in the data stream between flags, will insert a zero before continuing to transmit data. A reverse procedure is adopted at the receiver to delete the additional zeroes. Immediately following the opening flag is an address field which identifies which equipment on the network is to receive the frame. The control field contains information, such as handshake control, which is used to control the flow of frames between communicating devices. This is followed by the application specific data, a frame check sequence which validates the integrity of the frame with a cyclic redundancy check (CRC) code, and the closing flag. The HPC HDLC channels provide automatic framing functions such as opening and closing flag insertion and deletion, zero bit insertion and deletion (also known as bit-stuffing), CRC16 or CCITT implementations of CRC checking, and abort sequence transmission and recognition. The abort sequence in this case is a modified flag consisting of a zero followed by seven ones. In addition, the transmitters can be programmed to generate flags, abort sequences, or just idle (transmitting consecutive ones) between the transmission of consecutive frames.

first byte					**last byte**
FLAG	ADDRESS	CONTROL	DATA	CRC	FLAG
1	1 or 2	1	Typically < 1024	2	1

BYTES

FIGURE 4. The HDLC Frame Format

A feature which helps to reduce the CPU overhead in protocol processing is the address recognition logic. Each channel has two address recognition registers that can be programmed with a byte which can be compared in a number of different ways with the first two bytes received by an HDLC channel. The different comparison modes are intended to cope with a range of different communication network addressing modes. *Figure 5* shows the logical operation of three of the four possible modes. In mode one, the second byte received after the opening flag of the frame is compared with both address registers and a seven bit broadcast address pattern (0x7F). If any of the registers match the incoming address, then the HDLC channel will continue to receive the complete frame. If no match is detected, the HDLC channel will stop receiving the frame, discard the address already received, and start to look for the opening flag of the next frame. This particular address recognition mode is useful in ISDN communications because the second byte received will be the address of the terminal equipment, such as a telephone or perhaps a PC, on the ISDN network.



TL/DD/10361-4

TL/DD/10361-5

TL/DD/10361-6

FIGURE 5. The Address Recognition Logic for the HDLC Receivers

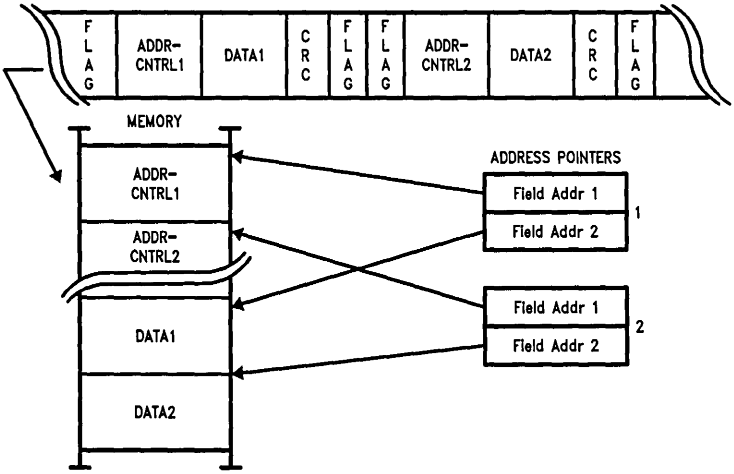
Mode two matches the first byte received with the first address register and an 8-bit broadcast address which could be used in X.25/LAPB applications. Mode three compares a 16-bit address field so that the contents of the first comparison register must match the first address byte received, and the contents of the second comparison register must match the second address byte received. Or, if the first byte corresponds to the 8-bit broadcast pattern, an address match will also be signalled to the CPU. The last mode, Mode zero, is the "transparent mode" in which all frames are received by the HDLC controller regardless of the address field contents. This mode would be used, for example, in a device which had to gather all information from the communications network and compute statistics about its communications loading.

Both HDLC channels are capable of implementing bit oriented protocols, such as IBMs SDLC, by programming the number of bits to be transmitted in the last byte of the information field. Further flexibility is achieved with a bypass mode which disables all of the HDLC framing functions allowing designers to implement their own byte-oriented synchronous protocols.

As mentioned earlier, all programmable features of the HPC are memory mapped, so the HDLC registers are mapped to an area of on-chip RAM above the BASEPAGE section in the range 0x1A0 and 0x1B8. Each HDLC channel has an identical set of registers, and each set contains receiver status, control, address comparison, and error status registers. In addition, there are two global registers which handle the enabling and servicing of interrupts from the HDLC channels. An interrupt can be generated whenever an HDLC channel signals an "End of Message" (EOM) which indicates that an HDLC frame has just been received or an HDLC frame has just finished being transmitted. Should a transmitter or receiver generate an EOM before the previous EOM has been serviced, then an overrun interrupt may be generated. All of these interrupt sources have a single interrupt service vector, and so the global registers contain bits which allow the source of the interrupt to be uniquely identified. Additional error conditions, such as reception of a bad CRC, reception of an abort sequence, or a framing error, cause bits to be set in the error status register which may also generate an interrupt, although this may lead to the generation of multiple interrupts. A more straight-forward approach would be to test the condition of the error status register once an EOM interrupt has been received.

The HPC16400 contains an on-chip four channel DMA controller. The operation of the DMA controller is closely linked to the HDLC channels because they are responsible for interfacing them to the memory. Hence, as each byte is received by an HDLC channel, it signals the DMA controller which requests and gains control of the processor bus and writes the received byte to a predetermined area of memory. Similarly, when an HDLC channel is transmitting a frame, it requests data from the DMA controller which transfers a byte from an area in memory to the HDLC channel. During DMA accesses the CPU loses control of the memory bus. However, for the HPC16400 running at 20 MHz, the CPU bus occupancy 1s only expected to decrease by 10% for an aggregate HDLC data rate of 2 Mbps. For typical Basic Rate ISDN applications the decrease is expected to be less than 2%.

The DMA channels contain several addressing features which allow convenient transmit and receive buffers to be created in memory. Each DMA channel supports a split-frame mode which allows the transmitted or received frame to be split into two sections with each section being stored in a different area of memory. In HDLC, it may be convenient to have all the address and control fields in one area of memory, and all the information fields in another. (The CRC and flag fields are stripped off or appended by the HDLC channels, and so are not present in the memory area.) In the DMA receiver, there are two pairs of address pointers, each pair pointing to the two sections of the same frame as shown in *Figure 6*. As the HDLC controller starts to receive data, the DMA channel places the first received byte in the memory pointed to by the first address pointer, and the pointer is then incremented. This continues until the number of bytes for the first segment, which can be programmed up to a maximum of 7 bytes in the DMA receiver control-status register, has been reached, at which point the contents of the second address pointer becomes the destination for the remainder of the received frame.



TL/DD/10361-7

FIGURE 6. Split-Frame Operation for HDLC/DMA Receiver

For the DMA channel which supports the HDLC transmitter, each pair of registers contains a single pointer and a byte counter which holds the number of bytes to be transmitted, as illustrated in *Figure 7*. When the split-frame mode is not used, each pair of registers in the transmit DMA, and each address pointer in the receive DMA, refers to a separate complete frame. This means that the HDLC receiver can receive four frames before the DMA address pointer registers need to be updated, provided the EOM is serviced after each frame to prevent an overrun interrupt.

In the previous section, the extended memory configuration of the HPC16400 using bankswitching was described. The DMA channels are capable of taking full advantage of this extended memory by a programmable field in the control-status registers whose value is written to the external bank-switch control lines during a DMA cycle. This allows the extended memory banks to be used for storing frame information.

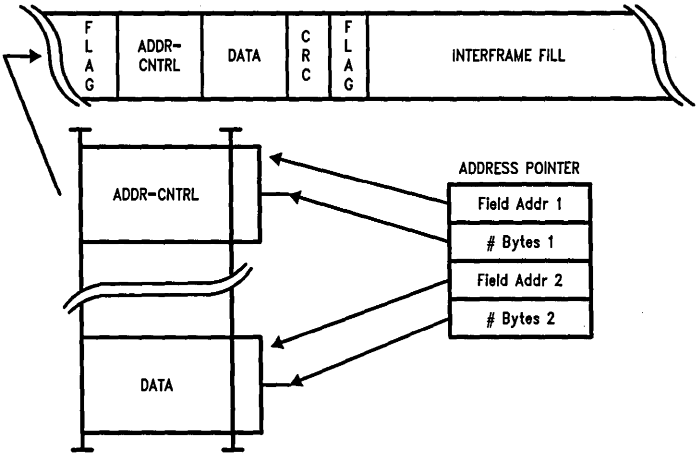
The DMA controller is only capable of taking control of the processor bus when the CPU has finished executing the current instruction. When the HPC16400 executes long instructions, such as the Multiply or Divide instructions, and the HDLC channels are being used at very high data rates (in excess of 2.2 Mbps with a 20 MHz HPC), it may be possible that the DMA cannot gain control of the processor bus in time to service the HDLC channels. In this situation, the receiver is forced to overwrite the last byte received and a receiver overrun is flagged in the error status register. When this occurs during transmission, the transmitter no longer has any valid information to send and so it transmits an abort character and sets a transmitter underrun bit in the error status register. Programming the HDLC/DMA controllers is relatively straightforward, both for their initialization and interrupt servicing. Because the DMA controllers have two sets of registers, it means that the pointers to the next message to be received or transmitted can be set up while reception or transmission is in progress, thereby maximizing the throughput of the HDLC channels.

THE SERIAL DECODER—BASIC RATE ISDN AS AN EXAMPLE

As already described, the HDLC channels of the HPC16400 can be used in general purpose communications and networking applications. To enhance their capabilities, and provide on-chip support for ISDN, a serial decoder has been implemented to time division multiplex the two HDLC channels onto common transmit and receive lines.

Each HDLC channel can be enabled and disabled both internally by the serial decoder, and externally by individual receiver and transmitter enable pins. The internal enable signals are generated by the serial decoder according six time division multiplexing (TDM) formats. The framing of these TDM formats, or modes, is synchronized by an externally generated frame sync. pulse which will normally be derived from an external clock signal used to clock the HDLC channels. With these inputs, the serial decoder generates the internal enable signals for the HDLC channels at the correct time within the frame according to mode that has been selected. The serial decoder can also be programmed to generate enable signals for the HDLC channels based on combinations of both the external enable signals and those generated internally by the serial decoder, thereby giving the designer a wide choice of possibilities.

As an example of the use of the serial decoder, we shall look at Basic Rate ISDN. Basic Rate ISDN specifies that a terminal equipment, such as a telephone or computer, should have two general purpose B channels (Bearer channels) for voice data or perhaps computer packet switched data, and a D channel which is used specifically for control of the ISDN network, such as setting up a call to another user. These 2B+D channels are time division multiplexed within a 125 μs frame on a bus which interconnects functional blocks within a piece of equipment. The time slot for each B channel is the transmission time for 8 bits at a data rate of 64 kbps, and the D channel time slot is 2 bits at 16 kbps.



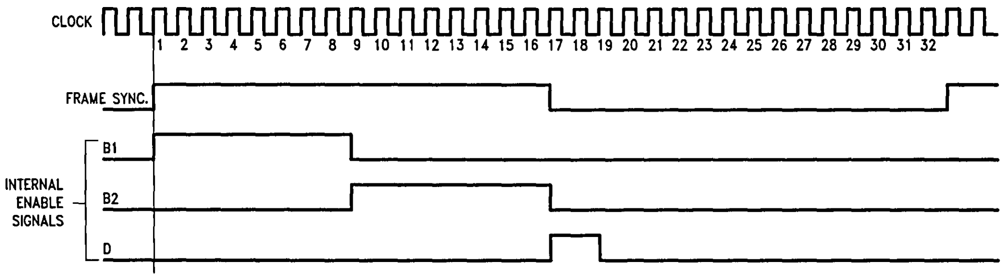
TL/DD/10361-8

FIGURE 7. Split Frame Operation for HDLC/DMA Transmitter

The overall scheme is shown in *Figure 8*. Now the HPC16400, having two HDLC channels, could be set up so that one HDLC channel is a B channel, and the other HDLC channel is the D channel. The serial decoder therefore has to be programmed so that its mode corresponds to the format shown in *Figure 7*, and that the enable signals are chosen internally such that the D time slot is assigned to one of the HDLC channels, and the correct B channel is assigned to the other HDLC channel. The remaining B channel could be occupied by any other device capable of generating a 64 kbps data stream within its time slot, such as a voice COMBO. The frame sync. signal and the HDLC clock will be generated externally to the HPC16400, typically by the ISDN line interface circuit as described in the next section.

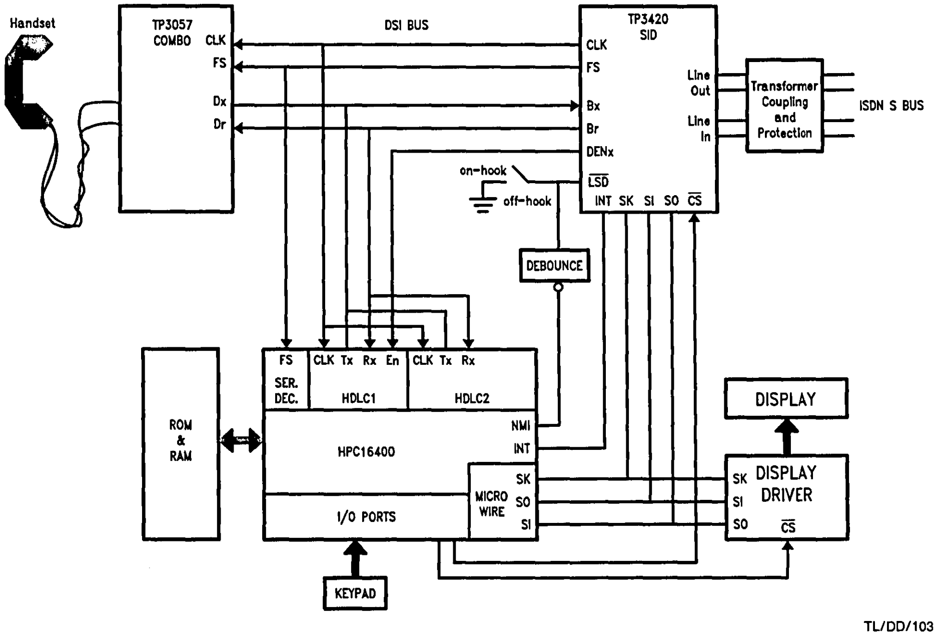
AN ISDN TELEPHONE

Figure 9 shows the block diagram of an ISDN telephone. The three main components of the system are the HPC16400 microcontroller, the TP3420 "S" Interface Device (SID) which is the line interface to the ISDN subscriber (S) link, and the TP3057 COMBO which provides the interface to the system for a handset. The inter-chip data bus, whose timing format was used as an example in the previous section, is called the Digital System Interface (DSI) bus, and combines the B and the D channels into common transmit and receive lines. Hence, the HDLC Tx outputs are tied together with the Dx output of the COMBO and are input to the SID DSI input pin Bx, and the HDLC Rx pins are combined with the Dr input from the COMBO and are driven by the SID DSI output pin Br. The SID, when configured in master mode, generates the frame sync. and clock signals which are derived from the received signal on the S bus. These signals are both connected to the HPC16400 and the COMBO so that the correct multiplexing format for the DSI bus as shown in *Figure 9* can be achieved. An additional output from the SID, DENx, indicates the presence of D channel bits on the DSI bus, and is used to enable the HDLC channel of the HPC16400 which has been assigned to handle the D channel communications, in this case HDLC channel 1.



TL/DD/10361-9

FIGURE 8. The Serial Decoder Format for ISDN



TL/DD/10361-10

FIGURE 9. Block Diagram of an ISDN Telephone

The SID is a programmable device with various modes and functions that conform to the CCITT I.430 specification for the physical layer of ISDN. Programming of the SID is achieved with the MICROWIRE/PLUS interface which is also used to drive the display for the telephone using a COP472-3 liquid crystal display controller. Selection of either the SID or the display driver is achieved with port lines from one of the general purpose I/O ports on the HPC16400 so that the chip selects for each device are software driven.

The Halt power saving mode can be used whenever the telephone is not active. This is indicated by the on/off hook signal from the handset which is interfaced to the NMI input of the HPC. When a telephone conversation is finished and the handset placed on-hook, the HPC can be put into Halt mode by software. When the handset is subsequently picked up for another call, and so goes off-hook, it will generate an NMI which will wakeup the HPC.

THE ISDN SOFTWARE

The control of end-to-end communications in a telephone system can be a complex procedure. Many things have to be taken into consideration, such as procedures for establishing a call, dial-plans, disconnecting calls, and so on. All of these procedures amount to the protocols which are part of ISDN. In particular, the control protocols for ISDN are those which are used on the D channel to establish and disconnect physical links between two (or more) users of the telephone network. *Figure 10* shows the three protocol layers of ISDN according to the ISO seven layer reference model for Open Systems Interconnection.

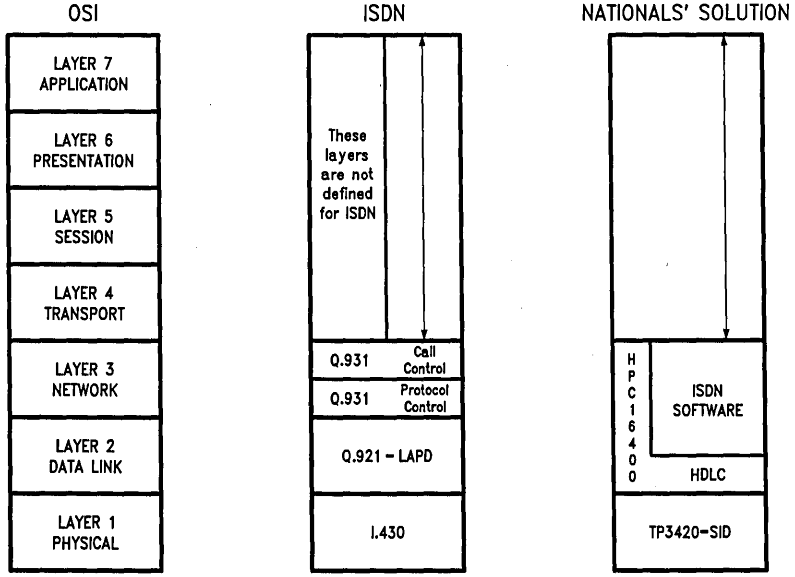
At the physical layer, the CCITT standard I.430 is used to specify the requirements of the ISDN S-Bus interface device. The TP3420 SID conforms to this specification, and in fact exceeds it in some aspects such as its ability to drive longer cable lengths. (The DSI bus is not part of this standard as it refers to the equipment side of the network.)

The data link layer protocol is responsible for the safe delivery of frames across the network. Here, ISDN uses the CCITT standard Q.921 which is more commonly known as LAPD, or "Link Access Protocol on the D Channel". LAPD defines the "HDLC" frame format and a set of procedures to control the flow of information on the network, and recovery from errors. It is similar to the LAPB link access protocol used in X.25 and true HDLC networks, but defines an expanded set of procedures to cope with communications on a telephone network instead of a typical computer network.

Finally, in Layer 3, the CCITT Q.931 standard specifies a series of procedures for establishing, maintaining, and disconnecting calls between users on the network. Part of these services are application dependent, so in order to make the ISDN standard generic as possible, the Layer 3 is split into two parts. The generic part of Layer 3 executes the "protocol control procedures" and the application dependent part performs the "Call Control Procedures".

Figure 10 also shows how the ISDN protocols are mapped onto the hardware components. The SID is the Layer 1 device and the HPC16400 provides hardware support, by means of its HDLC channels, for the Layer 2 protocol. The clear boundary between the Layer 1 and Layer 2 devices results in a well structured system architecture, with the DSI bus creating the physical interface between these two layers. The remaining parts of Q.921 and Q.931 are implemented as a software package which includes drivers for the SID and HDLC/DMA channels, and tools which aid the debugging of application tasks that interface to the software at the Layer 3 call control level.

Within the software, the individual layers and drivers are implemented as tasks which run under a multi-tasking executive. The operation of the executive has been optimized to work with layered tasks, and includes features such as a mail manager, timer manager, and memory manager. The entire software package is written in "C" so that application tasks can be developed, run with the layer software (excluding the drivers), and debugged on a PC before being ported to the target hardware.



TL/DD/10361-11

FIGURE 10. Nationals' Solution to ISDN

CONCLUSIONS

The HPC16400 is a versatile high performance 16-bit CMOS microcontroller for embedded communications applications. Its fast CPU together with dual HDLC channels provides an ideal platform for implementing proprietary or standard communication protocols that use the HDLC framing structure.

REFERENCES

1. *"Expanding the HPC Address Space"*, National Semiconductor Application Note 497.
2. *"Intuitive ISDN—An ISDN Tutorial"*, National Semiconductor Application Note 492.

AN-603

Signed Integer Arithmetic on the HPC™

National Semiconductor
Application Note 603
Raj Gopalan

This report describes the implementation of signed integer arithmetic operations on the HPC. HPC hardware support for unsigned arithmetic operation. In order to support signed integer arithmetic operations on the HPC, the user can represent negative numbers in two's complement form and perform the signed arithmetic operations explicitly through software.

The following signed integer arithmetic routines are implemented in the package:

Multiplication:

16 by 16 yielding 16-bit result
32 by 32 yielding 32-bit result

Division:

16 by 8 yielding 16-bit quotient and 16-bit remainder
32 by 16 yielding 16-bit quotient and 16-bit remainder
32 by 32 yielding 16-bit quotient and 16-bit remainder

Addition:

16 by 16 yielding 16-bit

Subtraction:

16 by 16 yielding 16-bit

Comparison:

16 by 16 for greater to, less than or equal to.

REPRESENTATION OF NEGATIVE NUMBERS:

For binary numbers, negative numbers are represented in two's complement form. In this system, a number is positive if the MSB is 0, negative if it is 1.

The decimal equivalent of two's complement number is computed the same as for an unsigned number, except that weight of the MSB is −2**n − 1 instead of +2**n − 1. The range of representable numbers is −(2**n − 1) through +(2**n − 1 − 1).

The two's complement of a binary number is obtained by complementing its individual bits and adding one to it.

The advantage of representing a negative number in two's complement form is that addition and subtraction can be done directly using unsigned hardware.

```
          .title        SIMUSL
          .sect         code,rom8,byte,rel
;Signed multiply (16 by 16)
;         B             Multiplicand
;         A             Multiplier
;         X ;A          return
;
          .public signed_mult_16
          .local
signed_mult_16:
          st            a,0.w
          mult          a,b                                ;do unsigned multiplication.
          sc
          ifbit         7,(1).b                            ;if multiplier is negative
          subc          x,b
          sc
          ifbit         7,(B+1).b                          ;if multiplicand is negative
          subc          x,0.w
$exit:
          ret

          .endsect
```

MULTIPLICATION

Method 1:

Signed multiplication can be achieved by taking care of the signs and magnitudes of the multiplicand and multiplier separately.

Perform the multiplication on the magnitudes alone.

The sign of the result can be set based on the signs of the multiplier and the multiplicand.

Method 2:

This method does not require finding the magnitude of the operands. Multiplication can be done using unsigned hardware on the two's complement numbers. The result will be signed based on the signs of the operands.

The algorithm is as follows:

Step 1. Result = op1 * op2

Step 2. If op1 < 0 then subtract op2 from upper half of the result.

Step 3. If op2 < 0 then subtract op1 from upper half of the result.

Now the Result will yield the correct value of the multiplication on two's complement numbers.

Method 3:

By sign extending the multiplier and multiplicand to the size of the result one can always obtain the correct result of signed multiplication using unsigned multiplication.

```
          .title      SIMULL
          .sect       code,rom8,byte,rel
;Multiply (Signed or Unsigned are the same)
;32 bit
;         K:A         Multiplicand
;         -4:6[SP]    Multiplier
;         K:A         return
;
          .public multiply_32
          .local
multiply_32:
          push        x                        ;(Argument now at -6:8[SP])
          st          a,0.w
          ld          a,k                      ;Multiply hi reg* lo stack
          mult        a,-8[sp].w
          x           a,0.w                    ;hold, retrieve lo reg
          push        a                        ;(argument now at -8:10[SP])
          mult        a-8[sp].w                ;Multiply lo reg* hi stack
          add         0.w,a                    ;add into hi partial
          pop         a                        ;(Argument now at -6:8[SP])
          mult        a,-8[sp].w               ;Multiply lo reg* lo stack
          add         x,0.w                    ;add in hi partial
          ld          k,x                      ;Position
          pop         x                        ;Restore
          ret

          .endsect
```

DIVISION

Similar to multiplication method 1, one can perform the division on the magnitudes of the dividend and divisor. The sign of the quotient can be set based on the signs of the dividend and the divisor. The sign of the remainder will be same as the dividend.

```
          .title        SIDVSS
          .sect         code,rom8,byte,rel


;Division & Remainder
;16,8 bit (signed only, unsigned uses inline code)
;         A             Dividend
;         -4[SP]        Divisor
;         A             return
;
          .public signed_divide_8,signed_remainder_8
          .public signed_divide_16,signed_remainder_16
          .local
signed_divide_8:
          jsr           $shared_8                           ;Uses shared routine
          ret
;
signed_remainder_8:
          jsr           $shared_8                           ;Uses shared routine
          ld            a,k                                 ;Return remainder
          ret
;
$shared_8:
          ifgt          a,#0x7f
          or            a,#0xff00
          st            a,k                                 ;Get arguments
          ld            a,-6[sp].w
          ifgt          a,#0x7f
          or            a,#0xff00
          jp            $shared
;
signed_divide_16:
          jsr           $shared_16                          ;Uses shared routine
          ret
;
signed_remainder_16:
          jsr           $shared_16                          ;Uses shared routine
          ld            a,k                                 ;Return remainder
          ret
;
$share_16:
          st            a,k                                 ;Get arguments
          ld            a,-6[sp].w
$shared
          ifeq          a,#0
          ret                                               ;division by zero
          push          x
          ifgt          a,#0x7fff
          jp            $unknown_negative                   ;unknown/negative
          x             a,k
          ifgt          a,#0x7fff
          jp            $negative_positive                  ;negative/positive
          div           a,k                                 ;Positive/positive is plus,plus
          jp            $positive_positive
```

```
$unknown_negative:                                          ;Unknown/negative
          comp          a
          inc           a
          x             a,k
          ifgt          a,#0x7fff
          jp            $negative_negative                  ; negative/negative
          div           a,k                                 ;Positive/negative is minus,plus
          comp          a
          inc           a
$positive_positive:
          ld            k,x
          jp            $exit
$negative_positive:                                         ;Negative/positive is minus,minus
          comp          a
          inc           a
          div           a,k
          comp          a
          inc           a
          jp            $negate_remainder
$negative_negative:                                         ;Negative/negative is plus,minus
          comp          a
          inc           a
          div           a,k
$negate_remainder:
          x             a,x
          comp          a
          inc           a
          st            a,k
          ld            a,x
$exit:
          pop           x
          ret

          .endsect
          .title        SIDVLS
          .sect         code,rom8,byte,rel

;Division & Remainder
;Signed 32 by 16 divide
;         X;A           Dividend
;         K             Divisor
;         X,A           return (remainder and quotient)
;
          .public signed_div_32
          .local
signed_div_32:
          sc
          ifeq          k,#0
          ret                                          ;Divide by zero, set carry and return
$shared_signed:
          ifbit         7,x+1.b
          jp            $negative_dividend
          jsr           $process_divisor               ;Skipping return
          ret                                          ;+/+=+,+
$negate_quotient:
          comp          a
          inc           a
          ret                                          ;+/-= -,+
$negative_dividend;
          comp          a
          add           a,#01
          x             a,x
          comp          a
          adc           a,#0
          x             a,x
          jsr           $process_divisor               ;skipping return
          jsr           $negate_quotient               ;-/+=-,-
$negate_remainder:                                     ;-/-=+,-
          x             a,x
          comp          a
          inc           a
          x             a,x
          ret
$process_divisor:
          ifbit         7,k+1.b
          jp            $negative_divisor
          divd          a,k                            ;?/+
          ret
$negative_divisor:
          x             a,k
          comp          a
          inc           a
          x             a,k
          divd          a,k                            ;?/-
          retsk

          .endsect
          .title        SUDVLL
          .sect         code,rom8,byte,rel

;Division & Remainder
;Signed 32 by 32 Divide
;         K:A           Dividend
;         -4:6[SP]      Divisor
;         K:A           return
;
;Stack frame as built and used consists of
;top:
;         0, initial subtrahend hi /dividend shifts into subtrahend
;         0, initial subtrahend lo /becomes remainder
;         k, dividend hi /dividend shifts into subtrahend, and
;         a, dividend lo /quotient shifts into dividend
;         b preserved
;         x preserved
;         return address
;         sp-4-12, divisor hi
;         sp-6-12, divisor lo
;Sign flag (0 = negative, 1 = positive, for test sense at exit)
;bit 0, divisor sign (1 = negative)
;bit 1, dividend sign (1 = positive)
;Inc of flag causes bit 1 = (bit 1 xor bit 0) by carry/nocarry out of bit 0
;so that two positives (010) or two negatives (001) indicate a positive
;quotient (011 or 010) in bit 1. Bit 1 always indicates sign if remainder.
;Operation is indicated by bit 3 of the flag, 1 = remainder.
;
          .public signed_divide_32, signed_remainder_32
          .public unsigned_divide_32, unsigned_remainder_32
          .local
signed_divide_32:
          ld            1.b,#0x02
          jp            $shared_signed
;
signed_remainder_32:
          ld            1.b,#0x0a
$shared_signed:
          ifbit         7,k+1.b                             ;Check dividend
          jsr           $negate                             ;Negate dividend and note sign
          ifbit         7,-6+3[sp].b                        ;Check divisor
          jp            $negate_divisor
          jmp           $shared
;


$negate_divisor:
          x             a,-6[sp].w                          ;Negate divisor and note sign
          comp          a
          add           a,#1
          x             a,-6[sp].w
          x             a,-4[sp].w
          comp          a
          adc           a,#0
          x             a,-4[sp].w
          sbit          0,1.b
          jp            $shared
;
unsigned_divide_32:
          ld            1.b,#0x02
          jp            $shared
;
unsigned_remainder_32:
          ld            1.b,#0x0a
```

```
$shared:
          push          x                                   ;Preserve registers
          push          b
          ld            b,sp                                ;Place dividend, becomes quotient
          push          a
          push          k
          ld            x,sp                                ;Set subtrahend, becomes remainder
          clr           a
          push          a
          push          a
          ld            k,#-18                              ;Access divisor argument
          add           k,sp
          ld            a,[k].w
          or            a,2[k].w
          ifeq          a,#0
          jmp           $zero                               ;division by zero
          ld            0.b,#32                             ;Set counter
$loop:
          ld            a,[b].w                             ;Shift Dividend:Quotient
          shl           a
          xs            a,[b+].w
          nop
          ld            a,[b].w
          rlc           a
          xs            a,[b-].w
          nop
          ld            a,[x].w
          rlc           a
          x             a,[x+].w
          ld            a,[x].w
          rlc           a
          x             a,[x-].w
          ifc
          jp            $subtract                           ;Carry out - dividend divisor
          sc                                                ;Check for dividend divisor
          ld            a,[x+].w
          subc          a,[k].w
          ld            a,[x-].w
          subc          a,2[k].w
          ifnc
          jp            $count                              ;dividend divisor
$subtract:
          ld            a,[x].w                             ;Subtract out divisor (c is set)
          subc          a,[k].w
          x             a,[x+].w
          ld            a,[x].w
          subc          a,2[k].w
          x             a,[x-].w
          sbit          0,[b].b                             ;Set quotient bit
$count:
          decsz         0.b                                 ;Count 32 shifts
          jmp           $loop
$zero:
          pop           k                                   ;Get Remainder and/or Quotient
          pop           a                                   ;and clear working off stack
          pop           x
          pop           b
          ifbit         3,1.b
          jp            $exit                               ;want remainder, have it
          ld            a,b                                 ;Want Quotient
          ld            k,x
          inc           1.b                                 ;Divisor's sign Xors Dividend's
```

```
$exit:
          pop           b                                    ;Restore registers
          pop           x
          ifbit         1,1.b
          ret                                                ;positive result
$negate:
          comp          a                                    ;Negate K:A
          add           a,#1
          x             a,k
          comp          a
          adc           a,#0
          x             a,k
          rbit          1,1.b                                ;Note sign (for entrance)
          ret

          .endsect
```

ADDITION

Two's complement numbers can be added by ordinary binary addition, ignoring any carries beyond the MSB. The result will always be the correct sum as long as the result doesn't exceed the range.

If the result is the same as for the subtrahend, then overflow has occurred.

```
          .title        SIADD
          .sect         code,rom8,byte,rel
;Signed add (16 by 16)
;         A             Operand1
;         B             Operand2
;         Carry         Return
          .public sign_add
          .local


sign_add:
          ld            0.b,#00
          ifbit 7,(A+1).b
          inc           0.b
          ifbit 7,(B+1).b
          inc           0.b

          ;if bit 0 of 0.b = 1 then op1 and op2 have different sign
          ;if bit 0 of 0.b = 0 then op1 and op2 sign are same
          ;then if bit 1 of 0.b = 0 both operands are positive
          ;else both operands are negative.

          add           a,b                                ;Perform unsigned addition
          rc
          ifbit 0,0.b                                      ;both operands are different sign
          ret
          ifbit 1,0.b                                      ;both op1 and op2 are negative
          jp $negatives
$positives:                                                ;both op1 and op2 are positive
          ifbit 7,(A+1).b                                  ;if result sign is negative then
                                                            set overflow bit
          sc                                               ;overflow
          ret
$negatives:
          ifbit 7,(A+1).b                                  ;if sign bit of result is
                                                            negative, then no overflow


          ret
          sc                                               ;overflow
$exit:
          ret

          .endsect
```

SUBTRACTION

Subtraction can be achieved by negating the subtrahend and perform the addition operation.

Overflow can be detected as mentioned before by checking the signs of minuhend and the negation of the subtrahend and that of the sum.

```
          .title        SISUB
          .sect         code,rom8,byte,rel


;Signed subtract (16 by 16)

;         B             Operandl
;         A             Operand2
;         Carry,A       Return
          .public sign_sub
          .local
sign_sub:
          ld            0.b,#00                             ;initialize sign flags
          ifbit         7,(B+1).b
          inc 0.b
$negate_A:
          comp A
          inc A
$ngative_comp_A:
          ifbit 7,(A+l).b
          inc           0.b
          ;if bit 0 of 0.b = 1 then opl and op2 have different sign
          ;if bit 0 of 0.b = 0 then opl and op2 sign are same
          ;then if bit 1 of 0.b = 0 both operands are positive
          ;else both operands are negative.
          add A,B                                           ;Perform unsigned addition
          rc
          ifbit 0,0.b                                       ;both operands are different sign
          ret
          ifbit 1,0.b                                       ;both opl and op2 are negative
          jp $negatives
$positives:                                                 ;both opl and op2 are positive
          if bit 7, (A+l).b                                 ;if result sign is negative then
                                                             set overflow bit
          sc                                                ;bit 0 of byte 0.b is set to
                                                             indicate overflow
          ret
$negatives:
          ifbit 7, (A+l).b                                  ;if sign bit of result is
                                                             negative, then no overflow
          ret
          sc                                                ;sign bit of result is positive,
                                                             hence overflow.
$exit:ret


          .endsect
          .title        NSISUB
          .sect         code,rom8,byte,rel

;Signed sub (16 by 16)

;         A             Operand1
;         B             Operand2
;         Carry         Return
          .public sign_sub
          .local
sign_sub:
          ld            0.b,#00
          ifbit 7,(A+1).b
          inc           0.b
          ifbit 7,(B+1).b
          inc           0.b
          ;if bit 0 of 0.b = 1 then op1 and op2 have different sign
          ;if bit 0 of 0.b = 0 then op1 and op2 sign are same
          ;then if bit 1 of 0.b = 0 both operands are positive
          ;else both operands are negative.
          sc
          subc          a,b                                 ;Perform unsigned addition
          rc
          ifbit 0,0.b                                       ;both operands are different sign
          jp            $chkovf
          ret                                               ;both operands are same sign,
                                                             can't produce overflow
$chkovf:
          ifbit         7,(B+1).b
          jp            $negminu
$posminu:
          ifbit         7,(A+1).b
          sc
          ret
$negminu:
          ifbit         7,(A+1).b
          sc
          ret

          .endsect
```

COMPARISON

To do signed comparison on n bit two's complement numbers first add 2**(n − 1) to the numbers. This will basically shift the numbers from −(2**n − 1) to +(2**n − 1 − 1) range to 0 to 2**n − 1.

Now comparison operations on the numbers will produce the correct result.

```
          .title        SICMP
          .sect         code,rom8,byte,rel

;Signed compare (16 by 16)

;         A             Operand1
;         B             Operand2
;         0.b           Return=00                            if a = b
;                              02                            if a > b
;                              01                            if a < b
signed_compare:
          push          a
          push          b
          add           a,#08000
          add           b,#08000
          ifgt          a,b
          jp            $great
          ifeq          a,b
          jp            $equ
$less:
          ld            0.b,#01
          pop           b
          pop           a
          ret
$great:
          ld            0.b,#02
          pop           b
          pop           a
          ret
$equ:
          ld            0.b,#00
          pop           b
          pop           a
          ret

          .endsect
```

Section 6

MICROWIRE and MICROWIRE/PLUS Peripherals

Section 6 Contents

National Semiconductor

MICROWIRE™ and MICROWIRE/PLUS™: 3-Wire Serial Interface

National's MICROWIRE and MICROWIRE/PLUS provide for high-speed, serial communications in a simple 3-wire implementation.

Originally designed to interface COP400 microcontrollers to peripheral devices, the MICROWIRE protocol has been extended to both the COP800 and HPC™ families with the enhanced version, MICROWIRE/PLUS.

Because the shift clock in MICROWIRE/PLUS can be internal or external, the interface can be designated as either bus master or slave, giving it the flexibility necessary for distributed and multiprocessing applications.

With its simple 3-wire interface, MICROWIRE/PLUS can connect a variety of nodes in a serial-communication network.

This simple 3-wire design also helps increase system reliability while reducing system size and development time.

MICROWIRE/PLUS consists of an 8-bit serial shift register (SIO), serial data input (SI), serial data output (SO), and a serial shift clock (SK).

Because the COP800 and HPC families have memory-mapped architectures, the contents of the SIO register can be accessed through standard memory-addressing instructions.

The control register (CNTRL) is used to configure and control the mode and operation of the interface through user-selectable bits that program the internal shift rate. This greatly increases the flexibility of the interface.

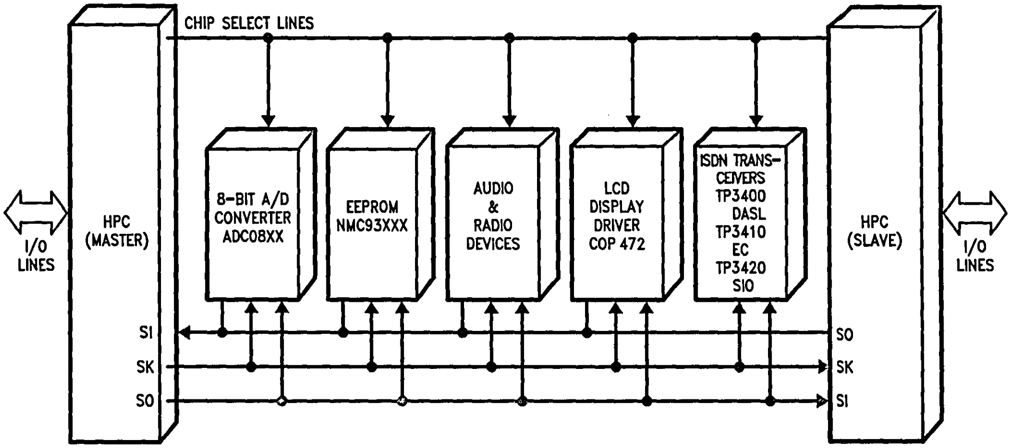
MICROWIRE/PLUS can also provide additional I/O capability for COP800 and HPC microcontrollers by connecting, for example, external 8-bit parallel-to-serial shift registers to 8-bit serial-to-parallel shift registers.

And it can interface a wide variety of peripherals:

- Memory (CMOS RAM and EEPROM)
- A/D converters
- Timers/counters
- Digital phase locked-loops
- Telecom peripherals
- Vacuum fluorescent display drivers
- LED display drivers
- LCD display drivers

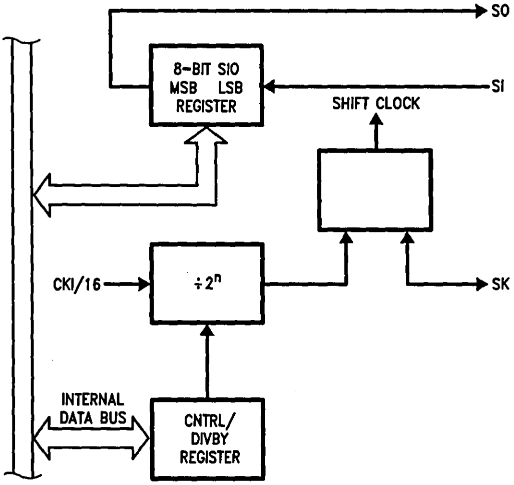
Both MICROWIRE and MICROWIRE/PLUS give all the members of National's microcontroller families the flexibility and design-ease to implement a solution quickly, simply, and cost-effectively.

MICROWIRE/PLUS System Block



TL/XX/0074-1

MICROWIRE/PLUS Block Diagram



TL/XX/0074-2

MICROWIRE and MICROWIRE/PLUS Peripherals

Part Number	Description	Databook
A/D CONVERTERS AND COMPARATORS		
ADC0811	11 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0819	19 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0831	1 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0838	8 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0832	2 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0833	4 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0834	4 Channel 8-Bit A/D Converter with Multiplexer	Linear
ADC0852	Multiplexed Comparator with 8-Bit Reference Divider	Linear
ADC0854	Multiplexed Comparator with 8-Bit Reference Divider	Linear
DISPLAY DRIVERS		
COP472-3	3 x 12 Multiplexed Expandable LCD Display Driver	Microcontroller
MM5450	35 Output LED Display Driver	Interface
MM5451	34 Output LED Display Driver	Interface
MM5483	31 Segment LCD Display Driver	Interface
MM5484	16 Segment LED Display Driver	Interface
MM5486	33 Output LED Display Driver	Interface
MM58201	8 Backplane and 24 Segment Multiplexed LCD Driver	Interface
MM58241	32 Output High Voltage Display Driver	Interface
MM58242	20 Output High Voltage Display Driver	Interface
MM58248	35 Output High Voltage Display Driver	Interface
MM58341	32 Output High Voltage Display Driver	Interface
MM58342	20 Output High Voltage Display Driver	Interface
MM58348	35 Output High Voltage Display Driver	Interface
MEMORY DEVICES		
NMC9306	16 x 16 NMOS EEPROM	Memory
NMC9313B	16 x 16 NMOS EEPROM	Memory
NMC9314B	64 x 16 NMOS EEPROM	Memory
NMC9346	64 x 16 NMOS EEPROM	Memory
NMC93C06	16 x 16 CMOS EEPROM	Memory
NMC93C46	64 x 16 CMOS EEPROM	Memory
NMC93CS06	16 x 16 CMOS EEPROM with Write Protect	Memory
NMC93CS46	64 x 16 CMOS EEPROM with Write Protect	Memory
NMC93CS56	128 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C56	128 x 16 CMOS EEPROM	Memory
NMC93CS66	256 x 16 CMOS EEPROM with Write Protect	Memory
NMC93C66	256 x 16 CMOS EEPROM	Memory

MICROWIRE and MICROWIRE/PLUS Peripherals (Continued)

Part Number	Description	Databook
TELECOM DEVICES		
TP3420	S Interface Device (SID)	Telecom
AUDIO AND RADIO DEVICES		
DS8906	AM/FM Digital PLL Synthesizer	Interface
DS8907	AM/FM Digital PLL Frequency Synthesizer	Interface
DS8908	AM/FM Digital PLL Frequency Synthesizer	Interface
DS8911	AM/FM/TV Sound Up-Conversion Frequency Synthesizer	Interface
LMC1992	Stereo Volume/Tone/Fade with Source Select	Linear
LMC1993	Stereo Volume/Tone/Fade/Loudness with Source Select	Linear
LMC835	7 Band Graphic Equalizer	Linear

National Semiconductor

COP472-3 Liquid Crystal Display Controller

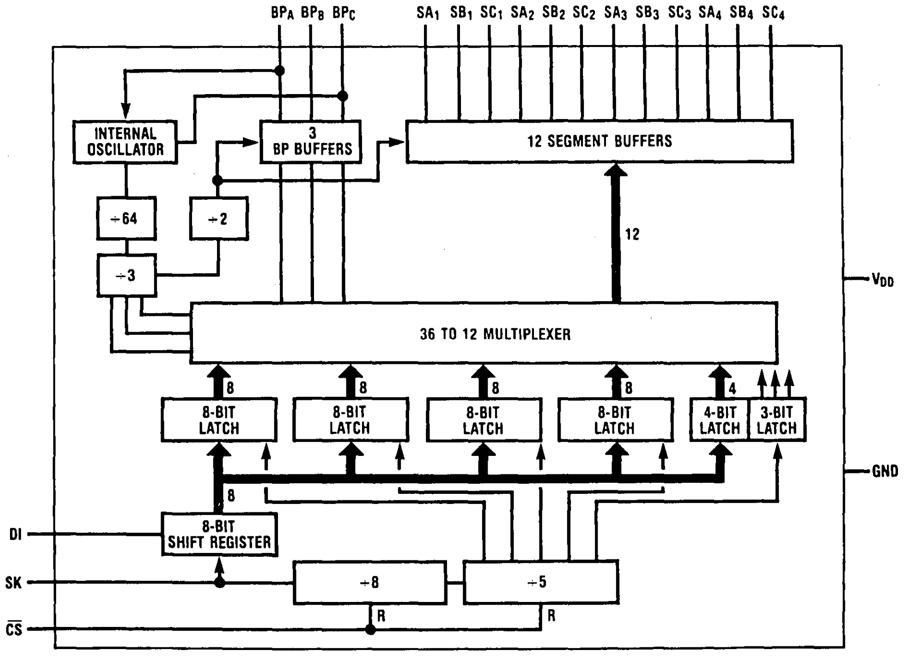
General Description

The COP472–3 Liquid Crystal Display (LCD) Controller is a peripheral member of the COPS™ family, fabricated using CMOS technology. The COP472-3 drives a multiplexed liquid crystal display directly. Data is loaded serially and is held in internal latches. The COP472-3 contains an on-chip oscillator and generates all the multi-level waveforms for backplanes and segment outputs on a triplex display. One COP472-3 can drive 36 segments multiplexed as 3 x 12 (4½ digit display). Two COP472-3 devices can be used together to drive 72 segments (3 x 24) which could be an 8½ digit display.

Features

- Direct interface to TRIPLEX LCD
- Low power dissipation (100 μW typ.)
- Low cost
- Compatible with all COP400 processors
- Needs no refresh from processor
- On-chip oscillator and latches
- Expandable to longer displays
- Software compatible with COP470 V.F. Display Driver chip
- Operates from display voltage
- MICROWIRE™ compatible serial I/O
- 20-pin Dual-In-Line package

Block Diagram



TL/DD/6932-1

Absolute Maximum Ratings

Voltage at CS, DI, SK pins	−0.3V to +9.5V	Storage Temperature	−65°C to +150°C
Voltage at all other Pins	−0.3V to V_{DD}+0.3V	Lead Temp. (Soldering, 10 Seconds)	300°C
Operating Temperature Range	0°C to 70°C		

DC Electrical Characteristics

GND = 0V, V_{DD} = 3.0V to 5.5V, T_A = 0°C to 70°C (depends on display characteristics)

Parameter	Conditions	Min	Max	Units
Power Supply Voltage, V_{DD}		3.0	5.5	Volts
Power Supply Current, I_{DD} (Note 1)	V_{DD}=5.5V		250	μA
	V_{DD}=3V		100	μA
Input Levels				
DI, SK, CS				
V_{IL}			0.8	Volts
V_{IH}		0.7 V_{DD}	9.5	Volts
BPA (as Osc. in)				
V_{IL}			0.6	Volts
V_{IH}		V_{DD}−0.6	V_{DD}	Volts
Output Levels, BPC (as Osc. Out)				
V_{OL}			0.4	Volts
V_{OH}		V_{DD}−0.4	V_{DD}	Volts
Backplane Outputs (BPA, BPB, BPC)				
$V_{BPA, BPB, BPC}$ ON	During	V_{DD}−ΔV	V_{DD}	Volts
$V_{BPA, BPB, BPC}$ OFF	BP+ Time	⅓ V_{DD}−ΔV	⅓ V_{DD}+ΔV	Volts
$V_{BPA, BPB, BPC}$ ON	During	0	ΔV	Volts
$V_{BPA, BPB, BPC}$ OFF	BP− Time	⅔ V_{DD}−ΔV	⅔ V_{DD}+ΔV	Volts
Segment Outputs (SA_1 ~ SA_4)				
V_{SEG} ON	During	0	ΔV	Volts
V_{SEG} OFF	BP+ Time	⅔ V_{DD}−ΔV	⅔ V_{DD}+ΔV	Volts
V_{SEG} ON	During	V_{DD}−ΔV	V_{DD}	Volts
V_{SEG} OFF	BP− Time	⅓ V_{DD}−ΔV	⅓ V_{DD}+ΔV	Volts
Internal Oscillator Frequency		15	80	kHz
Frame Time (Int. Osc.÷192)		2.4	12.8	ms
Scan Frequency (1/T_{SCAN})		39	208	Hz
SK Clock Frequency		4	250	kHz
SK Width		1.7		μs
DI				
Data Setup, t_{SETUP}		1.0		μs
Data Hold, t_{HOLD}		100		ns
$\overline{CS}$				
t_{SETUP}		1.0		μs
t_{HOLD}		1.0		μs
Output Loading Capacitance			100	pF

Note 1: Power supply current is measured in stand-alone mode with all outputs open and all inputs at V_{DD}.

Note 2: ΔV=0.05V_{DD}.

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Voltage at CS, DI, SK Pins	−0.3V to +9.5V
Voltage at All Other Pins	−0.3V to V_{DD}+0.3V
Operating Temperature Range	−40°C to +85°C
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

DC Electrical Characteristics

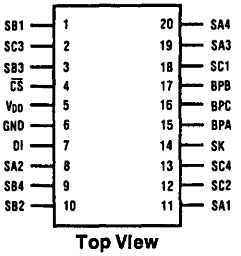
GND = 0V, V_{DD} = 3.0V to 5.5V, T_A = −40°C to +85°C (depends on display characteristics)

Parameter	Conditions	Min	Max	Units
Power Supply Voltage, V_{DD}		3.0	5.5	Volts
Power Supply Current, I_{DD} (Note 1)	V_{DD}=5.5V		300	μA
	V_{DD}=3V		120	μA
Input Levels DI, SK, CS V_{IL} V_{IH}		 0.7 V_{DD}	 0.8 9.5	 Volts Volts
BPA (as Osc. In) V_{IL} V_{IH}		 V_{DD}−0.6	 0.6 V_{DD}	 Volts Volts
Output Levels, BPC (as Osc. Out) V_{OL} V_{OH}		 V_{DD}−0.4	 0.4 V_{DD}	 Volts Volts
Backplane Outputs (BPA, BPB, BPC) $V_{BPA, BPB, BPC}$ ON $V_{BPA, BPB, BPC}$ OFF	 During BP+ Time	 V_{DD}−ΔV ⅓ V_{DD}−ΔV	 V_{DD} ⅓ V_{DD}+ΔV	 Volts Volts
$V_{BPA, BPB, BPC}$ ON $V_{BPA, BPB, BPC}$ OFF	During BP− Time	0 ⅔ V_{DD}−ΔV	ΔV ⅔ V_{DD}+ΔV	Volts Volts
Segment Outputs (SA_1 ~ SA_4) V_{SEG} ON V_{SEG} OFF	 During BP+ Time	 0 ⅔ V_{DD}−ΔV	 ΔV ⅔ V_{DD}+ΔV	 Volts Volts
V_{SEG} ON V_{SEG} OFF	During BP− Time	V_{DD}−ΔV ⅓ V_{DD}−ΔV	V_{DD} ⅓ V_{DD}+ΔV	Volts Volts
Internal Oscillator Frequency		15	80	kHz
Frame Time (Int. Osc. ÷ 192)		2.4	12.8	ms
Scan Frequency (1/T_{SCAN})		39	208	Hz
SK Clock Frequency		4	250	kHz
SK Width		1.7		μs
DI Data Setup, t_{SETUP} Data Hold, t_{HOLD}		 1.0 100		 μs ns
$\overline{CS}$ t_{SETUP} t_{HOLD}		 1.0 1.0		 μs μs
Output Loading Capacitance			100	pF

Note 1: Power supply current is measured in stand-alone mode with all outputs open and all inputs at V_{DD}.

Note 2: ΔV = 0.05 V_{DD}.

Dual-In-Line Package

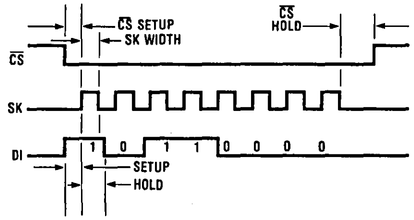


TL/DD/6932-2

Order Number COP472MW-3 or COP472N-3
See NS Package Number M20A or N20A

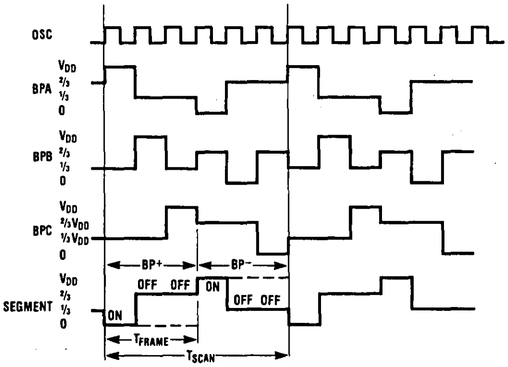
Pin	Description
$\overline{CS}$	Chip select
V_{DD}	Power supply (display voltage)
GND	Ground
DI	Serial data input
SK	Serial clock input
BP_A	Display backplane A (or oscillator in)
BP_B	Display backplane B
BP_C	Display backplane C (or oscillator out)
SA1 ~ SC4	12 multiplexed outputs

FIGURE 2. Connection Diagram



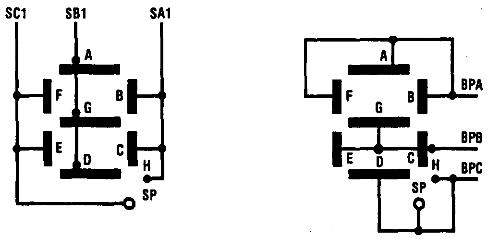
TL/DD/6932-3

FIGURE 3. Serial Load Timing Diagram



TL/DD/6932-4

FIGURE 4. Backplane and Segment Waveforms



TL/DD/6932-5

FIGURE 5. Typical Display Internal Connections
Epson LD-370

Functional Description

The COP472-3 drives 36 bits of display information organized as twelve segments and three backplanes. The COP472-3 requires 40 information bits: 36 data and 4 control. The function of each control bit is described below. Display information format is a function of the LCD interconnections. A typical segment/backplane configuration is illustrated in *Figure 5*, with this configuration the COP472-3 will drive 4 digits of 9 segments.

To adapt the COP472-3 to any LCD display configuration, the segment/backplane multiplex scheme is illustrated in Table I.

Two or more COP472-3 chips can be cascaded to drive additional segments. There is no limit to the number of COP472-3's that can be used as long as the output loading capacitance does not exceed specification.

TABLE I. COP472-3 Segment/Backplane Multiplex Scheme

Bit Number	Segment, Backplane	Data to Numeric Display	
1	SA1, BPC	SH	Digit 1
2	SB1, BPB	SG	
3	SC1, BPA	SF	
4	SC1, BPB	SE	
5	SB1, BPC	SD	
6	SA1, BPB	SC	
7	SA1, BPA	SB	
8	SB1, BPA	SA	
9	SA2, BPC	SH	Digit 2
10	SB2, BPB	SG	
11	SC2, BPA	SF	
12	SC2, BPB	SE	
13	SB2, BPC	SD	
14	SA2, BPB	SC	
15	SA2, BPA	SB	
16	SB2, BPA	SA	
17	SA3, BPC	SH	Digit 3
18	SB3, BPB	SG	
19	SC3, BPA	SF	
20	SC3, BPB	SE	
21	SB3, BPC	SD	
22	SA3, BPB	SC	
23	SA3, BPA	SB	
24	SB3, BPA	SA	
25	SA4, BPC	SH	Digit 4
26	SB4, BPB	SG	
27	SC4, BPA	SF	
28	SC4, BPB	SE	
29	SB4, BPC	SD	
30	SA4, BPB	SC	
31	SA4, BPA	SB	
32	SB4, BPA	SA	
33	SC1, BPC	SPA	Digit 1
34	SC2, BPC	SP2	Digit 2
35	SC3, BPC	SP3	Digit 3
36	SC4, BPC	SP4	Digit 4
37	not used		
38	Q6		
39	Q7		
40	SYNC		

SEGMENT DATA BITS

Data is loaded in serially, in sets of eight bits. Each set of segment data is in the following format:

SA	SB	SC	SD	SE	SF	SG	SH

Data is shifted into an eight bit shift register. The first bit of the data is for segment H, digit 1. The eighth bit is segment A, digit 1. A set of eight bits is shifted in and then loaded into the digit one latches. The second set of 8 bits is loaded into digit two latches. The third set into digit three latches, and the fourth set is loaded into digit four latches.

CONTROL BITS

The fifth set of 8 data bits contains special segment data and control data in the following format:

SYNC	Q7	Q6	X	SP4	SP3	SP2	SP1

The first four bits shifted in contain the special character segment data. The fifth bit is not used. The sixth and seventh bits program the COP472-3 as a stand alone LCD driver or as a master or slave for cascading COP472-3's. BPC of the master is connected to BPA of each slave. The following table summarizes the function of bits six and seven:

Q7	Q6	Function	BPC Output	BPA Output
1	1	Slave	Backplane Output	Oscillator Input
0	1	Stand Alone	Backplane Output	Backplane Output
1	0	Not Used	Internal Osc. Output	Oscillator Input
0	0	Master	Internal Osc. Output	Backplane Output

The eighth bit is used to synchronize two COP472-3's to drive an 8½-digit display.

LOADING SEQUENCE TO DRIVE A 4½-DIGIT DISPLAY

Steps:

1. Turn $\overline{\text{CE}}$ low.
2. Clock in 8 bits of data for digit 1.
3. Clock in 8 bits of data for digit 2.
4. Clock in 8 bits of data for digit 3.
5. Clock in 8 bits of data for digit 4.
6. Clock in 8 bits of data for special segment and control function of BPC and BPA.

0	0	1	1	SP4	SP3	SP2	SP1

7. Turn $\overline{\text{CS}}$ high.

Note: $\overline{\text{CS}}$ may be turned high after any step. For example to load only 2 digits of data, do steps 1, 2, 3, and 7.

$\overline{\text{CS}}$ must make a high to low transition before loading data in order to reset internal counters.

LOADING SEQUENCE TO DRIVE AN 8½-DIGIT DISPLAY

Two or more COP472-3's may be connected together to drive additional segments. An eight digit multiplexed display is shown in *Figure 7*. The following is the loading sequence to drive an eight digit display using two COP472-3's. The right chip is the master and the left the slave.

Steps:

1. Turn $\overline{\text{CS}}$ low on both COP472-3's.
2. Shift in 32 bits of data for the slave's four digits.
3. Shift in 4 bits of special segment data: a zero and three ones.

1	1	1	0	SP4	SP3	SP2	SP1

This synchronizes both the chips and BPA is oscillator input. Both chips are now stopped.

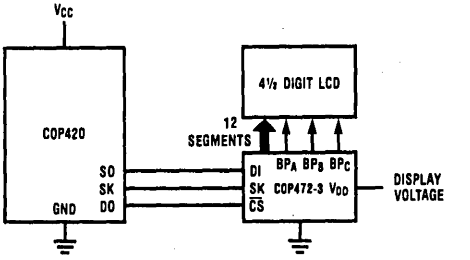
4. Turn CS high to both chips.
5. Turn CS low to master COP472-3.
6. Shift in 32 bits of data for the master's 4 digits.
7. Shift in four bits of special segment data, a one and three zeros.

0	0	0	1	SP4	SP3	SP2	SP1

This sets the master COP472-3 to BPA as a normal backplane output and BPC as oscillator output. Now both the chips start and run off the same oscillator.

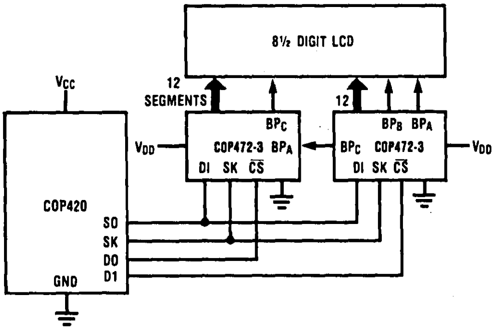
8. Turn $\overline{\text{CS}}$ high.

The chips are now synchronized and driving 8 digits of display. To load new data simply load each chip separately in the normal manner, keeping the correct status bits to each COP472-3 (0110 or 0001).



TL/DD/6932-6

FIGURE 6. System Diagram – 4½ Digit Display



TL/DD/6932-7

FIGURE 7. System Diagram – 8½ Digit Display

Example Software

Example 1

COP420 Code to load a COP472-3 [Display data is in M(0, 12)-M(0, 15), special segment data is in M(0, 0)]

```
                LBI 0, 12               ; POINT TO FIRST DISPLAY DATA
                OBD                     ; TURN CS LOW (DO)
LOOP:           CLRA
                LQID                    ; LOOK UP SEGMENT DATA
                CQMA                    ; COPY DATA FROM Q TO M & A
                SC                      ; SET C TO TURN ON SK
                XAS                     ; OUTPUT LOWER 4 BITS OF DATA
                NOP                     ; DELAY
                NOP                     ; DELAY
                LD                      ; LOAD A WITH UPPER 4 BITS
                XAS                     ; OUTPUT 4 BITS OF DATA
                NOP                     ; DELAY
                NOP                     ; DELAY
                RC                      ; RESET C
                XAS                     ; TURN OFF SK CLOCK
                XIS                     ; INCREMENT B FOR NEXT DATA
                JP LOOP                 ; SKIP THIS JUMP AFTER LAST DIGIT
                SC                      ; SET C
                LBI 0, 0                ; ADDRESS SPECIAL SEGMENTS
                LD                      ; LOAD INTO A
                XAS                     ; OUTPUT SPECIAL SEGMENTS
                NOP                     ;
                CLRA                    ;
                AISC 12                 ; 12 to A
                XAS                     ; OUTPUT CONTROL BITS
                NOP                     ;
                LBI 0, 15               ; 15 to B
                RC                      ; RESET C
                XAS                     ; TURN OFF SK
                OBD                     ; TURN CS HIGH (DO)
```

Example Software (Continued)

Example 2

COP420 Code to load two COP472-3 parts [Display data is in M(0, 12)-M(0, 15) and M(1, 12)-M(1, 15), special segment data is in M(0, 0) and M(1, 0)]

```
INIT:          LBI      0, 15
               OBD               ; TURN BOTH CS'S HIGH
               LEI      8        ; ENABLE SO OUT OF S. R.
               RC
               XAS               ; TURN OFF SK CLOCK
               LBI      3, 15    ; USE M(3, 15) FOR CONTROL BITS
               STII     7        ; STORE 7 TO SYNC BOTH CHIPS
               LBI      0, 12    ; SET B TO TURN BOTH CS'S LOW
               JSR      OUT      ; CALL OUTPUT SUBROUTINE
```

MAIN DISPLAY SEQUENCE

```
DISPLAY        LBI      3, 15
               STII     8        ; SET CONTROL BITS FOR SLAVE
               LBI      0, 13    ; SET B TO TURN SLAVE CS LOW
               JSR      OUT      ; OUTPUT DATA FROM REG. 0
               LBI      3, 15
               STII     6        ; SET CONTROL BITS FOR MASTER
               LBI      1, 14    ; SET B TO TURN MASTER CS LOW
               JSR      OUT      ; OUTPUT DATA FROM REG. 1
```

OUTPUT SUBROUTINE

```
OUT:           OBD               ; OUTPUT B TO CS'S
               CLRA
               AISC     12       ; 12 TO A
               CAB               ; POINT TO DISPLAY DIGIT (BD=12)
LOOP           CLRA
               LQID              ; LOOK UP SEGMENT DATA
               CQMA              : COPY DATA FROM Q TO M & A
               SC
               XAS               ; OUTPUT LOWER 4 BITS OF DATA
               NOP               ; DELAY
               NOP               ; DELAY
               LD                ; LOAD A WITH UPPER 4 BITS
               XAS               ; OUTPUT 4 BITS OF DATA
               NOP               ; DELAY
               NOP               ; DELAY
               RC                ; RESET C
               XAS               ; TURN OFF SK
               XIS               ; INCREMENT B FOR NEXT DISPLAY DIGIT
               JP       LOOP     ; SKIP THIS JUMP AFTER LAST DIGIT
               SC                ; SET C
               NOP
               LD                ; LOAD SPECIAL SEGS. TO A (BD=0)
               XAS               ; OUTPUT SPECIAL SEGMENTS
               NOP
               LBI      3, 15
               LD                ; LOAD A
               XAS               ; OUTPUT CONTROL BITS
               NOP
               NOP
               RC
               XAS               ; TURN OFF SK
               OBD               ; TURN CS'S HIGH (BD=15)
               RET
```

Section 7

Microcontroller Development Support

Section 7 Contents

National Semiconductor

Development Support

Our job doesn't end when you buy a National microcontroller, it only begins.

The next step is to help you put that microcontroller to work—delivering real-world performance in a real-world application.

That's why we offer you such a comprehensive, powerful, easy-to-use package of development tools.

Microcontroller On-Line Emulator

Our Microcontroller On-Line Emulator Development System is a complete, inexpensive system designed to support both hardware and software development of all NSC microcontrollers.

Using standard computer platforms (IBM PC, VAX, and others), this system gives you the tools to write, assemble, debug, and emulate software for your target microcontroller, whether it belongs to the COP400 4-bit family, the COP800 8-bit family, or the HPC 16-bit family.

The Development system itself consists of two circuit boards that interface to each other and to the host computer using a software package.

One board is called the Brain Board. It provides the major functional features of the system, linking the various elements, including other Brain Boards for a multi-workstation system tied to a single host.

The other board is called the Personality Board and is different for each microcontroller family. It provides the unique emulation functions for the system.

Your own computer CPU provides a powerful, cost-effective base for bulk storage of object code, disk editing and assembly, and for high-speed processing.

Using resident firmware in the Monitor section of each Personality Board, you can download results from your host computer, you can display and alter code in both hex and mnemonic format, you can set Breakpoints and Traces, you can execute Time measurements, and you can examine and modify internal registers and I/O.

Once you've got debugged code, you can transmit it directly to National, where we'll use it to create the tooling necessary for manufacturing the appropriate masks for your microcontroller.

Dial-A-Helper On-Line Applications Support

Dial-A-Helper lets you communicate directly with the Microcontroller Applications Engineers at National.

Using standard computer communications software, you can dial into the automated Dial-A-Helper Information System 24 hours a day.

You can leave messages on the electronic bulletin board for the Applications Engineers, then retrieve their responses.

You can select and then download specific applications data.

Dial-A-Helper

Voice:	(408) 721-5582 (8 a.m.–5 p.m. PST)
Modem:	(408) 739-1162 (24 Hrs./day)
Setup:	Baud rate 300 bps or 1200 bps 8 bits, no parity, 1 stop

Dedicated Applications Engineers

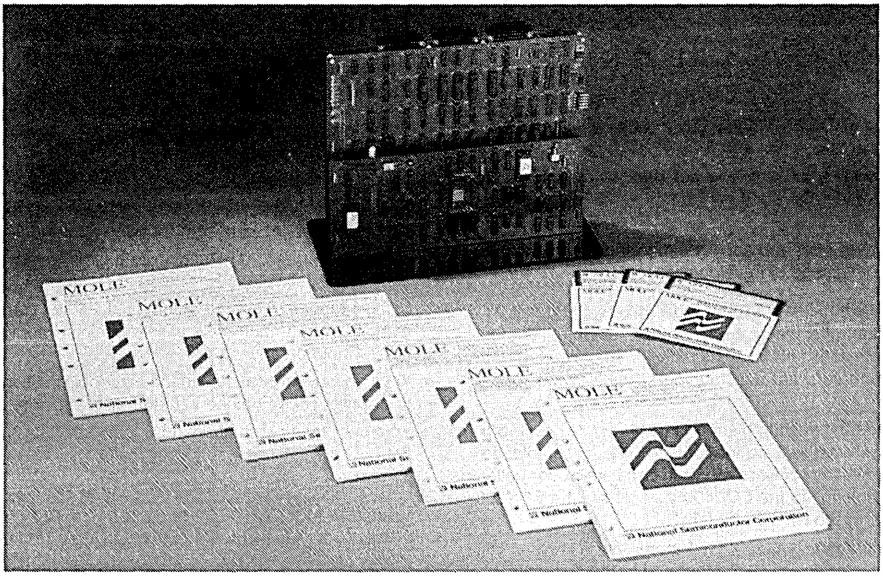
We've assembled a dedicated team of highly trained, highly experienced engineering professionals to help you implement your solution quickly, effectively, efficiently and to ensure that it's the best solution for your specific application.

At National, we believe that the best technology is also the most usable technology. That's why our microcontrollers provide such practical solutions to such real design problems. And that's why our microcontroller development support includes such comprehensive tools and such powerful engineering resources.

No one makes more microcontrollers than National and no one does more to help you put those microcontrollers to work.

National Semiconductor

Microcontroller Development Support



TL/DD/8830-14

Development Tools

The NSC Microcontroller On Line Emulator Development System is designed to support the development of NSC Microcontroller products. These include COPS™ family, and the HPC™ family of products. This system provides effective support for the development of both software and hardware in Microcontroller-based applications.

A system consists of three components: a Brain Board, a Personality Board, and software for a host computer. The host may be an IBM®-PC, or one of a number of inexpensive PC compatibles. The cross-assemblers and cross-compilers provided by National Semiconductor will run under control of the host computer MS-DOS operating system.

The Brain Board provides the development system with the capability of communicating with the user's Host CPU. Resident firmware on the Brain Board allows the user to download assembled load modules over the RS-232 link from the host computer, display and alter code in both hex and mnemonic format, initiate Breakpoints, Traces, and timing on addresses and external events, examine and modify the internal resources of the Microcontroller being emulated. The Brain Board also provides all the hardware and firmware to program standard EPROMs up to 27256's (32k x 8).

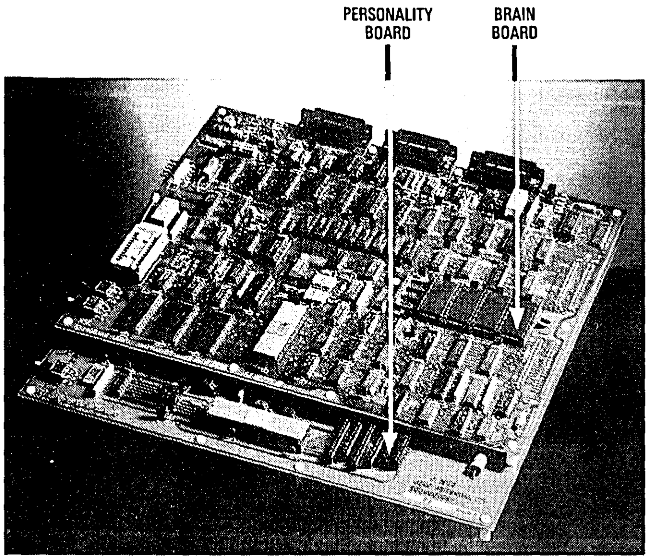
Development system flexibility is provided by the Personality board. This component tailors the system to emulate a single microcontroller family or device. For instance, one Personality Board supports the COP400 CMOS and NMOS family. This Personality Board provides emulation capability for 42 Microcontroller device types.

Personality boards are also available for the HPC and COPS family of M^{2}CMOS products.

The host CPU contributes cost effective bulk storage and high speed processing. Disk editing and assembly operations are controlled by the host CPU. The results are down loaded to the Brain Board over the RS-232 link.

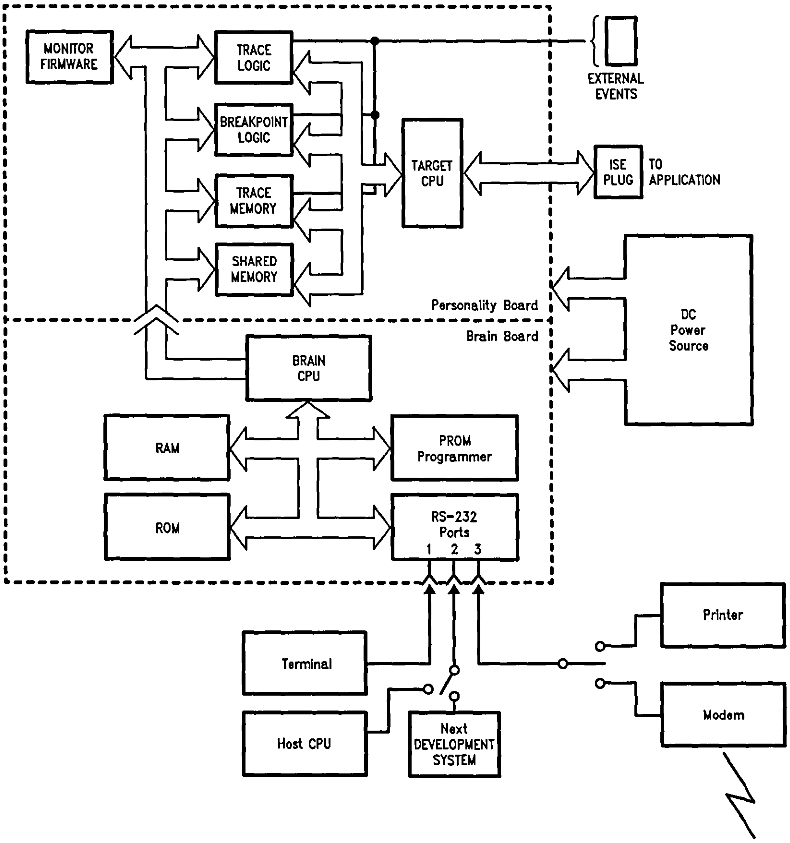
Once the application program has been completely debugged, the code may be submitted to National Semiconductor for use in creating the tooling necessary for manufacturing the masked Microcontroller device.

The Microcontroller On-Line Emulator Development System concept provides the user with a powerful development system based around a familiar host. The Brain Board/Personality Board/Host combination provides FULL emulation capability. This modular design provides maximum flexibility and maximum utility for the development of Microcontroller based systems.



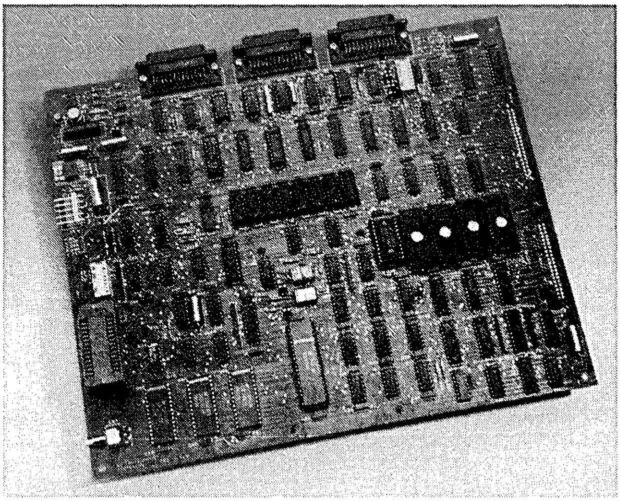
TL/DD/8830-3

System Block Diagram



TL/DD/8830-2

Brain Board



TL/DD/8830-19

General Description

The Brain Board is the pivotal component of the development system concept. In conjunction with a terminal and Personality Board it provides the user with a freestanding workstation for Microcontroller emulation. It ties the system together by communicating with the Personality Board, printers, modems, optional host computer, and other Brain Boards. Multiple Brain Boards, tied to a common host, can function as emulators for individual projects where each Brain Board is a separate workstation. They can also function as individual Microcontroller emulators within a multicontroller system.

The Brain Board utilizes a NSC800™ Microprocessor with 64k RAM and firmware ROM. It has an EPROM/EEPROM programmer for on-line changes. There are three RS-232 ports and a bus to connect the Brain to the Personality Board for actual emulation of code in the user's application system.

The RS-232 ports are used via the communication routines in firmware to interface with a host computer, terminal, modem, printer, or other development systems, for greater flexibility during system development.

The development system firmware is controlled by an EXEC. There are three major sets of EXEC commands. The first set of commands are calls to other main programs. These are:

COMM	Invoke Communications Program
DIAG	Invoke Diagnostics Program
MONITOR	Invoke Personality Emulation Monitor
PROG	Invoke PROM Programming Program

The second set of EXEC commands are:

CALC	Adds/Subtracts decimal and hex numbers
COMPARE	Compares one buffer with another
ERASE	Used to erase all or part of a buffer
HELP	Prints a summary of EXEX commands
MOVE	Moves data from one buffer to another
STATUS	Display status of buffers, display and alter RS-232 parameters

The third set of commands are used exclusively for multiple system configurations and they are:

CONNECT	Connect the user with the requested system
DISCONNECT	Disconnects the system
IDENT	Identifies the system

The Brain Board supports NSC's entire family of development system Personality boards.

Features

- Single 5V operation
- Ability to interface to host computers
- Full communication control of other systems with host computer and a modem
- Three RS-232 ports
- Auto baud selection (110, 300, 600, 1200, 2400, 4800, 9600, 19200 baud)
- Self diagnostics

Features (Continued)

- Program EPROMS
  - — MM2716, NMC27C16
  - — MM2732, NMC27C32
  - — NMC2764—NMC27256
- Program EEPROMs
  - — 9816
- Program emulator devices

PHYSICAL SIZE
10″ x 12″

POWER REQUIREMENTS
+5V DC @ 3.5A
+12.5/+21V or +25V @ 50 mA
(Optional—required only for PROM programming)

ORDER P/N:
MOLE-BRAIN

MOLE-BRAIN PACKAGE CONTAINS
Brain Board
Brain User's Manual
2 RS-232 Cables
Power Cable
Miscellaneous Hardware

Personality Boards

The Personality Board lends personality to the development system. The Monitor debugger firmware that is resident on the Personality Board is customized for the microcontroller that the Personality Board is designed to emulate, thereby giving the development systems "personality". The Monitor firmware allows the user to display the application program in either hex or mnemonic format. The user can alter or deposit hex data into the program memory. A one-line assembler is also available to allow the user to put new instructions into the application program. Breakpoint, Singlestep, Trace or Time functions are available. They allow triggering on addresses or external events. The Monitor also provides the ability to examine and modify the internal RAM and registers of the Microcontroller being emulated.

Each Personality Board has its own Monitor; however, each Monitor implements a standard set of functions. This gives all HPC, COP800 and COP400 development systems a common set of functions with identical syntax. This commonality is designed to help provide a clear and simple migration path from the low-cost COP400 4-bit microcontrollers to the high performance HPC 16-bit microcontrollers without the need to relearn the development tool.

Debug Features

The standard set of functions common to all Personality Boards is as follows.

TABLE I. Common Monitor Commands

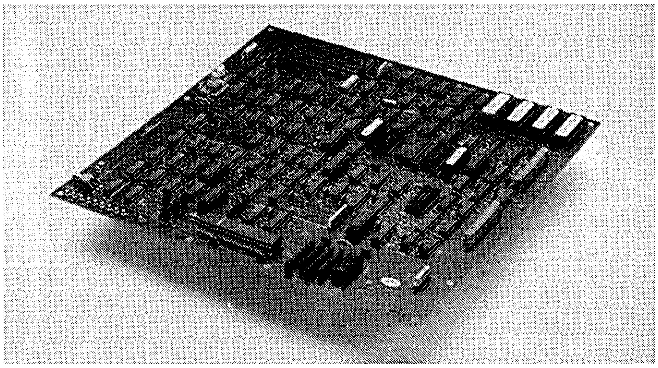
Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
DIagonstic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and development Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace data
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory

These commands are implemented on the HPC, COP800 and COP400 development systems.

Additionally, each Personality board has its own special Monitor functions that give that system additional capabilities.

COP400 Family Personality Board

COPS Personality Board



TL/DD/8830-6

General Description

The COPS Family Personality Board supports the emulation of COP400 family of Microcontrollers. The Personality Board allows the user to emulate the appropriate Microcontroller in the user's end system for fast development of application code and hardware. The Personality Board consists of: a Monitor, the hardware to control the operation of the Microcontroller in the emulation system, and an emulation cable to connect the emulator to the application system. The cable has the same pin configuration as the final masked part.

The Personality Board Monitor is contained in firmware ROM, contains an assembler and disassembler and is directly executable by the NSC800 on the Brain Board. The Monitor commands will allow the user to execute the application code, examine and modify internal registers and I/O, examine and alter object code in hex or mnemonic format, execute Time measurements, and set Trace and Breakpoints.

The Personality Board also contains 2k bytes of shared memory (RAM) for application code and the necessary hardware for Trace and Breakpoint operation.

Features

- Supports entire COPS CMOS and NMOS family
- Single 5V operation
- Firmware monitor
- Firmware diagnostics
- Firmware Line-by-Line Assembler and Unassembler
- 2k bytes of shared memory
- 256 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges, or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Features (Continued)

Common Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
DIagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and development system Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory

COP400 Monitor Special Functions

Chip	Specify COP device to emulate
Option	Specify COP chip options being emulated
Set	Set special emulation options

PHYSICAL SIZE
12″ x 12″

POWER REQUIREMENTS
+5V @ 3.5A

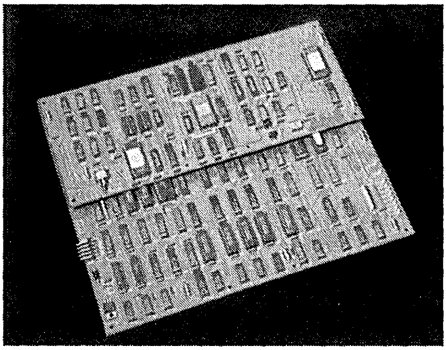
ORDER P/N:
MOLE-COPS-PB1

MOLE-COPS-PB1 PACKAGE CONTAINS
CMOS COPS Personality Board
CMOS COPS Personality Board Manual
3 Emulator Cables
Power Cable
Miscellaneous Hardware

SOFTWARE ORDERED SEPARATELY
See How To Order

COP800 Family Personality Board

COP800 Personality Board



TL/DD/8830-18

General Description

The COP800 Family Personality Board allows the development system to emulate the COP800 family. The Personality Board consists of a firmware Monitor, 8k bytes of shared memory, 2000 deep Trace memory, Port recreation logic to recapture the pins used for emulation, emulation hardware, and an In System Emulator (ISE) cable. The ISE cable has the same pinout as a socketed masked part and allows the Personality Board to function within the application system.

The NSC800 CMOS Microprocessor, located on the Brain Board, directly executes the Personality Board Monitor firmware. The Monitor allows execution of application code, examination and alteration of internal registers, examination and alteration of shared memory, and the setting of trace and breakpoints. The ISE cable connects these capabilities to the application system. Up to eight external events as well as 15-bit address and 8-bit data busses can be traced in the 2000 deep trace memory. Multiple breakpoints, plus assemble and unassemble commands are at the user's disposal.

Application programs of up to 32k bytes from Personality Board RAM may be emulated.

Features

- Supports COP800 microcontroller family
- Single 5V operation
- Firmware monitor
- Firmware diagnostics
- Firmware Line-by-Line Assembler and Unassembler
- 8k bytes of shared program memory
- 2000 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Features (Continued)

Common Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
DIagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and development system Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Diassembler for Trace or shared memory

COP8 Monitor Special Functions

CYcles	Capture COP8 execution cycles Trace memory
End	Exit Monitor and return to Exec
EXclusion	Specify address ranges to exclude from Trace
ListUnassemble	List shared memory in mnemonic form
TypeUnassemble	Type Trace memory in mnemonic form

PHYSICAL SIZE
12″ x 12″

POWER REQUIREMENTS
+5V @ 3.5A

ORDER P/N:

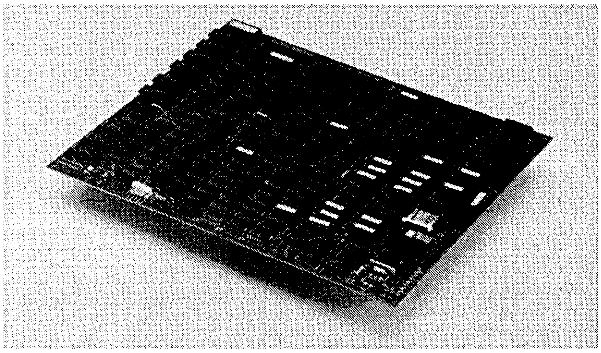
MOLE-COP8-PB1	**COP820/840**
MOLE-COP8-PB2	**COP888**

MOLE-COP8-PB1/2 PACKAGE CONTAINS
CMOS COP8 Personality Board
CMOS COP8 Personality Board Manual
Emulator Cables
Power Cable
Miscellaneous Hardware

SOFTWARE ORDERED SEPARATELY
See How To Order

HPC Family Personality Board

HPC Personality Board



TL/DD/8830-10

General Description

The HPC Family Personality Board allows the development system to emulate the High Performance Controller (HPC) family. The Personality Board consists of a firmware Monitor, 16k bytes of shared memory, 2k x 48 Trace memory, Port recreation logic to recapture the pins used for emulation, emulation hardware, and an In System Emulator, ISE, cable. The ISE cable has the same pinout as a socketed masked part and allows the Personality Board to function within the application system.

The NSC800 CMOS Microprocessor, located on the Brain Board, directly executes the Personality Board Monitor firmware. The Monitor allows execution of application code, examination and alteration of internal registers, examination and alteration of shared memory, and the setting of trace and breakpoints in either shared or user memory. The ISE cable connects these capabilities to the application system. Up to eight external events as well as 16-bit address and 16-bit data busses can be traced in the 2k deep trace memory. Multiple breakpoints, and chip error conditions plus assemble and unassemble commands are at the user's disposal.

Applications programs of up to 16k bytes from Personality Board RAM or 64k bytes from user system RAM may be emulated.

Features

- Supports HPC microcontroller family
- Single 5V operation
- Firmware monitor directly
- Firmware diagnostics directly
- Firmware Line-by-Line Assembler and Unassembler
- 16k bytes of shared program memory
- 2000 deep trace memory
- Eight external event inputs
- Trace on multiple addresses, address ranges or external events
- Breakpoint on multiple addresses, address ranges or external events
- List and alter shared memory
- Print and modify internal registers
- Singlestep
- Next—singlestep around subroutine calls
- Trigger output for logic analyzer
- Real time emulation

Features (Continued)

Common Monitor Commands

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint, Time and Trace functions
Deposit	Deposit byte value into range of shared memory
DIagnostic	On-board test routine for system checkout
Find	Find data or string in shared memory
Go	Start program execution or enable function
Help	On-screen Help menu
List	List data in shared memory
Modify	Modify on-chip RAM or Registers during Breakpt
Next	Singlestep through subroutine
Put	One-line assembler
Reset	Reset chip
RGo	Reset chip and execute Go automatically
SEarch	Search Trace memory for data or address
Singlestep	Execute one instruction, then Breakpoint
STatus	Show chip and development system Status
TIme	Time program execution or external events
TRace	Specify triggers for capturing Trace date
Type	Type Trace data or on-chip data during Breakpt
Unassemble	Disassembler for Trace or shared memory

HPC Monitor Special Functions

AlterWord	Alter consecutive words in shared memory
BAnk	Specify bank trigger information
CHip	Select chip and specify system memory map
DepositWord	Deposit word value in range of shared memory
End	Exit Monitor and return to Exec
ERror	Enable/disable HPC access error checking
EXclusion	Specify address ranges to exclude from Trace
FindWord	Find word values in shared memory
ListWord	List shared memory or memory range as words
MAp	Specify address range of memory on-board development system
XMove	Move data from one address range to another

PHYSICAL SIZE
12″ x 12″

POWER REQUIREMENTS
+5V @ 8A

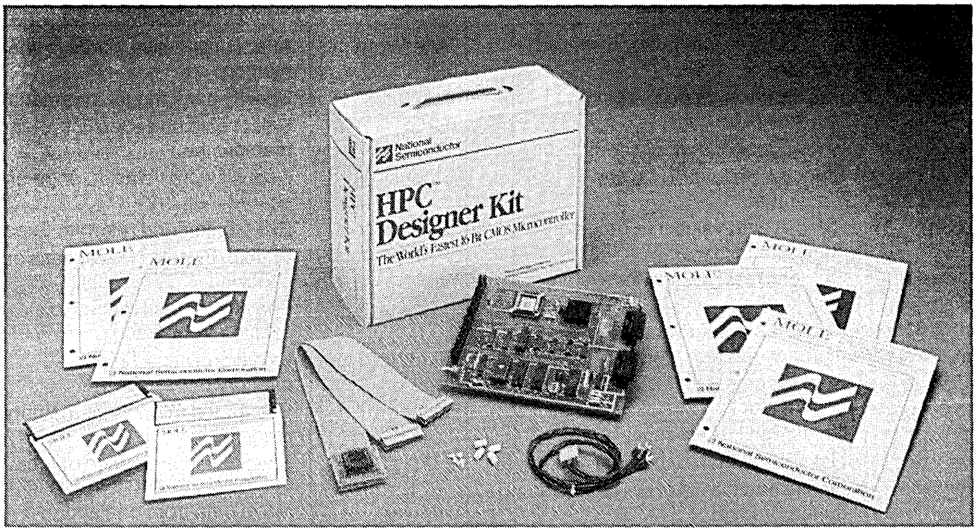
ORDER P/N:
MOLE-HPC-PB1 HPC16083 & HPC16064
MOLE-HPC-PB2 HPC16400

MOLE-HPC-PB1 PACKAGE CONTAINS
HPC Personality Board
HPC Personality Board User's Manual
1 Emulator Cable
Power Cable
Miscellaneous Hardware

SOFTWARE ORDERED SEPARATELY
See How To Order

HPC Designer's Kits

HPC Designers Kits



TL/DD/8830-20

General Description

The HPC Designer Kits are a 16-bit microcontroller Development System for program development and real-time emulation. An on-board HPC microcontroller executes monitor firmware and also acts as the target processor.

When used as the target processor, all of the features of the HPC are available for use in the application. All operating modes of the HPC are supported, with up to 56k bytes of addressable memory available for application programs.

This kit contains all of the components, manuals, and software to design an HPC system. Just add an IBM or compatible PC, +5V DC 1.5-Amp power supply and RS232 cables.

Several kits are offered (see how to order). The evaluation package contains evaluation software that allows up to 1000 lines of code to be assembled and linked. The development package has a complete Assembler/Linker/Librarian with no code limitations.

Features

- Supports HPC microcontroller family
- Single 5V operation
- Firmware monitor directly executed by the HPC
- Firmware diagnostics directly executed by the HPC
- Firmware Line-by-Line Assembler and Unassembler
- 56k bytes of addressable program memory
- Breakpoint on multiple addresses
- List and alter memory
- Print and modify internal registers
- Singlestep
- Real time emulation
- Evaluation module that allows up to 1000 lines of code to be developed for evaluation purposes

Features (Continued)

HPC Development Board Monitor

Alter	Alter consecutive bytes in shared memory
AUtoprint	Specify information to be printed on Breakpoint
BAud	Set or display the host or terminal Baud rate
BYpass	Connect terminal port to host port
Breakpoint	Set trigger point(s) for Breakpoint
Clear	Clear Breakpoint function
Deposit	Deposit byte value into range of shared memory
Dlagonstic	On-board test routine for system checkout
Go	Start program execution
Help	On-screen Help menu
List	List data in shared memory
ListUnassemble	List shared memory in mnemonic form
LOad	Load hex object file from terminal or host port
Modify	Modify on-chip RAM or Registers during Breakpt
ModifyByte	Modify on-chip RAM or registers as bytes
ModifyWord	Modify on-chip RAM or registers as words
Put	One-line assembler
Restart	Restart HPC chip
Singlestep	Execute one instruction, then Breakpoint
Type	Type on-chip data during Breakpoint
Unassemble	Disassembler for shared memory

PHYSICAL SIZE
12″ x 12″

POWER REQUIREMENTS
+5V @ 1.5A

ORDER P/N:
HPC-MOLE-EVAL0 (17 MHz Evaluation Package)
HPC-MOLE-DEVL0 (17 MHz Development Package)

HPC-MOLE-EVAL PACKAGE CONTAINS
HPC Development Board
HPC Development Board User's Manual
ISE Cable w/connector for PGA socket
Development Board Communications Software
(MS-DOS)
HPC Assembler/Linker/Evaluation Software
HPC Software User's Manual
C Compiler Evaluation Module Software
HPC C Compiler User's Manual
HPC46083/46043/46003 User's Manual
HPC46083/46043/46003 Datasheet
Dial-A-Helper User's Manual

HPC-MOLE-DEVL PACKAGE CONTAINS
HPC Development Board
HPC Development Board User's Manual
ISE Cable w/connector for PGA socket
Development Board Communications Software
(MS-DOS)
HPC FULL Assembler/Linker/Librarian Software
HPC Software User's Manual
C Compiler Evaluation Module Software
HPC C Compiler User's Manual
HPC46083/46043/46003 User's Manual
HPC46083/46043/46003 Datasheet
Dial-A-Helper User's Manual

New Development Tools for National Semiconductor Microcontrollers

National Semiconductor has an on-going program to improve development support for National Semiconductor microcontrollers, including both hardware and software tools. This program includes products both from third party tool suppliers and from National Semiconductor. The following is a brief description of some of these new, upcoming products. Please contact the factory for current status on these new tools.

HPC DEVELOPMENT SYSTEM

The *HPC Development System*, upon release, will supercede the HPC-MOLE, and is an in-system emulator supporting the full HPC product range at speeds up to 20 MHz. It provides all of the features of the existing system, including real-time trace and hardware breakpoints, as well as the following enhancements:

- 64k bytes user memory, which may be "mapped" on or off as required
- Fully enclosed system, complete with power supply
- External emulation pod providing optimal AC emulation to target system and allowing easy upgrading to new HPC family members

30 MHz DESIGNER KIT

This kit (Order No.: MOLE-HPC-DEVLI) is an upgraded version of the original HPC designer kit, which will continue in production. The new kit provides the following enhancements over the original designer kit:

- 30 MHz 1 waitstate or 20 MHz 0 waitstate operation
- 60k bytes user memory
- Enhanced monitor commands
- Special connector facilitating use of a logic analyzer to add real-time trace capability.

LOGIC ANALYZER DISASSEMBLER

As part of the 30 MHz designer kit, Newlett-Packard has made available a disassembler package for the HPC running on the HP1650 or HPC16500 logic analyzers. The analyzer plus disassembler may be used to add a powerful real-time trace capability to the 30 MHz designer kit. The combination of these products is particularly powerful in measuring the execution time of HPC programs.

ENHANCED LINKER

The HPC Linker (LNHPC) will be enhanced to provide improved support for the use of expanded memory (greater than 64k bytes) with HPC devices. The new version of LNHPC will greatly simplify the linking procedure.

Development Systems

HOW TO ORDER DEVELOPMENT SYSTEMS

Development systems are available for a variety of microcontrollers. To order a complete development package, select the section for the microcontroller to be developed and order the parts listed.

Included, along with the cross assembler, in the software package are two file conversion routines to convert the assembler output (LM) to HEX and to convert HEX to LM. Also included in the software package is a COMM program which facilitates the downloading and uploading between the host and the development, and adds the capability to make the host act as a terminal.

Development Tools Selection Table

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-HPC-PB1*	Personality Board	HPC Personality Board Users Manual	420410477-001
	MOLE-HPC-IBMR	Relocatable Assembler Software for IBM	HPC Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410836-001 420040416-001
	MOLE-HPC-IBM-CR	C Compiler for IBM	HPC C Compiler Users Manual and Software Disk Assembler Software for IBM MOLE-HPC-IBM	424410883-001
	HPC-VMS-C	C Compiler/ Relocatable Assembler/Linker for VAX/VMS	Manuals and Software	424410883-001
	HPC-UNX-C	C Compiler/ Relocatable Assembler/Linker for VAX/UNIX	Manuals and Software	Future Product
	424410897-001	Users Manual		424410897-001
COP820/840	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB1	Personality Board	COP820/840 Personality Board Users Manual	420410806-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420410703-001	Users Manual		420410703-001
COP888	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COP8-PB2	Personality Board	COP888 Personality Board Users Manual	420420084-001
	MOLE-COP8-IBM	Assembler Software for IBM	COP800 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424410527-001 420040416-001
	420411060-001	Users Manual		420411060-001
COP400	MOLE-BRAIN	Brain Board	Brain Board Users Manual	420408188-001
	MOLE-COPS-PB1	Personality Board	COP400 Personality Board Users Manual	420408189-001
	MOLE-COPS-IBM	Assembler Software for IBM	COP400 Software Users Manual and Software Disk PC-DOS Communications Software Users Manual	424409479-001 420040416-001
	424410284-001	Users Manual		424410284-001

*For HPC16400 order MOLE-HPC-PB2

Designer Kits

HOW TO ORDER DESIGNER KITS

Designer Kits are self contained development systems that contain all of the components, manuals and software to design a microcontroller based system. Just add an IBM-PC or compatible PC, +5V DC 1.5 Amps power supply and RS232 cables.

Several different kits are offered. The Evaluation package contains evaluation software that allows limited code to be developed. The Development package has no restrictions on the assembler software.

Microcontroller	Order Part Number	Description	Includes	Manual Number
HPC 17 MHz	MOLE-HPC-EVAL0	HPC Designer's Kit Evaluation Version	HPC-DB1 Board Evaluation Compiler, Assembler/Linker, Manuals	420410901-1
	MOLE-HPC-DEVL0	HPC Designer's Kit Development Version	HPC-DB1 Board Evaluation Compiler, FULL Assembler/Linker, Manuals	420410901-1

Development System Accessories and Replacement Parts

Part Type	Order Part Number	Description
EMULATOR CABLES		
68-Pin PGA Cable	MOLE-CBL-68PGA	Cable used for in-system emulation of the HPC in a 68 PGA package. For HPC development systems.
44-Pin PLCC Cable	MOLE-CBL-44PCC	Cable used for in-system emulation of the COP8 in a 44 PLCC package. For COP8 development systems.
28-Pin PLCC Cable	MOLE-CBL-28PCC	Cable used for in-system emulation of the COP8 in a 28 PLCC package. For COP8 development systems.
40-Pin DIP Cable	MOLE-CBL-40DIP	Cable used for in-system emulation of the COP8 in 40-pin DIP packages. For COP8 development systems.
28-Pin DIP Cable	MOLE-CBL-28DIP	Cable used for in-system emulation of COP4 and COP8 devices in the 28-pin DIP package. For use with COP4 and COP8 development systems.
24-Pin DIP Cable	MOLE-CBL-24DIP	Cable used for in-system emulation of COP4 and COP8 devices in the 24-pin DIP package. For use with COP4 and COP8 development systems.
20-Pin DIP Cable	MOLE-CBL-20DIP	Cable used for in-system emulation of COP4 and COP8 devices in the 20-pin DIP package. For use with COP4 and COP8 development systems.

Designer Kits (Continued)

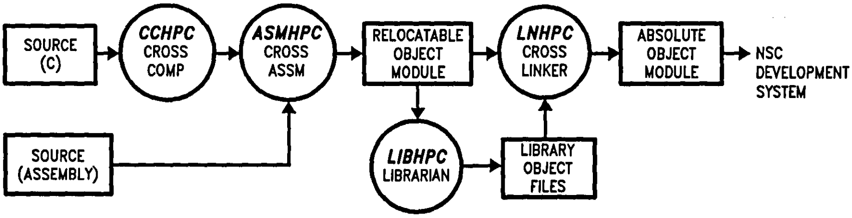
Development System Accessories and Replacement Parts (Continued)

Part Type	Order Part Number	Description
SUPPORT PRODUCTS		
COP4 PIGS	COP420P COP444CP COP444LP	Piggy-back emulator products designed to provide programmable form, fit and function emulation for the COP4XXC products in a 28-lead DIP package. An 8k x 8 EPROM sits piggy-back in a socket on top of a hybrid packaged 28-lead COP4 controller (see datasheet).
COP820/840 PIG	COP820CP-X COP840CP-X	A piggy-back emulator product designed to provide programmable form, fit and function emulation for the COP820 and COP840 products in a 28-lead DIP package (see datasheet).
COP8720 Programmer	MOLE-COP8-PROG	Adapter board for use in programming the COP8720, 8721 or 8722 devices on the development system-Brain board.
COP888 PIG	COP888CLP-X COP888CGP-X COP888CFP-X	A piggy-back emulator product designed to provide programmable form, fit and function emulation for the COP888 family (see datasheet).
COP888 Emulator	COP888CLMH COP888CGMH COP888CFMH	A form, fit, function programmable emulator for 44-lead COP888 devices (see datasheet).
HPC Emulator	HPC16083MH	A form, fit and function programmable emulator for the 68-lead PLCC HPC16083 device used in single-chip mode. Programmed with an adapter board on the development system-Brain.
HPC16083MH Programmer	MOLE-HPC-PROG	Adapter board for programming the HPC16083MH.
SYSTEM HARDWARE		
MOLE-Brain	MOLE-BRAIN	Main board component of the Microcontroller On-Line Emulator Development System.
MOLE SOFTWARE SUPPORT FOR THE IBM-PC		
HPC Assembler-IBM	MOLE-HPC-IBMR	Relocating ASMHPC Assembler/Linker/Librarian.
HPC C Compiler-IBM	MOLE-HPC-IBM-CR	CCHPC C Compiler. Includes the HPC Assembler.
HPC C Compiler-VAX/VMS	HPC-VMS-C	CCHPC C Compiler. Includes HPC Assembler.
HPC C Compiler-VAX/UNIX	HPC-UNX-C	CCHPC C Compiler. Includes HPC Assembler.
HPC Evaluation Software	MOLE-HPC-IBMEVL	HPC Evaluation software. Includes: ASMHPC and CCHPC evaluation modules and manuals.
COP8 Assembler	MOLE-COP8-IBM	COP800 Assembler.
COP4 Assembler	MOLE-COPS-IBM	COP400 Assembler.
Dial-A-Helper	MOLE-DIAL-A-HLP	Dial-A-Helper manual and communications software.

National Semiconductor

PRELIMINARY

HPC™ Software Support Package



TL/DD/9727-1

- **Choice of host systems**
  - **—IBM® XT/AT PC-DOS**
  - **—VAX™ VMS™**
  - **—VAX UNIX®**
- **CCHPC C Compiler**
  - **—ANSI Draft Standard C (February 1986)**
  - **—Additional storage class modifiers supported**
  - **—Additional statement types included**
  - **—Supports embedded assembly code**
  - **—Supports multiple source files**
- **LIBHPC Librarian**
  - **—Supports user developed library modules**
- **ASM HPC Assembler**
  - **—Macro and conditional assembly**
  - **—Instruction size optimization**
  - **—Symbol table and cross reference output**
  - **—Object files are linkable and relocatable**
- **LNHPC Linker**
  - **—Links multiple relocatable object modules**
  - **—Selects required modules from library files**

General Description

The HPC software support packages provide development system support for the HPC family of 16-bit single chip microcontrollers. Two software packages are offered that support the HPC: HPC Assembler/Linker/Librarian and HPC C Compiler. Both packages are available for a choice of host systems: IBM XT/AT PC DOS, VAX VMS and VAX UNIX.

The assembler produces relocatable object modules from the HPC macro assembly language instructions. The object modules are then linked and located to absolute memory locations. The absolute object module may be downloaded to the HPC Development System for debugging.

The C compiler generates assembly source. The C Compiler may optionally pass symbolic information through the assembler and linker to the absolute object module.

HPC C Compiler—CCHPC Introduction

The HPC C Compiler (CCHPC) is a full and complete implementation of ANSI Draft Standard C (Feb 1986) for freestanding environment. Certain additions are included to take advantage of special features of the HPC (for the specific needs of microcontrollers). The enhancements include the support of two non-standard statement types (loop and switchf), non-standard storage class modifiers and the ability to include assembly code in-line. The compiler supports enumerated types of structures by value, functions returning structures, function prototyping and argument checking.

Symbol Names, both internal and external, are 32 characters. Numerics are 16-bit for **short** or **int**, 32-bit for **long**, and 8-bit for **char**, all as either **signed** or **unsigned**; floating point is offered as **float** or **double**, both using IEEE format.

All data types, storage classes and modifiers are supported. Additional storage class modifiers are provided:

BASEPAGE place **static** variable in faster and more efficient on-chip basepage memory.

NOLOCAL declare function without local variables, thus no stack frame.

INTERRUPTn declare function to execute in response to specific interrupt(s).

ACTIVE declare function to be accessed via faster and more efficient function call mechanism.

All statement types are supported, and two additions are provided:

loop (count) simpler, more efficient **for** looping command.

switchf (value) faster form of **switch** command without constraint checking.

CCHPC SPECIFICATIONS

Note: Enhancements are boldface.

Name length		32 letters, 2 cases
Numbers		
Integer,	Signed and Unsigned	16–32 bits
	Short and Long	16 bits and 32 bits
Floating,	Single and Double	32 bits and 32 bits

- Preprocessor
  - #include
  - #define #define() #undef
  - #if #ifdef #ifndef #if defined #else #elif #endif
- Declarations
  - auto register const volatile **BASEPAGE**
  - static static global static function **NOLOCAL INTERRUPTn ACTIVE**
  - extern extern global extern function
  - char short int long signed unsigned float double void
  - struct union bit field enum
  - pointer to array of function returning
  - type cast typedef initialization
- Statements
  - ; { ... } expression ; assignment ; structure assignments ;
  - while () ... ; do ... while () ; for(; ; ;) ... ; **loop () ... ;**
  - if () ... else ... ; switch () ... ; case : ... ; default : ... ; **switchf () ... ;**
  - return ; break ; continue ; goto ... ; ... :
- Operators

primary:	function() array[] struct_union . struct_pointer ->
unary:	* & + - ! ~ ++ -- sizeof (typecast)
arithmetic:	* / % + - << >>
relational:	< > <= >= == !=
boolean:	& ^ | && ||
assignment:	= += -= *= /= %= >>= <<= &= ^= |=
misc.:	?: ,

- Functions
  - arguments: Numbers, Pointers, Structures
  - return values: Numbers, Pointers, Structures
  - forward reference (argument checking)
- Library Definition **Limited-Freestanding environment**
- **Embedded Assembly Code**

HPC C Compiler—CCHPC Introduction (Continued)

All operators are supported, and anachronisms have been eliminated (as per the standard). Structure assignment, structure arguments, and structure functions are also supported. Forward reference functions and argument type checking is supported.

Assembly code may be embedded within C programs between special delimiters.

COMPILER COMMAND FEATURES

The CCHPC runs under different host operating systems. Depending on the host system and the CCHPC command line options, ordering of the elements and their syntax may vary. In all cases, the command line consists of the command name, options or switches, and the filename to be compiled.

The compiler output, in the form of ASMHPC assembler source statements, is put in a file with the extension ".asm".

The following is a description of the CCHPC options or switches:

Include C code in assembler code output—Assembler output file contains the C source code lines as comments.

Invoke C preprocessor before compilation—Allows the C preprocessor invocation to be skipped.

Invoke an alternative C preprocessor before compilation—Allows an alternative preprocessor to be used.

Setting the stack size—This switch takes a numeric argument in the form of a C constant. If the module being compiled contains the function main, the compiler uses the number as the size of the program's execution stack, in words. The option is ignored if the module does not contain main.

Creating 8-bit wide code—This switch creates code that can be executed from 8-bit wide memory by avoiding the use of instructions that fetch 16-bit operands (such as JIDW). This option DOES NOT allow the use of 16-bit values or data in 8-bit memory.

Placing string literals in ROM—The ANSI draft language standard calls for string literals, and individual copies for each usage of the literal to be stored in RAM. This switch allows CCHPC to override this requirement for efficiency, saving startup time, RAM and ROM space. Turn off compiler warning messages.

Indicating directories for include files—This switch takes a string argument which is passed to the C preprocessor. The C preprocessor uses it as a directory to search for include files.

Defining symbol names—This switch passes the string argument to the C preprocessor. It instructs the preprocessor to perform the same function as the #define, allowing the symbol definitions to be moved to the invocation line.

Undefining symbol names—Similarly, this switch passes a string argument to the C preprocessor. It removes any previous definitions.

Permit old-fashioned constructs—Certain anachronisms from Kernighan and Ritchie C that are not permitted in ANSI C will be accepted by the compiler if this option is specified. This option is a convenience for users porting a C program to CCHPC from a Kernighan and Ritchie compiler.

Set chip revision level—This switch is used to generate code to work around bugs in specified chip revisions.

Generate symbolic debug information—This option causes the compiler to create symbolic debug information which is passed to the output assembly file.

BASIC DEFINITIONS

Names may be arbitrarily long, but only the first 32 characters are significant. Case distinctions are respected.

Constants may be of type decimal, octal, hex, character and string.

Escape sequences for new line, horizontal and vertical tab, backspace, carriage return, form feed, alert, backslash, single quote, double quote, octal and hexadecimal numbers are supported.

Comments imbedded in the source code begin with "/*" and end with "*/". Comments can not be nested.

CCHPC supports the following Data types:

Name	Size in Bits
char	8
short	16
int	16
enum	8 or 16
long	32
signed char	8
signed short	16
signed int	16
signed long	32
unsigned char	8
unsigned short	16
unsigned int	16
unsigned long	32
float	32
double	32
long double	32
struct	sum of component sizes
union	maximum of component sizes

The type "char" is treated as signed. Unsigned operations are treated the same as signed operation, except for multiplication, division, remainder, right shifts and comparisons. For signed integers, the compiler uses an arithmetic right shift. For unsigned integers, a logical shift is used when shifting right.

HPC C Compiler—CCHPC Introduction (Continued)

Keywords const and volatile can be applied to any data. Const indicates that the symbol refers to a location which is read-only. If the symbol is in static or global storage, it will be assigned to ROM memory. Volatile indicates that optimization must not change or reduce the accesses to the symbol.

Since the HPC supports 8-bit operations, CCHPC does not automatically promote "char" types to "int" when evaluating expressions. For a binary operation, the compiler promotes a "char" to an "int" only if the other operand is a 16-bit (or more) value or if the result of the operation is required to be a 16-bit (or more) value. The use of 8-bit operations yields efficient code without compromising the correctness of the result.

CCHPC uses the standard C preprocessor and any standard preprocessor functions, including "#define", "#include" and macros with arguments are supported.

A program is set of intermixed variable and function definitions. Variables must always be defined before use, functions may be defined in any order.

Variable initialization is performed according to the draft ANSI standard rules.

Standard C operators, and their hierarchy are as described in the ANSI standard draft.

CCHPC allows the programmer to imbed assembler code directly in the C source. All data between "/$" and "$/" is copied directly to the assembler output file generated by CCHPC.

CCHPC IMPLEMENTATION DEPENDENT CONSIDERATIONS

Memory

CCHPC is designed to execute in a 16-bit environment. Special care must be taken when using CCHPC in an 8-bit HPC system.

Storage Classes

CCHPC supports the following storage classes:

auto
static
register
typedef
extern

Due to HPC architectural features, the "register" storage class is limited. A variable can be assigned a "register" only if it is of type pointer and only if a register is available. The first "register" pointer variable encountered is assigned to the HPC B register, the second to the HPC X register and any subsequent ones are treated as "auto" (unless NOLOCAL is in effect, in which case it will be treated as "static").

The default storage class for global declarations is "static". The default storage class for declarations within functions is "auto".

Storage Class Modifiers

To make maximum efficient use of HPC architectural features CCHPC supports the notion of "storage class modifiers". A storage class modifier may appear with or in place of a storage class. Following is the set of storage class modifiers:

Keyword	Applicable to
BASEPAGE	variable
ACTIVE	function
NOLOCAL	function
INTERRUPTn (where n=1 to 7)	function

Storage class modifiers may be supplied with each variable or function declaration. The effect of each storage class modifier is described in the following:

BASEPAGE—The variable will be allocated in the BASE section. Accessing a basepage variable is more efficient than accessing any other type of variable but the amount of basepage storage is limited.

ACTIVE—The address of the function is placed in the 16 word JSRP table. Calls to the function will require 1 byte of code. The most frequently called functions should be considered for designation as ACTIVE functions for maximum code efficiency.

NOLOCAL—The functions local variables are not allocated on the run-time stack. Instead, they are allocated in static storage. Access to local variables in a NOLOCAL function will be more efficient since access can be direct rather than indexed from the frame pointer. If a function has no arguments or local variables, then entry and exit from the function will be much more efficient since there will be no need to adjust the frame pointer on entry and exit of the function.

INTERRUPTn—These modifiers can be used to set interrupt vectors (one through seven) to point to a particular function. Any function which has an INTERRUPT storage class modifier has special entry and exit code generated. This code will push all HPC registers (A, B, K, X, PSW and word at RAM address 0) onto the stack before executing normal function entry code. Exit code restores all registers before returning from the interrupt.

C Stack Formation

The Stack Pointer (SP) is initialized to the start address assigned by the linker. The Stack Pointer always points to the next free location at the top of the stack.

Within a function, the compiler maintains a Frame Pointer which is used to access function arguments and local automatic variables. The Frame Pointer location is reserved by the compiler at location Oxbe.

HPC C Compiler—CCHPC Introduction (Continued)

To call a function, the compiler pushes arguments onto the stack in reverse order, performs a jump subroutine to the function, then decrements the Stack Pointer by the number of bytes pushed onto the stack. Since all stack pushes are 16-bits, any 8-bit arguments are automatically promoted to 16-bits. On function entry, the compiler creates new stack and frame pointers for the function. On exit, the stack and frame pointers are restored to the values they had on entry to the function.

Using In-Line Assembler Code

CCHPC allows in-line assembler code to be entered in the body of a C function. The assembler code can access any of the currently active variables or can get the address of a variable.

Efficiency Considerations

HPC code size and execution time can be optimized by making maximum use of BASEPAGE variables. When BASEPAGE is full, static variables are next most efficient. The least efficient variables are automatic since they require an indirect indexed access. Minimizing the use of longs and floats will improve efficiency. The HPC architecture strongly supports unsigned arithmetic, so the programmer should use unsigned variables except for cases that absolutely require signed arithmetic. The compiler does not attempt to identify common subexpressions for computation only once, so this must be done by the programmer.

Statements and Implementation

The following C statements are supported by CCHPC:

expression;
if
if . . . else
while . . .
do . . . while
for . . .
break
goto
continue
return
return . . .
case . . .
default
switch . . .
switchf . . .
loop . . .

The switch statement will generate an efficient jump table for a set of cases if the cases are sufficiently close, or it will generate individual tests for each case. The switchf statement is the same as the switch statement except that when a jump table is generated for the switchf statement the compiler does not generate the code necessary to check the bounds of the value to be switched on. This creates a more efficient form of the switch statement but the programmer must insure that the value being switched on is in range.

The loop statement is an extension to the ANSI standard. Loop allows the programmer to create a code efficient loop by using the HPC DECSZ instruction. The loop statement may be nested. A break statement inside the loop will cause an immediate exit from the loop.

Run-Time Notes

During evaluation of complex expressions, the compiler uses the stack to store intermediate results.

All HPC C programs start with a call to the function "main" with no arguments. Before calling "main", run-time start-up code initializes RAM. The initial values of static or global variables with initialization are stored in ROM and copied to the appropriate variables in RAM. Static or global variables without initialization are cleared to zero. The function "main" must be defined. When "main" returns to the run-time start-up routine it executes the HALT macro provided which puts the chip in an infinite loop.

Since the run-time stack is of fixed size and there is no check for stack overflow, it is up to the programmer to insure that the stack area is large enough to prevent stack overflow.

Memory location zero is reserved by the compiler.

The HPC C Compiler User's Manual provides additional information on the features and functions of CCHPC.

HPC Cross Assembler—ASMHPC

INTRODUCTION

The HPC cross-assembler (ASMHPC) is a cross-assembler for the NSC HPC family of microcontrollers. ASMHPC translates symbolic input files into object modules and generates an output listing of the source statements, machine code, memory locations, error messages, and other information useful in debugging and verifying programs.

ASMHPC has the following useful features—

- Macro capability that allows common code sequences to be coded once.
- Conditional code assembly is supported.
- Translates symbolic assembly code modules into object code. Object modules are linkable and relocatable.
- Symbolic names may be defined for any HPC register, memory location or I/O port. Symbols may be defined as byte or word size.
- Symbol table and cross-reference output is provided.
- Full set of Assembler directives are provided for ease of generating vector tables for interrupts, short subroutine calls, jump indirects and other data generation within the object program.
- Data and code sections are user definable. Sections may be relocatable or absolute. Sections

HPC Cross Assembler—ASMHPC (Continued)

may be assigned to 8-bit memory to support the HPC 8-bit mode. Data sections may be assigned to basepage RAM on the HPC to maximize efficient access to variables.

— Accepts assembly source code generated by the HPC C Compiler, CCHPC.

— Full set of Assembler controls for greater flexibility in debugging modules and programs created by ASMHPC.

ASSEMBLY LANGUAGE ELEMENTS

Assembly Language Statement

Assembly language statements are comprised of four fields of information.

Label field—This is an optional field. It may contain a symbol used to identify a statement referenced by other statements. A symbol used in this manner is called a label.

Operation field—This field contains an identifier which indicates what type of statement is on the line. The identifier may be an instruction mnemonic or an assembler directive. The operation field is required on all assembler statement lines, except those lines which consist of only a label and/or comment.

Operand field—The operand field contains entries that identify data to be acted upon by the operation defined in the operation field. Operand examples are source or target addresses for data movement, immediate data for register initialization, etc.

Comment field—Comments are optional descriptive notes that are included in the program and listings for programmer reference and program documentation. Comments have no effect on the asembled object module file.

Character Set

Each assembly language statement is written using the following characters:

Letters—A through Z (a through z)
Numbers—0 through 9
Special Characters—!$%'( )* + , − ./;:< = > & # ?_b^

Note: Upper and lower case are distinct; b^ indicates a blank.

Location Counter

There is a separate location counter for each program section, and the counter is relative to the start of that section. The assembler uses the location counter in determining where the current statement goes in the current program section. If the program section is relocatable, the linker does the final job of assigning an absolute address to the instruction.

Symbols and Labels

Symbols and labels are used to provide a convenient name for values and statements. Symbols and labels have the same rules for construction, only their use distinguishes a symbol from a label.

Rules for symbol or label construction are:

1. The first character must be either a letter, a question mark (?), an underscore (_), a dollar sign ($) or a period (.).
2. All other characters may be any alphanumeric character, dollar sign ($), question mark (?) or underscore (_).
3. The maximum number of characters in a symbol or label may be selected by the user with the SIZESYMBOL control. The default is 64.
4. Symbols starting with dollar sign ($) are local symbols and are defined only within a local region.
5. Labels and symbols are case sensitive.

Operand Expression Evaluation

The expression evaluator in the assembler evaluates an expression in the operand field of a source program. The expressions are composed of combinations of terms and operators. An expression may consist of a single term or may consist of two or more terms combined using operators. Terms are—numbers in decimal, hexadecimal, octal or binary, string constants, labels and symbols or the location counter symbol. Each term has four attributes: its' value, relocation type, memory type and size. The relocation type is either absolute or relocatable. The memory type indicates whether the term represents a BASE, RAM8, ROM8, RAM16, ROM16 or null (in the case of an absolute term). The size of a term is null, byte or word.

The operators allowed in ASMHPC are: arithmetic, logical, relational, upper and lower byte extraction and untype operators. Arithmetic operators are +, −, *, /, MOD, SHL, ROL and ROR. The logical operators are NOT, AND, OR and XOR. The relational operators are EQ, NE, GT, LT, GE and LE. Upper and lower extraction operators are HIGH and LOW. The untype operator is &.

Parentheses are permitted in expressions. Parentheses in expressions override the normal order of evaluation, with the expression(s) within parentheses being evaluated before the outer expressions.

Numbers are represented in ASMHPC in 16-bit 2's complement notation. Signed numbers in this representation have a range of −32768 (x'8000) to +32767 (x'7FFF). Unsigned numbers are in the range of 0 to 65535. String constants are internally represented in the 8-bit ASCII code. All expression evaluation is done treating terms as unsigned numbers, for example, −1 is treated as having the value x'FFFF. The magnitude of the expression must be compatible with the memory storage available for the expression. For example, if the expression is to be stored in an 8-bit memory location, then the value of the evaluated expression must not exceed x'FF.

HPC Cross Assembler—ASMHPC (Continued)

ASSEMBLY PROCESS

The ASMHPC assembler performs its functions by reading the assembly language statements sequentially from the beginning of a module or a program to the end, generating the object code and a program as it proceeds.

The ASMHPC assembler is a multi-pass assembler which allows it to resolve forward referenced symbols and labels efficiently. The number of passes can be selected using the PASS control. This allows the user to select the level of optimization of forward referenced instructions.

MACROS

Macros help make an assembly language program easier to create, read and maintain. A macro definition is an assembly statement or statements that are referred to by a macro name. The macro may have parameters that are operated upon by the assembly statements. ASMHPC will substitute the macro definition for the macro name with the appropriate parameters during the assembly process. Repetitive or similar code can be defined as macros and the programmer can use the macros to build a library of basic routines. Variables unique to particular applications can be defined in and passed to a particular macro when called by main programs.

Defining a Macro

Macros must be defined before they are used in a program. Macro definitions do not generate code. Code is generated only when the macros are called by the assembly program. Macro definitions have a Macro name by which the macro will be referred in the program, declaration of any parameters to be used in the macro, assembler statements that are contained in the macro body and directives that define the boundaries of the macro.

Following is the macro definition structure:

```
.MACRO mname [,parameters]
        •
        •
        •
macro body
        •
        •
        •
.ENDM
```

where:

— .MACRO is the assembler directive which initiates the macro definition.

— mname is the name of the macro. Multiple macros can have the same name. The last macro defined is the macro definition used. Macro definitions are retained in the macro definition table; if the current macro is deleted by the .MDEL directive, the previous definition becomes active. If mname is the same as a valid instruction mnemonic, the macro name is used in place of the normal instruction.

— Parameters are the optional list of parameters used in the macro. Parameters are delimited from mname and additional parameters with commas.

— The macro body is a sequence of assembly language statements and may consist of simple text, text with parameters, and/or macro-time operators.

— .ENDM identifies the end of the macro and must be used to terminate the macro definition.

Calling a Macro

Once a macro has been defined, it may be called by a program to generate code. A macro is called by placing the macro name in the operation field of the assembly language statement, followed by the actual value of the parameters to be used (if any). The form of a macro call is:

mname [parameters]

where:

— mname is the previously assigned name in the macro definition and

— parameters are the optional list of input parameters. When a macro is defined without parameters, the parameter list is omitted from the call.

The macro call as well as the expanded macro assembly code will appear on the assembler listing if the appropriate controls are enabled.

Using Parameters

The power of a macro can be increased with the use of optional parameters. The parameters allow variable values to be declared when the macro is called.

When parameters are included in a macro call, the following rules apply to the parameter list:

1. One comma and zero or more blanks delimit parameters.
2. A semicolon terminates the parameter list and starts the comment field.
3. Single quotes (') may be included as part of a parameter except as the first character of a parameter.
4. A parameter may be enclosed in single quotes ('), in which case the quotes are removed and the string is used as the parameter. This function allows blanks, commas, or semicolon to be included in the parameter. To include a quote in a quoted parameter, include two quotes ('').
5. Missing or null parameters are treated as strings of length zero.

The macro operator @ references the parameter list in macro call. Using the operator @ in an expression, the number of parameters can be used to control conditional macro expansion. The @ operator may also be

HPC Cross Assembler—ASMHPC (Continued)

used with a constant or symbol to reference the individual parameters in the macro parameter list. These capabilities eliminate the need for naming each parameter in the macro definition, which is useful when there are long parameter lists. Using the @ parameter count operator it is possible to create macros which have a variable number of parameters.

The macro operator for concatenation is ^. In a macro expansion the ^ operator is removed and the strings on each side of the operator concatenated after parameter substitution. This operator provides the ability of creating variable labels through the use of macros.

Local Symbols

When a label is defined within a macro, a duplicate definition results with the second and each subsequent call of the macro. This problem can be avoided by using the .MLOC directive to declare labels local to the macro definition.

Conditional Expansion

The conditional assembly directives allow the user to generate different lines of code from the same macro simply by varying the parameter values used in the macro calls.

Nested Macro Calls

Nested macro calls are supported. A macro definition may call another macro. The number of allowable levels of nesting depends on the sizes of the parameter lists, but at least ten is typical.

A logical extension of the nested macro call is the recursive macro call, that is a macro that calls itself. This is allowed, but the programmer must insure that the call does not create an infinite loop.

Nested Macro Definitions

A macro definition may be nested within another macro. Such a macro is not defined until the outer macro is expanded and the nested macro is executed. This allows the creation of special purpose macros based on the outer macro parameters. Using the .MDEL directive and the nested macro capability a macro can be defined only within the range of the macro that uses it.

Macro Comments

All lines within a macro definition are stored with the macro, however, any text following "; ;" is removed before being stored. This text will appear on the listing of the macro definition but will not appear on the macro expansion.

ASSEMBLY LISTING

The listing generated by ASMHPC contains program assembly language statements, line numbers, page numbers, error messages and a list of the symbols used in the program. The listing of assembly language statements which generate machine code includes the hexadecimal address of memory locations used for the statement and the contents of these locations. To the left of the instruction, an "R" indicates a relocatable argument in this instruction, "X" indicates an external argument, "C" indicates a complex argument and "+" indicates macro expansion.

The assembler listing optionally includes an alphabetical listing of all symbols used in the program together with their values, absolute or relocatable type, word or byte or null type, section memory type and public or external. Optionally a cross reference of all symbol usage by source line number is given; the defining line number is preceded by a "-".

The total number of errors and warnings, if any, is printed with the listing. Errors and warnings associated with assembly language statements are flagged with descriptive messages on the appropriate statement lines.

Directives

Directive statements control the assembly process and may generate data in the object program. The directive name may be preceded by one or more labels, and may be followed by a comment. The directive's name occupies the operation field. Some directives require an operand field expression.

Assembler Controls

An assembler control is a command that may be used in the source program on a control line or on the invocation line as an option. A control line is indicated by a # in column 1 of the source line. Comments may be included on a control line by preceding the comment with a semicolon. Invocation line controls are masters and override the same controls in the program source. Examples of assembler control capabilities are: format control of the assembly listing, enable/disable listing of conditional code and conditional directives, listing of comment lines, macro expansion lines, macro object lines only. Cross references and symbol tables can be generated in the listing file, macro local symbols and constants can be put into the symbol table, number of assembler passes specified, assembler controls saved and restored . . .

ASSEMBLER INVOCATION

ASMHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for ASMHPC invocation are: the name of the assembly program(s) or module(s) to be assembled, list of assembler options and the name of a command file that contains additional invocation line source filenames and/or options. An assembler invocation line option is an assembler control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the ASMHPC HELP menu is displayed.

HPC Cross-Linker—LNHPC

INTRODUCTION

The HPC cross-linker (LNHPC) links object files generated by ASMHPC. The result is an absolute load module in various formats, such as the MOLE ".lm" format, INTEL Hex or COFF formats. LNHPC combines a number of ASMHPC relocatable object modules into a single absolute object module with all the relocatable addresses assigned. All external symbol references between modules are resolved, and library object modules are linked as required.

LNHPC creates two outputs:

1. An absolute object module file that can be downloaded to the HPC Development System for emulation and debugging. The output could also be used by the HPC Source Level Debugger if the SYMBOL option was used on CCHPC to create symbolic information.
2. A load map that shows the result of the link with an optional cross reference listing.

LNHPC MEMORY ALLOCATION

The Linker places each section in memory based on the attributes of the section and the memory that is available. Available memory is specified by the RANGE command. Each section has the following attributes:

Memory type—BASE, ROM8, ROM16, RAM8, RAM16

Size—determined from the object modules

Absolute—section was specified as absolute in assembler

Fixed—starting address was specified by the SECT command

Ranged—memory range was specified by the SECT command.

Memory is allocated section by section. Sections are allocated in the following order:

1. Each absolute or fixed section is placed in memory at its specified address.
2. Each ranged section is placed in memory within the specified range, regardless of whether this memory has been allocated in the Range Definition. An error will occur if the section can not be located.
3. All remaining sections are allocated as follows: As each section is processed, the ranges for its memory type are examined to find enough free space to allocate the section. Each range is examined in order. The first space large enough to contain the section is used. At this point, the memory allocated is marked used. If not enough memory is available to allocate the section, an error message is displayed. For efficiency, sections which may contain word aligned data (ROM16, RAM16, BASE which are word aligned) are allocated first. The user will benefit if the word aligned data is placed in these sections and byte data in other sections.

The load map shows the following:

— Range definitions showing the memory ranges specified by the /RANGE option or by the default.

— The Memory Order Map showing the starting and ending addresses of each contiguous range of memory used.

— The Memory Type Map showing how memory is allocated organized by the memory type.

— The Total Memory Map showing the allocation of all ROM and all RAM.

— The Section Table showing each section in the link, along with its starting and ending address. Section attributes are also displayed.

LINKER INVOCATION

LNHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for LNHPC invocation are—the name of the object file(s), module(s) or libraries to be linked, list of linker options and the name of a command file that contains additional invocation line source filenames and/or options. A linker invocation line option is a linker control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the LNHPC HELP menu is displayed.

HPC Cross-Librarian—LIBHPC

INTRODUCTION

The HPC cross-librarian (LIBHPC) reads object modules produced by ASMHPC and combines them into one file called a library. The linker can then search the library for any undefined external symbols and link the object module associated with the external symbol. LNHPC will only link in those library object modules required to satisfy external references to maximize efficient use of memory space. LIBHPC is a librarian utility that is provided to allow the user to develop standard modules and place them in libraries. The user may add, delete and list modules in a library file. A library of typical C functions is supplied with the HPC C Compiler (CCHPC). This library is an example of the type of library that could be created for an HPC application program. It is intended to be used as a template for the user to create a custom library specific to the application for maximum code efficiency.

HPC Cross-Librarian—LIBHPC (Continued)

LIBRARIAN INVOCATION

LIBHPC invocation will vary somewhat depending upon the host operating system being used. All systems have a similar invocation line format. The arguments for LIBHPC invocation are—the name of the library file to process, list of librarian options and the name of a command file that contains additional invocation line source filenames and/or options. A librarian invocation line option is a librarian control that is specified in a manner consistent with the requirements of the operating system. When specifying a filename, the name may include a directory path. If no arguments are specified on the invocation line, the LIBHPC HELP menu is displayed.

National Semiconductor

ISDN Basic Rate Interface Software for the HPC16400 High Performance Data Communications Microcontroller

General Description

The ISDN Basic Rate Interface Software Package implemented on the National Semiconductor HPC™ Microcontroller Family contains the software elements that are necessary to implement CCITT standards Q.921 and Q.931 as approved by T1D1 for North America.

The software package is designed to be easily unbundled and used independently by a software developer. Each layer or function is written as a separate software task. This modular design and well defined task interface make it easy to interface application dependent software to the modules provided. The coding standards for software development have been designed to ensure development of consistent, structured code, which can be easily used and maintained over the life of the code.

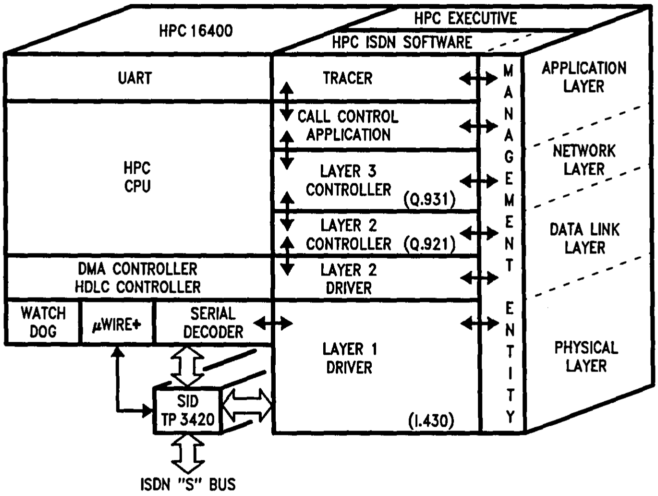
This software is supplied as a disk set and is used in conjunction with HPC development tools and software.

Features

- Multi-tasking executive
- Preemptive scheduling
- Modular software design
- Multiple timer facility
- HPC physical layer I/O interface
- Layer 2 link access procedure for the D channel (LAPD)
- Layer 2 link access procedure for the B channel (LAPB)
- Layer 3 protocol control procedure for a terminal endpoint
- Layer management entity support
- Demonstration Call Control Task
- Task_View task exerciser and debugger
- Message trace capability
- Split frame message formatting
- Source code in C language

Block Diagram

HPC ISDN Software



TL/DD/10077-1

1.0 Architectural Description

1.1 INTRODUCTION

This description defines the software required to implement the ISDN Basic Rate Interface on the HPC family of microcontrollers, including the HPC16400 which has onboard hardware specifically designed for Data Communication and ISDN applications.

The software consists of the following main parts, shown in overview in *Figure 1.1:*

- HPC Executive, providing an operating environment and services for the ISDN software and for additional application software written by OEM users of the HPC.
- I/O Drivers, interfacing to the DMA/HDLC controllers on the HPC16400 and to the TP3420 "S" Interface Device.
- Data Link Layer Software, implementing the CCITT Q.921 and X.25 link access procedures (LAPD and LAPB).
- Network Layer Software, implementing the Protocol Control Procedures defined in the CCITT Q.931 standard.
- Demonstration Call Control Module, allowing a development engineer at a terminal to make and receive ISDN phone calls which exercise the above software.
- Tracer Module, allowing a development engineer at a terminal to monitor the operation of the above software.
- Management Entity Module

1.2 SOFTWARE ARCHITECTURAL PRINCIPLES

1.2.1 Modular Multitasking Environment

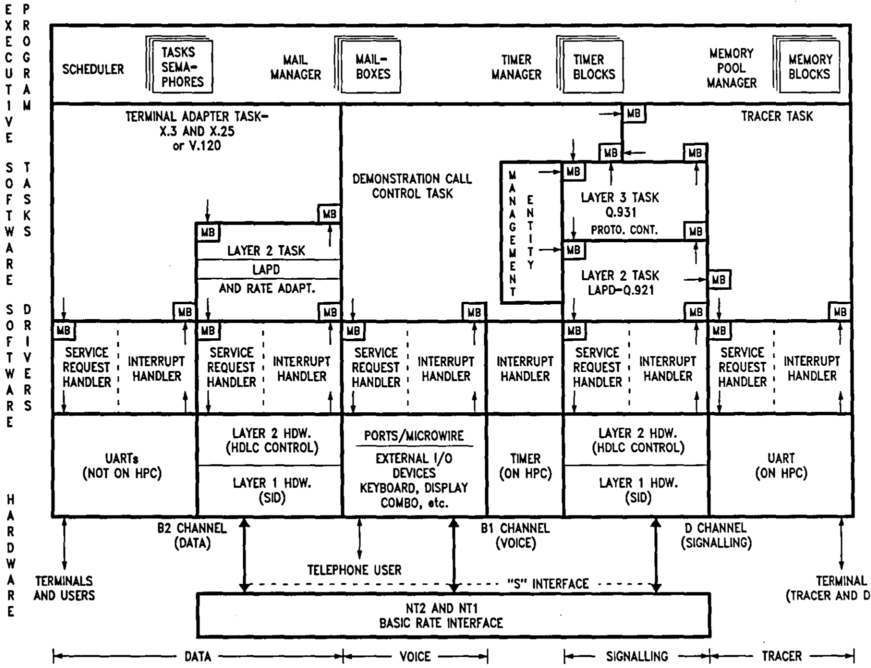
Each layer or function is written as a separate software task. Intertask communications and the interface between tasks and I/O drivers is by means of mail messages and semaphores, which are managed by the multi-tasking HPC Executive.

This modular design and well defined intertask interface make it easy for users to interface application-dependent software to the modules provided. The services of the HPC Executive (mail, semaphores, timers, memory management) facilitate the writing of software tasks and I/O drivers. These services are available to all tasks and to interrupt-level drivers.

1.2.2 Event-Driven State-Machine Architecture

Telecommunication software typically involves many invocations of the same code (one per call, one per logical connection, etc.) and requires a particular software architecture: tasks must be structured as event-driven state machines. Each task has one or more mailboxes and operates by picking up mail, one message at a time, from the mail queue, and processing the message to completion before returning for the next mail message.

Each "entity" within a task (each call, each logical connection, etc.) has a state block, indicating its current state. After picking up a mail message, the task identifies the entity involved in the message, accesses the state block for that entity, processes the message based on the state of the entity, and finally sets the state block to the new state of the entity.



NOTE: THE SYMBOL: MB REPRESENTS AN INTERTASK MAILBOX.

TL/DD/10077-2

FIGURE 1.1 HPC16400 Software for ISDN

1.0 Architectural Description (Continued)

1.2.3 Coding Standards

The coding standards for software development have been designed to ensure development of consistent, structured code, which can be easily used and easily maintained over the life of the code.

1.3 HPC EXECUTIVE

The HPC Executive provides an operating environment for the Layer 2 and Layer 3 tasks, the application tasks, the various support tasks, and the I/O drivers which interface to the hardware. It provides the following services to the tasks and I/O drivers:

- Scheduling of tasks that are ready to run, based on task priority. Preemptive scheduling and time slicing can be optionally enabled.
- Task-task and driver-task communication, by means of mail messages, which can be sent and picked up, and semaphores, which can be signaled and awaited.
- Timers, which are equivalent to mail messages with a specified delay and which allow tasks and drivers to time their activities and time out when an expected event does not occur.
- Memory management, to allocate and deallocate fixed-size buffers as needed by tasks and drivers.

Application tasks and I/O drivers developed by users of the HPC can easily be inserted in the HPC Executive environment and can take full advantage of its services.

1.4 ISDN TELECOMMUNICATIONS STANDARDS

1.4.1 CCITT Standards

The Layer 2 Task implements CCITT specification Q.921 (LAPD) and Layer 2 (LA{B) of CCITT specification x.25. The last CCITT published version of specification Q.921 is the 1984 Red Book with subsequent CCITT produced revisions. The Layer 2 Task LAPD implementation is based on the version issued in December 1986, CCITT Document COM XI-R 43-E, with the minor revisions issued in September 1987, Temporary Document 644-E Rev. 1. This version is expected to be very close to the next CCITT published version, The 1988 Blue Books. The Layer 2 Task LAPB implementation is based on the 1984 Red Book.

The Layer 3 Task implements the Protocol Control Procedures of CCITT Specification Q.931. Since this specification was still subject to revision, the software is based on the latest version of Q.931 distributed at the ECSA/ANSI T1D1 meeting in September 1987. This version is significantly changed from the 1984 Red Books.

However it is expected to be close to the next CCITT published version, the 1988 Blue Books.

In terms of the September 1987 T1D1 version of the specification, the Layer 3 Task implements the circuit-switched procedures described in Section 5. The Layer 3 Task implements the Protocol Control procedures and some of the Resource Management. The Call Control Task implements a demonstration version of the Call Control Procedures and the balance of the Resource Management.

In terms of the specification and description language (SDL) diagrams in the Q.931 specification, the Layer 3 Task implements *Figure 38* (26 pages).

The establishment and release of logical links are fully covered in the Layer 2 specifications (Q.921), but the Layer 3 aspects of this are not handled in the version of Q.931 on which the Layer 3 Task is based. Therefore, additional Layer 3 states and SDL diagrams have been created and additional software has been written to handle this requirement.

1.5 ISDN TELECOMMUNICATIONS SOFTWARE

The software packages described below are designed to be easily "unbundled" and used independently by a software developer.

1.5.1 Layer 1 I/O Driver

The Layer 1 I/O Driver controls the HPC MICROWIRE/PLUS™ interface, and the onboard Serial Decoder. This driver is responsible for the hardware initialization, the control of the Serial Decoder, the activation and the deactivation of the Layer 1 I/O device. Use of the HPC Executive mail and semaphore services makes this driver simple to implement and easy to enhance by users that require additional Layer 1 hardware interfaces.

1.5.2 Layer 2 I/O Driver

The Layer 2 I/O Driver controls the HDLC/DMA controllers onboard the HPC16400, and interfaces this hardware to the Layer 2 Task. This driver is responsible for the hardware initialization, the reception of frames toward the HPC, the transmission of frames away from the HPC, and appropriate error handling. Use of the HPC Executive mail and semaphore services makes this driver simple to implement and easy to replace with alternative drivers that a user may wish to develop.

1.5.3 Layer 2 Task

The Layer 2 Task implements the full LAPD protocol defined in Q.921, providing error free in-sequence transmission, reception and multiplexing of messages received by an HDLC controller connected to the D signaling channel. The event-driven state machine architecture, described above, enables a single software module to support simultaneous activity on multiple logical connections. The Layer 2 Task also supports X.25 LAPB processing for messages received by a second HDLC controller connected to a bearer B channel.

1.5.4 Layer 3 Task

The Layer 3 Task implements the user side of the Protocol Control Procedures of Q.931, which are used to setup, answer, suspend, resume, and disconnect a call. Specifically, it implements all of *Figure 2*/Q.931 of Q.931. The event-driven state machine architecture, described above, enables a single software module to support simultaneous activity relating to calls on both bearer B channels.

1.5.5 Demonstration Call Control Task

The latest versions of Q.931 separate the Layer 3 procedures into Protocol Control Procedures and Call Control Procedures. Call Control Procedures are application dependent. These procedures handle bearer channel selection and actual establishment of the voice channel. As Q.931 notes, these procedures can also be considered to be part of the Applications Layer. The Call Control Task implements a minimal subset of the Call Control Procedures, for demonstration purposes. In an actual application, this task will be replaced by an application-specific task, tailored to the capabilities of the actual terminal equipment (number of terminals, handsets, etc.).

1.5.6 Management Entity Task

The Management Entity Task, which is only generically defined in Q.921 and Q.931, handles housekeeping functions

1.0 Architectural Description (Continued)

for all layers. These functions include TEI negotiation with the network management entity, and the handling of unrecoverable errors. This task implements as much of the management entity as is currently defined and in addition whatever is necessary for the operation of the other tasks.

1.5.7 Tracer Task

The Tracer Task serves two purposes; to demonstrate the lower ISDN layers via a menu-driven telephone emulation mode, and to trace system mail message traffic.

1.5.8 Task_View Task Exerciser and Debugger

Task_View is a special-purpose task that can be inserted into the multi-tasking Executive environment in place of the Tracer Task. It reads and interprets a user supplied ASCII scenario file. Under control of this senario file, Task_View sends mail messages to a specified mailbox (or mailboxes), where they are read by the task under test. Mail messages sent by the task under test in response to this input are then displayed by Task_View. In this way the task may be exercised and debugged.

2.0 Functional Description

2.1 INTRODUCTION

This description defines the functional requirements of the ISDN Basic Rate Interface Software Package implemented on the National Semiconductor HPC Microcontroller Family. Specifically, the HPC16400 Software Package implements or supports the following high-level functions:

- Multi-Tasking Executive
- HPC Physical Layer I/O Interface
- Layer 2 Link Access Procedure for the D Channel (LAPD)
- Layer 2 Link Access Procedure for the B Channel (LAPB)
- Layer 3 Protocol Control Procedure for a Terminal Endpoint
- Management Entity Support
- Call Control Demonstration Task
- Message Trace Capability

The HPC ISDN Software Package has been divided into several functional software elements, as illustrated in the HPC ISDN Functional Block Diagram, *Figure 2.1*. These functional elements correspond to software modules. The purpose of this section is to introduce the various software elements, to define their interactions, and to relate their functionality to the appropriate ISDN standards, where applicable.

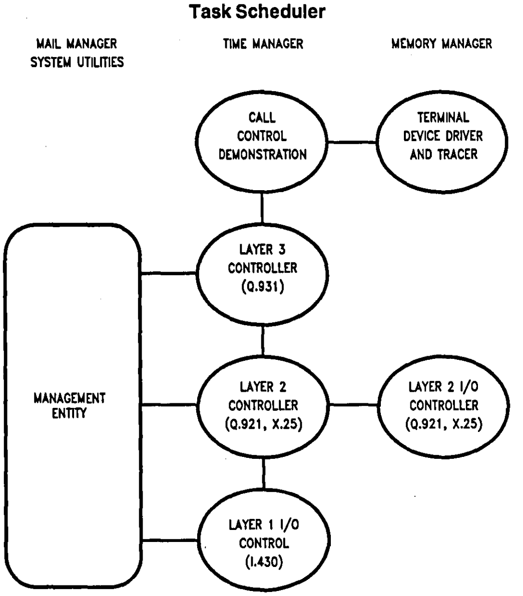
The HPC ISDN Software Package will require additional software drivers and application-specific tasks prior to serving as a useful ISDN Terminal Endpoint (TE) entity. The HPC ISDN software has been coded and documented to allow easy integration of additional application code.

The HPC ISDN Software elements illustrated in *Figure 2.1* have been divided into the following categories.

• HPC Executive	(2.2)
• I/O Device Drivers	(2.3)
• ISDN Layer Protocol Tasks	(2.4)
• Application Tasks	(2.5)
• System Utilities	(2.6)

The HPC Executive contains software elements that are necessary for HPC ISDN Applications. These elements include a Multi-Tasking Scheduler, a Memory Manager, a Timer Manager and a Mail Manager. The HPC Executive software elements are tightly coupled, and streamlined for the National Semiconductor HPC family of controllers.

The I/O Device Drivers interface the HPC hardware elements to the HPC ISDN Software. The Layer 1 Driver implements the ISDN PHYSICAL Layer 1 requirements for the HPC ISDN system. The Layer 2 Driver interfaces the HPC DMA/HDLC controller channels to the Layer 2 Link Access


Task Scheduler
MAIL MANAGER
SYSTEM UTILITIES
TIME MANAGER
MEMORY MANAGER
CALL CONTROL DEMONSTRATION
TERMINAL DEVICE DRIVER AND TRACER
LAYER 3 CONTROLLER (Q.931)
MANAGEMENT ENTITY
LAYER 2 CONTROLLER (Q.921, X.25)
LAYER 2 I/O CONTROLLER (Q.921, X.25)
LAYER 1 I/O CONTROL (I.430)

TL/DD/10077-3

FIGURE 2.1 HPC ISDN Software Functional Block Diagram

2.0 Functional Description (Continued)

Procedures. The Terminal Device Driver interfaces the HPC on-board UART to the ISDN Software. Device initialization sequences, service request tasks and accompanying interrupt service routines are all defined in the I/O Device Driver section of this document.

The Layer Protocol Tasks implement the ISDN DATA LINK Layer 2 and the NETWORK Layer 3 requirements for the HPC ISDN system. These tasks are designed to be hardware configuration and application independent. The Layer 2 Task provides both the "USER SIDE" and the "NETWORK SIDE" implementation of the CCITT Specification Q.921. The Layer 3 Task provides the "USER SIDE" implementation of CCITT Specification Q.931.

The Layer 2 Task has been designed to use many of the same routines to implement the link access procedures on either the signaling D channel or the bearer B channel (LAPD or LAPB). Design decisions have also been made to facilitate the implementation of V.120, the new rate adaption scheme that processes LAPD frames on a bearer B channel.

The Management Entity Task and the Call Control Task are Application (Specific) Tasks that are closely coupled to the specific system hardware configuration and the Central Office Network Entity Software. These tasks are provided for demonstration purposes to drive the ISDN layer entities. Application users must either replace or extensively rewrite these tasks to match their particular ISDN Application environment.

The System Utilities include the power-up reset Main Task, the NMI handler, the Timer interrupt handler, and the Watchdog Task.

The Tracer utility provides the capability of on-line tracing of intertask mail messages and task states. Tracer is primarily a passive task; it displays messages that it receives from other tasks. Tracer also provides a user interface for Telephone Simulation.

The remainder of this document is devoted to defining each of the software elements at the functional level. Where applicable, specific ISDN standard documents such as CCITT Q.921, Q.931 and X.25 will be referenced, rather than duplicating the information here.

2.2 HPC EXECUTIVE

The HPC Executive provides a multitasking environment within which the ISDN and applications tasks can run and it provides various system services to those tasks. The services of the Executive are available to both tasks and interrupt service routines.

2.2.1 Tasks, Priorities, and the Ready Queue

A task is a subroutine which can be run (called) by the Executive. Tasks are managed by the Executive as Task Control Blocks (TCB's). A task's TCB contains all the parameters needed by the Exeuctive to handle the task, in particular, the task's priority and its current starting address.

Tasks which are not blocked waiting for a semaphore or for mail are considered to be ready to run and their TCB's are queued on the Ready Queue, in the order of the tasks' priorities. The Task Scheduler runs the task at the head of the Ready Queue, i.e., the highest priority task that is ready to run. In this way the processor is always given to the highest priority task that is ready to run.

Once a task is started, it continues to run until it does a Semaphore Wait, ReadMail, or Return or, until a higher priority task is put on the Ready Queue, at which time the scheduler has the opportunity to once again choose the task at the head of prioritized Ready Queue and run that task.

A task may change the priority of any task, including itself. The priority change takes place immediately, to the extent that the target task's TCB is updated with the new priority and the queue in which the target task's TCB is waiting is resorted to reflect the new priority.

If the target task is in the Ready Queue and its new priority is higher than the priority of the running task, then the target task will run once all protected sections are exited. See Section 2.2.3, below.

2.2.2 Semaphores

A semaphore is a global variable, accessed through the Executive, which can be Signaled (incremented) by one task and Waited on by another task. A semaphore is typically used to manage the sharing between tasks of some resource, e.g., an I/O device, mail messages, etc. At any moment the value of a semaphore may be positve, negative, or zero. A positive value indicates the number of resources available, a negative value indicates the number of tasks waiting for resources and a zero value indicates that there are no resources available and no tasks waiting for them.

When a task Waits on a semaphore, if the semaphore has a nonzero positive value, the task will immediately go on the Ready Queue and the semaphore value will be decremented by one. On the other hand, if the semaphore has a zero or negative value, the task will be queued on the semaphore and the semaphore value will be decremented by one. When a task Signals a semaphore, the semaphore's value is incremented by one and the highest priority task waiting on the semaphore is put on the Ready Queue.

A common use for a semaphore is the management of a non-shareable resource, such as an I/O device. When the device is available, the associated semaphore has the value +1. When a task wishes to obtain exclusive use of the device, it Waits on the semaphore, which is then decremented to 0, with the task going immediately back on the Ready Queue. If another task then attempts to use the device, its Wait call will cause it to be placed on the Semaphore Queue and the value of the semaphore will be decremented to −1. Other tasks may also Wait on the semaphore, each decrementing its value by one. The negative value of the semaphore indicates the number of tasks Waiting for the device. The waiting tasks are ordered in the semaphore queue according to their priority. When the first task is done with the device, it Signals the semaphore, which moves the first waiting task to the Ready Queue and increments the semaphore or, if there are no waiting tasks, returns the semaphore to its original value of +1.

2.2.3 Preemptive Scheduling

Preemptive scheduling enables the executive to respond quickly to high priority events. If a task that is waiting on a Semaphore Queue modes to the Ready Queue and if that task is of higher priority than the currently running task, then, as soon as the currently running task emerges from all critical sections and non-preempt sections, the currently running task will stop running. The task that was moved to the Ready Queue will run. The preempted task will be placed on the Ready Queue in the normal manner.

2.0 Functional Description (Continued)

Executive functions allow preemption to be selectively turned on or off by task or for an entire application.

2.2.4 Time Slicing

Time slicing modifies the task scheduling algorithm as follows: at each "tick" of the timer clock (the clock which also controls the time-out timers), if the currently running task has the same priority as the task at the head of the Ready Queue, then, if the currently running task is not in a non-preempt section, it will stop running and the task at the head of the Ready Queue will run. The task that stops running is placed on the Ready Queue in the normal manner, i.e., after all tasks of equal priority. Time slicing enables the Executive to share the processor equally between tasks of equal priority.

2.2.5 Mailboxes and Mail Messages

The main form of intertask communication is the sending and receiving of mail. Mailboxes exist independently of tasks; any task may send mail to any mailbox and any task may read mail from any mailbox. However, in a typical system, each task has one mailbox from which it reads all its mail and to which other tasks send mail destined for that task.

Mail is prioritized. When a task calls upon the Executive to perform a SendMail, it specifies the priority of the message, which is inserted in the specified mailbox queue sorted by priority.

2.2.6 Timers

The Executive includes a timing facility specifically designed to handle the time-outs typical of telecom protocols and other real-time applications.

Timers are essentially a form of delayed mail. When a task sets a timer, the task provides a mailbox identifier, a mail message, and a time delay value. When the specified time delay is up, i.e. when the timer "expires", the mail message is mailed to the specified mailbox. When a task sets a timer, it receives a timer ID, which can be used to cancel the timer, if necessary, before it expires.

2.2.7 Memory Management

The memory manager is responsible for allocating and deallocating fixed-size memory blocks from fixed-size pools, which are completely defined at compile time. A memory pool may reside in extended memory.

2.2.8 System Module and Interface Module

The Multi-Tasking Executive Software is available either as source code or as object code. The interface module, which must be modified to insert application tasks, is always supplied as source code.

2.3 I/O DEVICE DRIVERS

I/O Device Drivers serve as interface routines between the HPC hardware machinery and the HPC Executive and Application Tasks. "Input" operations (data heading toward Application Tasks) are typically fielded by an Interrupt Service Routine (ISR). The ISR may SEND information to the appropriate task via the system mail facility, or it may signal the appropriate semaphore to schedule an I/O task. "Output" operations (data heading away from Application Tasks) are typically fielded by Service Request (SRQ) Tasks. SRQ Tasks communicate directly with the hardware control registers to initiate output operations. These tasks often work closely with their accompanying ISR for output initiation and completion. Higher layer tasks send mail messages to he SRQ Tasks, using the system mail facility to queue messages pending output.

The HPC ISDN Software includes three I/O Device Drivers: the Layer 1 Driver, the Layer 2 Driver and the Terminal Driver. The functionality of these drivers is defined below. Details of particular Device Driver ISR and SRQ Task interactions are defined in the Software User's Manual.

2.3.1 Layer 1 I/O Device Driver

The Layer 1 I/O Device Driver provides implementation of the ISDN PHYSICAL Layer 1 for the HPC environment. This Device Driver controls the NSC MICROWIRE/PLUS Interface to the NSC TP3420 "S" Interface Device (SID), and the HPC16400 onboard, Serial Decoder. Control of a COMBO™ Codec, a display, and a keypad has been implemented later by either adding to this driver, or using it as a model for additional drivers.

The primary responsibility of this driver is to initialize and control the SID. The higher layer ISDN tasks mail activation and deactivation messages to the Layer 1 Service Request Task. This task sends the appropriate command to the SID via the MICROWIRE/PLUS Interface. The SID interrupts the HPC whenever it changes state. The Layer 1 Interrupt Service Routine fields responses when the SID changes state and mails the information to the Layer 2 Controller Task and to the Management Entity Task.

The Serial Decoder is initialized to MODE 4, with the ISDN D Channel terminated by DMA/HDLC Channel #1, and Bearer Channel B2 terminated by DMA/HDLC Channel #2. The SID can swap B1 and B2 internally to allow voice or data on either channel.

The Layer 1 I/O Device Driver can communicate with any other Task via the System Mail Utilities.

2.3.2 Layer 2 I/O Device Driver

The Layer 2 I/O Device Driver interfaces the two HPC16400 onboard DMA/HDLC channels; one to the 16 kbit per second "D" signaling channel, and one to the 64 kbit per second bearer (B2) channel. The Layer 2 Service Request Task receives Physical Layer (PH) Primitives from the Layer 2 Controller Task via the system mail utility. The Layer 2 Interrupt Service Routine handles block messages received from the DMA Controller and mails them as Physical Layer (PH) Data Primitives to the Layer 2 Controller Task. This generic mail message interface allows an Application User to easily introduce external DMA and HDLC Controllers, and accompanying device drivers, that either replace or complement the existing onboard controllers.

HDLC/DMA Channel #1 is attached to the ISDN signaling D channel, and will be referred to as the LAPD Channel. HDLC/DMA Channel #2 is attached to bearer channel B2, and will be referred to as the LAPB Channel. The two channels operate independently of each other as much as possible. Since they share the same interrupt hardware, the Layer 2 Interrupt Service Routine must poll the Message Pending Register and the Error Status Register to determine the source of each interrupt. Both HDLC/DMA channels use the HPC field separation feature for transmission and reception of data. This feature relieves some memory concerns, since it allows small memory buffers to be used for mes-

2.0 Functional Description (Continued)

sages that only have headers. In the transmit direction this feature allows large contiguous buffers to be broken up into smaller send buffers without having to copy them following a header. Issues specific to the HDLC/DMA Channels are defined below.

HDLC/DMA Channel #2, the LAPB Channel, requires frame sizes to be nominally 130 bytes, 2 bytes of header and 128 bytes of information. Provision can be made for messages with up to 1026 bytes, 2 bytes header and 1024 bytes of information.

The presentation of data between the Layer 2 Driver and Layer 2 Controller is identical regardless of which channel the frames are associated with.

2.3.3 Terminal Device Driver and Tracer

The Terminal Device Driver interfaces to the HPC onboard UART. The associated SRQ Driver Task, referred to as Tracer, serves primarily as a high-level demonstration vehicle. Tracer can field mail messages from any other task in the system, as well as keystroke mail messages from its accompanying ISR. Tracer's responsibilities include the following functions:

- Management of the Telephone Simulation User Interface,
- Display Management of the System Trace Mail Messages,
- Proper Display of Task-Related Information

The Telephone Simulation function of Tracer allows the user to enter "telephony-like" keystroke characters, that are passed to Tracer, then on to the ISDN layer tasks for processing. Menu responses are fielded by Tracer to select various levels of the Trace function, as well as to enter and exit the Telephone Simulation mode.

Depending on the level of trace that is selected, Tracer receives mail messages from the system tasks and properly formats them on the CRT display. Tracer offers various levels of trace capability. Trace can be turned off all together, in which case only the application layer Telephone Simulation inputs will be displayed. Trace can display all messages from every layer, or it can be set to "zoom" to display only the messages at a particular layer. Messages will generally have address fields and data fields.

The Terminal Driver's Interrupt Service Routine (ISR) handles keyboard characters from the UART and mails them to the Tracer SRQ Task for further processing. The ISR also handles transmission completion of a character that has been sent to the CRT.

The data structures and hardware interface requirements for the Terminal Device Driver, and capabilities of Tracer, are defined in the Software User's Manual.

2.4 ISDN LAYER PROTOCOL TASKS

The ISDN Layer Protocol Tasks provide implementation of the DATA LINK Layer 2 and the NETWORK Layer 3 in accordance with the protocol definitions of the CCITT Specifications. The two Layer Protocol Tasks (the Layer 2 Controller Task and the Layer 3 Controller Task) are designed to satisy the ISDN Basic Rate Interface (BRI) Terminal Equipment requirements. They are independent of user applications and hardware environment. The PHYSICAL Layer 1 implementation is defined in the I/O Device Driver section of this document. Implementation of layers above the NETWORK Layer 3 are specific to user applications. Two such layer tasks are provided, the Demonstration Call Control Task and the Management Entity Task. These tasks are defined in the Application Task section of this document.

The purpose of the Layer 2 Controller Task is to provide the NETWORK Layer 3 with an error free, sequenced data frame service. The Layer 2 Controller Task uses CCITT Specifications Q.921 and X.25 and the primary functional specifications. The Layer 2 Controller Task satisifies the Link Access Procedures for both the D Channel and the B Channel (LAPD and LAPB). Design considerations have also been included for the future implementation of V.120, the new CCITT rate adaption scheme.

The Layer 2 Controller Task's data frame delivery service allows the Layer 3 Controller Task to confidently setup and teardown user voice and data calls on the available facilities. The Layer 3 Controller Task uses CCITT Specification Q.931 as the primary functional specification. Note that the X.25 Layer 3 packet processor task is not included in the initial software package.

The Layer Protocol Tasks require a somewhat non-conventional task architecture in order to simultaneously manage a significant number of multiple logical connections. This event-driven state-machine architecture requires that a state memory block be created and maintained for each logical connection. When a Layer Protocol Task "wakes up" due to the arrival of mail, the message's address is interrogated to determine which logical connection is to receive the mail. The particular logical connection's state block is retrieved and the mail message is processed per the CCITT Specification requirements, depending on the state of the particular logical connection. Typically, processing the mail message results in sending a Primitive message to another task, and updating the logical connection's state block. The Layer Protocol Task then returns to its mail box to pick up any subsequent mail.

The interface between all of the ISDN Layer Tasks is deliberately achieved via the System Mail Utilities. This ensures a distinct, uniform layering mechanism in the event that application programmers wish to replace layers with their own implementations.

2.4.1 Layer 2 Controller Task

The primary job of the ISDN Data Link Layer 2 is to deliver error-free, sequenced data frames to the Network Layer 3. The Layer 2 Controller Task implements the following Layer 2 Link Access Procedures (LAP) for the HPC ISDN Software Package:

- LAPB per the X.25 CCITT Specification.
- LAPD per the Q.921 CCITT Specification.
- V.120 Terminal Adaption capability.

Since the Q.921 LAPD requirements were derived from the X.25 LAPB requirements, most of the same Layer 2 Controller Task routines can be used to implement both LAPB and LAPD. Design considerations have been made to allow future implementation of V.120.

The Layer 2 Controller Task communicates with the Layer 2 DMA/HDLC Controller Device Driver Task and the Management Entity Task, via the System Mail Utilities. These tasks interrogate the mail message headers to determine whether to process the frames using LAPB or LAPD procedures. The

2.0 Functional Description (Continued)

LPAD frames are mailed to the Q.931 Layer 3 Controller Task, while the LAPB frames are mailed to the X.25 Layer 3 Task.

The HPC16400 HDLC hardware handles the Layer 2 HDLC Procedures, which includes bit stuffing, address recognition, and Frame Check Sequence generation and detection. The Layer 2 Controller Task is responsible for the Layer 2 "Data Link Procedure", which includes the following functions:

- Data Transmission
- Protocol Exception Management
- LAPD-Specific Functions.

To accomplish these functions the Layer 2 Controller supports the full set of Layer 2 Peer-to-Peer messages defined in the CCITT Specification Q.921. These messages are listed below and defined further in the Software User's Manual.

UI	Unnumbered Information Frames
UA	Unnumbered Acknowledge
SABM(E)	Set Asynchronous Balanced Mode (Extended)
DISC	Disconnect Command
DM	Disconnect Mode
I	Acknowledged Information Frames
RR	Receiver Ready
RNR	Receiver Not Ready
REJ	Request Recrimination of Frames
FRMR	Unrecoverable Error, Frame Reject

The Layer 2 Controller Task also supports the primitives required to communicate with the other ISDN tasks.

2.4.1.1 Layer 2 Data Transmission

Layer 2 peer-to-peer Data Transmission is supported with two modes: Unacknowledged Data Mode and Multi-Frame Acknowledged Data Mode. The Unacknowledged Data Mode is used primarily for setting up logical connections and for peer-to-peer Management Entity communication. This mode uses the Unnumbered Information (UI) and the Unnumbered Acknowledge (UA) messages. The Multi-Framed Acknowledged Mode is established by the Set Asynchronous Balanced Mode (SABM) command. This mode provides the mechanism for acknowledgement of data frame transport in each direction. The Multi-Frame Acknowledged Mode is terminated with the Disconnect (DISC) command. The response to the DISC message can be either an Unnumbered Acknowledge (UA) message or a Disconnect Mode (DM) message. The actual Layer 2 data frames are transmitted in the Information (I) messages, while in the Multi-Framed Acknowledged Mode.

The Layer 2 Controller is responsible for avoiding message congestion and buffer overflow. A Layer 2 entity can issue the Receive Ready (RR) command to its peer to indicate that it is ready to continue data transmission. Likewise, the Layer 2 Controller can issue the Receiver Not Ready (RNR) command to its peer to indicate that it is not ready for data transmission.

2.4.1.2 Layer 2 Protocol Exception Management

The Layer 2 Controller Task is responsible for handling exceptions to the Data Link Protocol. These exceptions are of two types: recoverable and unrecoverable. Recoverable exceptions in the receive direction are typically failed frames, which are handled by requesting the retransmission of the failed frame with the Reject (REJ) command. Recoverable exceptions in the transmit direction include the expiry of a Layer 2 Timer. Timer expiry requires the retransmission of the frame that was not acknowledged in time, and all subsequent frames. Timer expiry also prompts a message to the Management Entity. Unrecoverable exceptions result in the Frame Reject (FRMR) response. A message to the Management Entity Task is also sent in this case.

2.4.1.3 Layer 2 LAPD-Specific Functions

The following Layer 2 Controller Task functions are LAPD specific. These functions involve establishing and maintaining multiple logical data link connections. Note that a LAPB connection will be maintained as a special independent logical connection.

A two byte address is required for each logical data link. This address is referred to as the Data Link Connection Identifier (DLCI). The DLCI consists of a Service Access Point Identifier (SAPI) and a Terminal Endpoint Identifier (TEI). The Layer 2 Controller Task is responsible for supporting the TEI Assignment Procedure and the TEI Verification Procedure. These procedures are both initiated by the Management Entity. The Layer 2 Controller Task supports both the Automatic and Non-Automatic TEI Assignment Procedures.

Establishment of the LAPD multi-frame acknowledged data transmission mode requires an extended command (SABME) to prompt the peer entity that the frames are intended for a particular logical data connection identified by the accompanying DLCI. The Layer 2 Controller Task maintains each logical link's state and data frames independently, as explained earlier in this section.

The Layer 3 Controller Task addresses and maintains independent logical connections via an identifier called a Connection Endpoint Suffix (CES). Since the CES is different from the Layer 2 Terminal Endpoint Identifier (TEI), a mapping function is required. The Layer 2 Controller Task maintains a CES–TEI translation procedure to properly address Layer 3 logical entities.

2.4.2 Layer 3 Controller Task

The Layer 3 Controller Task implements the application independent portion of the ISDN NETWORK Layer 3 protocol, per the Q.931 CCITT Specification. The primary responsibility of the Layer 3 Controller Task is to establish a network access connection link between a terminal and its peer in the Central Office.

The Layer 3 Controller Task communicates with both the Layer 2 Controller Task and the Call Control Task by sending primitives via the System Mail Utilities. The Layer 3 Controller Task also communicates with the Management Entity Task. The HPC ISDN Layer 3 Controller Task is responsible for the following NETWORK functions:

- Call Establishment and Clearing
- Call Suspension and Resumption
- Call Status and Notification
- Protocol Exception Management.

2.0 Functional Description (Continued)

The Layer 3 Controller Task supports all the Network Layer Peer-to-Peer messages defined in the CCITT Specification Q.931, i.e.:

- Call Establishment and Clearing Messages:

ALERT	Alerting
CALL PROC	Call Proceeding
CONN	Connect
CONN ACK	Connect Acknowledge
INFO	Information
PROG	Progress
SETUP	Setup
SETUP ACK	Setup Acknowledge
DISC	Disconnect
REL	Release
REL COM	Release Complete

- Call Suspension and Resumption Messages

RESUME	Resume
RESUME ACK	Resume Acknowledge
RESUME REJ	Resume Reject
SUSPEND	Suspend
SUSPEND ACK	Suspend Acknowledge
SUSPEND REJ	Suspend Reject

- Miscellaneous Messages

NOTIFY	Notify
STATUS	Status
STATUS EN	Status Enquiry
USER INFO	User Information

2.4.2.1 Call Establishment And Clearing

The Layer 3 Controller Task's primary responsibility is to establish and clear user network connections on available bearer channel facilties. The Q.931 CCITT Specifications include Call Establishment and Clearing of both circuit-switched and packet-switched calls. Initially, the HPC ISDN Software Package only supports circuit-switched call procedures on Basic Rate Interface (BRI) Bearer Channels. The Layer 3 Controller Task is responsible for Call Reference assignment and maintenance. The Layer 3 Controller Task supports Call Establishment using both the Overlap and Non-Overlap (enbloc) addressing modes.

The procedure for establishing and clearing network connections is defined in CCITT Specification Q.931. It is important to note that the Layer 3 Controller Task maintains an associated state block for each network connection. Primitive mail messages arriving at the Layer 3 Controller Task will be interrogated to determine which network connection is to receive the mail. The mail message is processed depending on the state of the network connection. This processing typically includes the transmission of a Primitive to another Layer Task, and the appropriate update of the network connection state block.

2.4.2.2 Call Suspension And Resumption

Call Suspension (SUSPEND) requires that the Bearer Channel facility and the Call Reference for a call be temporarily relinquished. The network connection is left intact, but in the suspend state. The RESUME command reactivates the call by obtaining a Bearer Channel facility and establishing a new Call Reference. The Suspend function is somewhat analogous to the call HOLD feature.

2.4.2.3 Call Status And Notification

The Network can request the status of a network connection at any time via the USER INFO, NOTIFY and STATUS Commands. The information includes Service Validation and Channel Configuration.

2.4.2.4 Layer 3 Protocol Exception Management

The Layer 3 Controller is responsible for handling exceptions to the Network Control Protocol. The primary Layer 3 Controller Task protocol exception is the expiry of the Layer 3 timer. Such an exception requires the retransmission of the particular command and may prompt a message to the Management Entity Task.

2.4.2.5 Timer Support

The Layer 3 Controller supports the following system timers per CCITT Specification Q.931:

T303	SETUP ACK Timer
T305	DISCONNECT ACK Timer
T308	RELEASE ACK Timer
T313	CONNECT ACK Timer

2.4.2.6 SDL Updates

The Layer 3 Controller Task very closely follows the SDL procedures illlustrated in CCITT Specification Q.931, with a few enhancements. These enhancements are listed here and fully defined in the Software User's Manual.

a. Three new SDL States have been added to accommodate establishing the Data Link corresponding to a particular CES. The new states are:

- IDLESTATE
- RELEASEWAIT
- ESTABLISHWAIT

b. The Q.931 NULLSTATE SDL now accepts a new command, CCBROADCASTRESP. This command is sent from the Call Control Task to allow transistion from the NULLSTATE(0) to the CALLPRESENT State(6) during a Network Originated call via the Broadcast mechanism.

2.5 APPLICATION TASKS

The Application Tasks are very dependent on both the terminal equipment configuration and the far-end Network Entity software implementation. The HPC ISDN Software Package includes two sample Application Tasks: the Demonstration Call Control Task and the Management Entity Task. Both of these tasks can be replaced or updated when ported to a particular application. These tasks are included in the HPC ISDN Software Package primarily to verify the operation of the OSI Layer Protocol Tasks and the HPC Device Drivers.

2.5.1 Demonstration Call Control Task

The Demonstration Call Control Task is closely coupled to the specific facilities of an application. The interaction between the Layer 3 Controller Task and Call Control is defined in CCITT Specification Q.931. In the HPC ISDN Application, the Call Control Task communicates with the Layer 3 Controller Task and the Tracer Task. The availability of two circuit switched voice bearer channels is simulated in the Call Control Task. The Call Control Task sends standard Terminal Equipment prompts and messages to the Tracer Task where they are displayed on a UART driven CRT. The Call Control Task has the following responsibilities:

- B Channel Resource Management

2.0 Functional Description (Continued)

- Connection Endpoint Suffix (CES) Allocation
- Conversion between L3 Primitives and Terminal Action.

The Call Control Task and the Layer 3 Controller Task communicate via the NL_DATA_REQ and NL_DATA_IND Primitives. The messages that are supported between these tasks are listed below.

- Commands from Call Control to Layer 3

CC_SETUP_REQ	Setup Request
CC_SETUP_RESP	Setup Response
CC_SETUP_REJ_REQ	Setup Reject
CC_INFO_REQ	Information
CC_DISCONNECT_REQ	Disconnect
CC_RELEASE_REQ	Release
CC_ALERTING_REQ	Alerting
CC_BROADCAST_RESP	Broadcast Response
CC_CALLPROC_REQ	Call Proceeding
CC_PROGRESS_REQ	Progress
CC_NOTIFY_REQ	Notify
CC_RESUME_RQ	Resume
CC_RESUME_REJ	Resume Reject
CC_SUSPEND_REQ	Suspend Request
CC_SUSPEND_REJ	Suspend Reject

- Command from Layer 3 to Call Control

CC_SETUP_IND	Setup
CC_SETUP_CONF	Setup Confirm
CC_SETUP_COMP_IND	Setup Complete
CCI_NFO_IND	Information Indication
CC_ALERTING_IND	Alerting
CC_PROGRESS_IND	Progress
CC_DISCONNECT_IND	Disconnect
CC_RELEASE_IND	Release
CC_CALLPROC_IND	Call Proceeding
CC_RELEASE_CONF	Release Confirm
CC_STATUS_IND	Status Indication
CC_ERORR_IND	Error Indication
CC_RESUME_CONF	Resume Confirm

The Call Control Task also communicates with the Tracer Task using single byte keystroke like commands. These commands are packaged mail messages containing two bytes: the first byte is the Sender Task's ID, the second byte is the keystroke command. The following messages are sent between Call Control and Tracer:

- Keystroke Commands from Tracer to Call Control Task

TR_ON_HOOK	ON Hook
TR_OFF_HOOK	OFF Hook
TR_DIGIT_1	Digit 1
TR_DIGIT_2	Digit 2
TR_DIGIT_3	Digit 3
TR_DIGIT_4	Digit 4
TR_DIGIT_5	Digit 5
TR_DIGIT_6	Digit 6
TR_DIGIT_7	Digit 7
TR_DIGIT_8	Digit 8
TR_DIGIT_9	Digit 9
TR_DIGIT_0	Digit 0
TR_DIGIT_STAR	Digit *
TR_DIGIT_POUND	Digit #

- Commands from Call Control Task to Tracer

TR_IDLE	Idle, ON HOOK
TR_DIALTONE	Dial Tone
TR_DIALING	Dialing
TR_RINGING	Ringing
TR_BUSY	Busy
TR_CONVERSATION	Conversation
TR_RINGBACK	Ringback
TR_ERROR	Error

2.5.2 Management Entity Task

The Management Entity Task is closely coupled to the accompanying Network Management Entity design and to the terminal hardware configuration. Implementation design decisions have been made that make the Management Entity Task unique to a particular application, while still following the general requirements of the CCITT Specifications. Modifications will be required in the Management Entity Task prior to its successful operation in a particular application environment. The Management Entity Task that is included in the HPC ISDN Software Package presumes a particular hardware configuration and Central Office Software implementation.

The Management Entity Task communicates with the Layer 3 Controller Task, the Layer 2 Controller Task, and the Layer 1 Device Driver Task via the System Mail Utilities.

The Management Entity Task has the following responsibilities:

- Initialization of the Terminal Equipment
- Configuration of the Terminal Equipment
- TEI Assignment and Verification
- Multiple Error Notification
- Unrecoverable Error Notification
- Activation/Deactivation of the Terminal Equipment.

2.6 SYSTEM UTILITIES

The system utilities initializes the HPC system upon power-up, and provide support for various machine specific features of the HPC.

2.6.1 Power-Up Reset Main Task

This task is the entry point upon system power-up. The Main Task is responsible for:

— Initializing the general HPC Hardware.

— Initializing the HPC Executive Data Structures.

— Queuing up the Tasks on the Ready Queue.

The Main Task starts with the highest priority, 255. After running, the Main Task has served its purpose and is removed from the system by waiting on a semaphore which is typically never signaled.

2.6.2 Nonmaskable Interrupt (NMI) Handler

Since terminal power is generally a concern, the HPC can go into an idle, low-power mode when the Terminal Equipment is idle. In this mode the HPC is awakened via an NMI, prompted by a local off hook indication, or by a far-end line

2.0 Functional Description (Continued)

signal detection signal from the SID. Conditions for determining when to go in and out of idle mode are application dependent.

2.6.3 Timer Interrupt Handler

The Timer Interrupt Handler fields interrupts from two of the HPC onboard timers. Timer T0, the Watchdog Timer, overflows every 65536 clock counts. When this occurs the Timer Interrupt Handler mails a message to start the Watchdog Task. Timer T1, the ISDN Software Timer, overflows every 10 ms. The ISDN Software Clock is incremented every tenth Timer T1 overflow, resulting in an ISDN Clock with 100 ms resolution, which is used by the Executive Timer facility.

2.6.4 Watchdog Task

A special task is performed by the HPC's watchdog feature to verify system sanity. The Watchdog Task waits for a mail message that is sent by the Timer Interrupt Handler when Timer T0 overflows. This operation requires that the Watchdog Task be regularly scheduled by the HPC Executive. The Watchdog Task is assigned the highest task priority, 255.

3.0 Ordering Information

3.1 LICENSE AGREEMENT

A license agreement is required for the use and sale of the National Semiconductor ISDN Software. Contact your local National Semiconductor field sales office for more information or contact the factory direct at:

National Semiconductor
ISDN Software Support
M/S 16-174
2900 Semiconductor Drive
Santa Clara, CA 95051
(408) 721-5719

3.2 SOFTWARE ORDER INFORMATION

ISDN software is available in either Object or Source Code format. A Demonstration package is also available. Manuals are included with the Demonstration package and with the Executive and Basic Rate Interface Software packages.

Basic Rate Interface (BRI) software is available for several different central office switches. The generic BRI includes a generalized CCITT Switch Interface.

Each BRI Package contains the following modules:

Layer 1 Driver (controls S device)
Layer 2 Driver (controls DMA/HDLC)
Layer 2 Controller (Q.921)
Layer 3 Controller (Q.931 Protocol Control)
Management Entity (Q.921 and Q.931)
Call Control (Demonstration Application)
Tracer (Demonstration and Development Tool)

The Multi-Tasking Executive contains two modules:

Executive Core Module
Executive Interface Module

The Executive Interface Module is always supplied as source code to allow modification to insert application tasks.

A Multi-Tasking Executive is required to run the Basic Rate Interface.

Order Part Number	Description
Multi-Tasking Executive	
HPC-ISDN-EXEC-O	Multi-Tasking Executive Object Code
Basic Rate Interface	
HPC-ISDN-BRI-S	Basic Rate Interface (Generic) Source Code
HPC-ISDN-BRID-S	Basic Rate Interface (DMS-100) Source Code
HPC-ISDN-BRI5-S	Basic Rate Interface (5ESS) Source Code
Demonstration Package	
HPC-ISDN-PCDEMO	ISDN Basic Rate Interface Demonstration (includes Multi-Tasking Executive and Basic Rate Interface Software Manuals)

4.0 Other Related Information

4.1 DEVICE INFORMATION

Additional technical information on devices referenced in this datasheet is available from National:

HPC16400	High Performance microController
HPC16083	High Performance microController
TP3076	COMBO II™
TP3420	CCITT S/T Interface

4.2 DEVELOPMENT SUPPORT INFORMATION

Development tools are available for the HPC Family of Microcontrollers. These tools support the ISDN development environment. ISDN software must be ordered separately.

4.2.1 ISDN Demonstration Kit

A kit is available that demonstrates the software and hardware discussed in this datasheet. Included in this kit is a TP3500 development board featuring the HPC16400, TP3070 COMBO II, TP3420 "SID" and ISDN Basic Rate Interface software in ROM. A complete set of manuals are included. This demonstration kit may be ordered from National, part number.

ISDN-TP3500-Kit

4.2.2 Development Systems

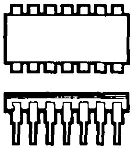
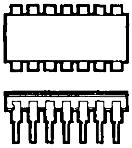
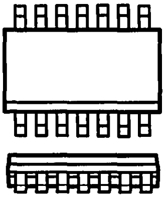
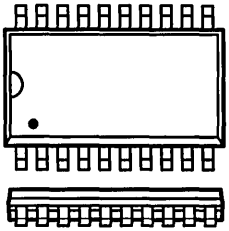
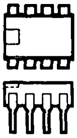
Several different Microcontroller-On-Line-Emulator Development Systems are available for hardware and software development of the HPC Family of Microcontrollers. Complete information on Development Systems and Accessories may be found in the Microcontroller Development Support Datasheet.

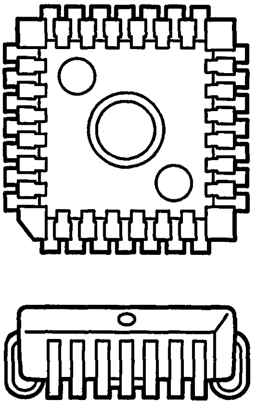
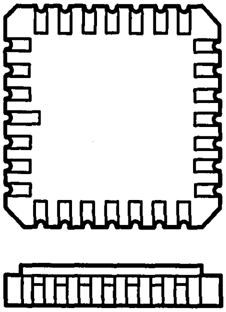
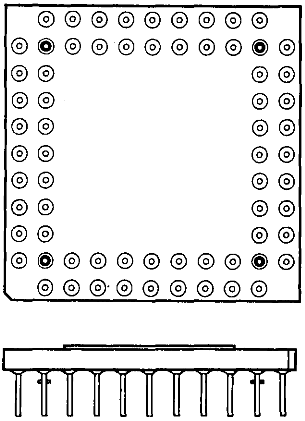
Section 8

Appendices/ Physical Dimensions

Section 8 Contents

Industry Package Cross-Reference Guide

		NSC	Signetics	Intel	Motorola	TI	RCA	Hitachi	NEC
	8-, 14-, 16-, 20- and 28-Lead Glass/Metal DIP	D	I	C	L		D	C	D
	8-, 14- 16-, 20-, 24- and 28-Lead Low Temperature Ceramic DIP	J	F	D	U	J		G	D
	SO (Narrow Body)	M	D		D	D	M	MP	G
	SO (Wide Body)	WM				DW			
	8-, 14- 16-, 20- 24- and 28-Lead Plastic DIP	N	V, A, B	P	P	P, N	E	P	C

		NSC	Signetics	Intel	Motorola	TI	RCA	Hitachi	NEC
	PCC	V	A	N	FN	FN	Q	CP	L
	LCC Leadless Ceramic Chip Carrier	E	G	R	U	FK/ FG/FH	BJ	CG	K
	PGA Ceramic Pin Grid Array	U		CG					

National Semiconductor

Surface Mount

Cost pressures today are forcing many electronics manufacturers to automate their production lines. Surface mount technology plays a key role in this cost-savings trend because:

1. The mounting of devices on the PC board surface eliminates the expense of drilling holes;
2. The use of pick-and-place machines to assemble the PC boards greatly reduces labor costs;
3. The lighter and more compact assembled products resulting from the smaller dimensions of surface mount packages mean lower material costs.

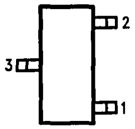
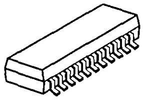
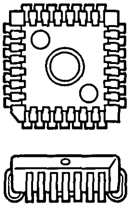
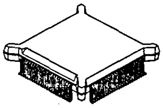
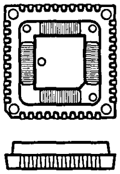
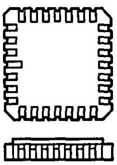
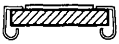
Production processes now permit both surface mount and insertion mount components to be assembled on the same PC board.

SURFACE MOUNT PACKAGING AT NATIONAL

To help our customers take advantage of this new technology, National has developed a line of surface mount packages. Ranging in lead counts from 3 to 360, the package offerings are summarized in Table I.

Lead center spacing keeps shrinking with each new generation of surface mount package. Traditional packages (e.g., DIPs) have a 100 mil lead center spacing. Surface mount packages currently in production (e.g., SOT, SOIC, PCC, LCC, LDCC) have a 50 mil lead center spacing. Surface mount packages in production release (e.g., PQFP) have a 25 mil lead center spacing. Surface mount packages in development (e.g., TAPEPAK®) will have a lead center spacing of only 12–20 mils.

TABLE I. Surface Mount Packages from National

Package Type	Small Outline Transistor (SOT)	Small Outline IC (SOIC)	Plastic Chip Carrier (PCC)	Plastic Quad Flat Pack (PQFP)	TAPEPAK® (TP)	Leadless Chip Carrier (LCC) (LDCC)	Leaded Chip Carrier
							
Package Material	Plastic	Plastic	Plastic	Plastic	Plastic	Ceramic	Ceramic
Lead Bend	Gull Wing	Gull Wing	J-Bend	Gull Wing	Gull Wing	—	Gull Wing
Lead Center Spacing	50 Mils	50 Mils	50 Mils	25 Mils	20, 15, 12 Mils	50 Mils	50 Mils
Tape & Reel Option	Yes	Yes	Yes	tbd	tbd	No	No

Lead Counts	SOT-23
	High Profile
	SOT-23
	Low Profile
		SO-8(*)
		SO-14(*)
		SO-14 Wide(*)
		SO-16(*)
		SO-16 Wide(*)
		SO-20(*)
		SO-24(*)
			PCC-20(*)
			PCC-28(*)
			PCC-44(*)
			PCC-68
			PCC-84
			PCC-124
				PQFP-84
				PQFP-100
				PQFP-132
				PQFP-196(*)
				PQFP-244
					TP-40 (*)
					TP-68
					TP-84
					TP-132
					TP-172
					TP-220
					TP-284
					TP-360
						LCC-18
						LCC-20(*)
						LCC-28
						LCC-32
						LCC-44 (*)
						LCC-48
						LCC-52
						LCC-68
						LCC-84
						LCC-124
							LDCC-44
							LDCC-68
							LDCC-84
							LDCC-124

*In production (or planned) for linear products.

LINEAR PRODUCTS IN SURFACE MOUNT

Linear functions available in surface mount include:

- Op amps
- Comparators
- Regulators
- References
- Data conversion
- Industrial
- Consumer
- Automotive

A complete list of linear part numbers in surface mount is presented in Table III. Refer to the datasheet in the appropriate chapter of this databook for a complete description of the device. In addition, National is continually expanding the list of devices offered in surface mount. If the functions you need do not appear in Table III, contact the sales office or distributor branch nearest you for additional information.

Automated manufacturers can improve their cost savings by using Tape-and-Reel for surface mount devices. Simplified handling results because hundreds-to-thousands of semiconductors are carried on a single Tape-and-Reel pack (see ordering and shipping information—printed later in this section—for a comparison of devices/reel vs. devices/rail for those surface mount package types being used for linear products). With this higher device count per reel (when compared with less than a 100 devices per rail), pick-and-place machines have to be re-loaded less frequently and lower labor costs result.

With Tape-and-Reel, manufacturers save twice—once from using surface mount technology for automated PC board assembly and again from less device handling during shipment and machine set-up.

BOARD CONVERSION

Besides new designs, many manufacturers are converting existing printed circuit board designs to surface mount. The resulting PCB will be smaller, lighter and less expensive to manufacture; but there is one caveat—be careful about the thermal dissipation capability of the surface mount package.

Because the surface mount package is smaller than the traditional dual-in-line package, the surface mount package is not capable of conducting as much heat away as the DIP (i.e., the surface mount package has a higher thermal resistance—see Table II).

The silicon for most National devices can operate up to a 150°C junction temperature (check the datasheet for the rare exception). Like the DIP, the surface mount package can actually withstand an ambient temperature of up to 125°C (although a commercial temperature range device will only be specified for a max ambient temperature of 70°C and an industrial temperature range device will only be specified for a max ambient temperature of 85°C). See AN-336, "Understanding Integrated Circuit Package Power Capabilities", (reprinted in the appendix of each linear databook volume) for more information.

TABLE II: Surface Mount Package Thermal Resistance Range*

Package	Thermal Resistance** (θ_{jA}, °C/W)
SO-8	120–175
SO-14	100–140
SO-14 Wide	70–110
SO-16	90–130
SO-16 Wide	70–100
SO-20	60–90
SO-24	55–85
PCC-20	70–100
PCC-28	60–90
PCC-44	40–60

*Actual thermal resistance for a particular device depends on die size. Refer to the datasheet for the actual θ_{jA} value.

**Test conditions: PCB mount (FR4 material), still air (room temperature), copper traces (150 × 20 × 10 mils).

Given a max junction temperature of 150°C and a maximum allowed ambient temperature, the surface mount device will be able to dissipate less power than the DIP device. This factor must be taken into account for new designs.

For board conversion, the DIP and surface mount devices would have to dissipate the same power. This means the surface mount circuit would have a lower maximum allowable ambient temperature than the DIP circuit. For DIP circuits where the maximum ambient temperature required is substantially lower than the maximum ambient temperature allowed, there may be enough margin for safe operation of the surface mount circuit with its lower maximum allowable ambient temperature. But where the maximum ambient temperature required of the DIP current is close to the maximum allowable ambient temperature, the lower maximum ambient temperature allowed for the surface mount circuit may fall below the maximum ambient temperature required. The circuit designer must be aware of this potential pitfall so that an appropriate work-around can be found to keep the surface mount package from being thermally overstressed in the application.

SURFACE MOUNT LITERATURE

National has published extensive literature on the subject of surface mount packaging. Engineers from packaging, quality, reliability, and surface mount applications have pooled their experience to provide you with practical hands-on knowledge about the construction and use of surface mount packages.

The applications note AN-450 "Surface Mounting Methods and their Effect on Product Reliability" is referenced on each SMD datasheet. In addition, "Wave Soldering of Surface Mount Components" is reprinted in this section for your information.

TABLE III. Linear Surface Mount Current Device Listing

Amplifiers and Comparators

Part Number	Part Number
LF347WM	LM392M
LF351M	LM393M
LF451CM	LM741CM
LF353M	LM1458M
LF355M	LM2901M
LF356M	LM2902M
LF357M	LM2903M
LF444CWM	LM2904M
LM10CWM	LM2924M
LM10CLWM	LM3403M
LM308M	LM4250M
LM308AM	LM324M
LM310M	LM339M
LM311M	LM365WM
LM318M	LM607CM
LM319M	LMC669BCWM
LM324M	LMC669CCWM
LM339M	LF441CM
LM346M	
LM348M	
LM358M	
LM359M	

Regulators and References

Part Number	Part Number
LM317LM	LM2931M-5.0
LF3334M	LM3524M
LM336M-2.5	LM78L05ACM
LF336BM-2.5	LM78L12ACM
LM336M-5.0	LM78L15ACM
LM336BM-5.0	LM79L05ACM
LM337LM	LM79L12ACM
LM385M	LM79L15ACM
LM385M-1.2	LP2951ACM
LM385BM-1.2	LP2951CM
LM385M-2.5	
LM385BM-2.5	
LM723CM	
LM2931CM	

Data Acquisition Circuits

Part Number	Part Number
ADC0802LCV	ADC1025BCV
ADC0802LCWM	ADC1025CCV
ADC0804LCV	DAC0800LCM
ADC0804LCWM	DAC0801LCM
ADC0808CCV	DAC0802LCM
ADC0809CCV	DAC0806LCM
ADC0811BCV	DAC0807LCM
ADC0811CCV	DAC0808LCM
ADC0819BCV	DAC0830LCWM
ADC0819CCV	DAC0830LCV
ADC0820BCV	DAC0832LCWM
ADC0820CCV	DAC0832LCV
ADC0838BCV	
ADC0838CCV	
ADC0841BCV	
ADC0841CCV	
ADC0848BCV	
ADC0848CCV	
ADC1005BCV	
ADC1005CCV	

Industrial Functions

Part Number	Part Number
AH5012CM	LM13600M
LF13331M	LM13700M
LF13509M	LMC555CM
LF13333M	LM567CM
LM555CM	MF4CWM-50
LM556CM	MF4CWM-100
LM567CM	MF6CWM-50
LM1496M	MF10CCWM
LM2917M	MF6CWM-100
LM3046M	MF5CWM
LM3086M	
LM3146M	

Commercial and Automotive

Part Number	Part Number
LM386M-1	LM1837M
LM592M	LM1851M
LM831M	LM1863M
LM832M	LM1865M
LM833M	LM1870M
LM837M	LM1894M
LM838M	LM1964V
LM1131CM	LM2893M
	LM3361AM
	LM1881M

Hybrids

Part Number	Part Number
LH0002E	LH0032E
LH4002E	LH0033E

A FINAL WORD

National is a world leader in the design and manufacture of surface mount components.

Because of design innovations such as perforated copper leadframes, our small outline package is as reliable as our DIP—the laws of physics would have meant that a straight "junior copy" of the DIP would have resulted in an "S.O." package of lower reliability. You benefit from this equivalence of reliability. In addition, our ongoing vigilance at each step of the production process assures that the reliability we designed in stays in so that only devices of the highest quality and reliability are shipped to your factory.

Our surface mount applications lab at our headquarters site in Santa Clara, California continues to research (and publish) methods to make it even easier for you to use surface mount technology. Your problems are our problems.

When you think "Surface Mount"—think "National"!

Ordering and Shipping Information

When you order a surface mount semiconductor, it will be in one of the several available surface mount package types. Specifying the Tape-and-Reel method of shipment means that you will receive your devices in the following quantities per Tape-and-Reel pack: SMD devices can also be supplied in conventional conductive rails.

Package	Package Designator	Max/Rail	Per Reel*
SO-8	M	100	2500
SO-14	M	50	2500
SO-14 Wide	WM	50	1000
SO-16	M	50	2500
SO-16 Wide	WM	50	1000
SO-20	M	40	1000
SO-24	M	30	1000
PCL-20	V	50	1000
PCL-28	V	40	1000
PCL-44	V	25	500
PQFP-196	VF	TBD	—
TP-40	TP	100	TBD
LCC-20	E	50	—
LCC-44	E	25	—

*Incremental ordering quantities. (National Semiconductor reserves the right to provide a smaller quantity of devices per Tape-and-Reel pack to preserve lot or date code integrity. See example below.)

Example: You order 5,000 LM324M ICs shipped in Tape-and-Reel.

- Case 1: All 5,000 devices have the same date code
  - You receive 2 SO-14 (Narrow) Tape-and-Reel packs, each having 2500 LM324M ICs
- Case 2: 3,000 devices have date code A and 2,000 devices have date code B
  - You receive 3 SO-14 (Narrow) Tape-and-Reel packs as follows:

    Pack #1 has 2,500 LM324M ICs with date code A

    Pack #2 has 500 LM324M ICs with date code A

    Pack #3 has 2,000 LM324M ICs with date code B

Short-Form Procurement Specification

TAPE FORMAT

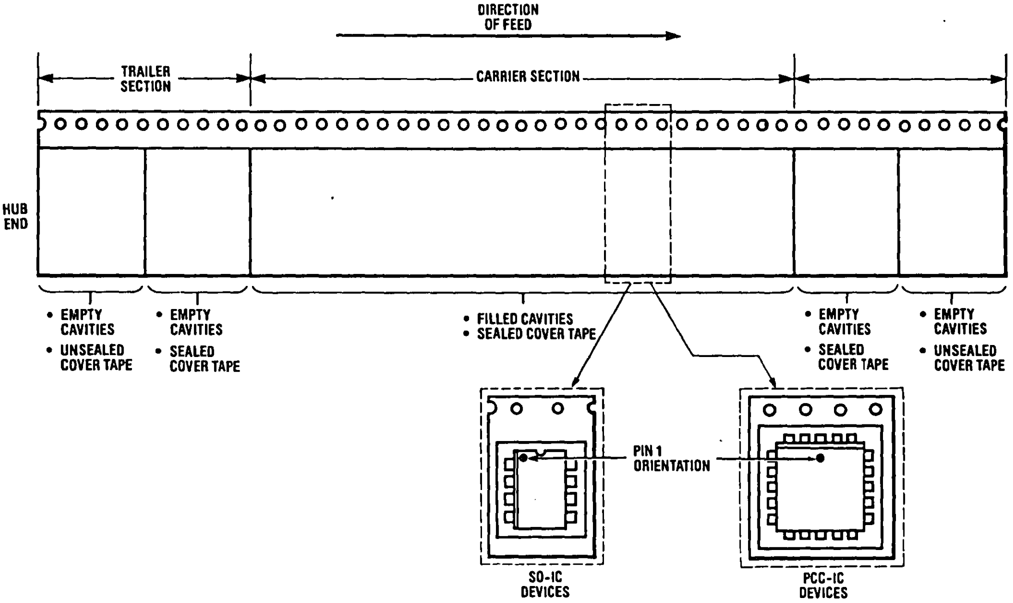
→ Direction of Feed

	Trailer (Hub End)*		Carrier*	Leader (Start End)*	
	Empty Cavities, min (Unsealed Cover Tape)	Empty Cavities, min (Sealed Cover Tape)	Filled Cavities (Sealed Cover Tape)	Empty Cavities, min (Sealed Cover Tape)	Empty Cavities, min (Unsealed Cover Tape)
Small Outline IC					
SO-8 (Narrow)	2	2	2500	5	5
SO-14 (Narrow)	2	2	2500	5	5
SO-14 (Wide)	2	2	1000	5	5
SO-16 (Narrow)	2	2	2500	5	5
SO-16 (Wide)	2	2	1000	5	5
SO-20 (Wide)	2	2	1000	5	5
SO-24 (Wide)	2	2	1000	5	5
Plastic Chip Carrier IC					
PCC-20	2	2	1000	5	5
PCC-28	2	2	750	5	5
PCC-44	2	2	500	5	5

***The following diagram identifies these sections of the tape and Pin #1 device orientation.**

Short-Form Procurement Specification (Continued)

DEVICE ORIENTATION

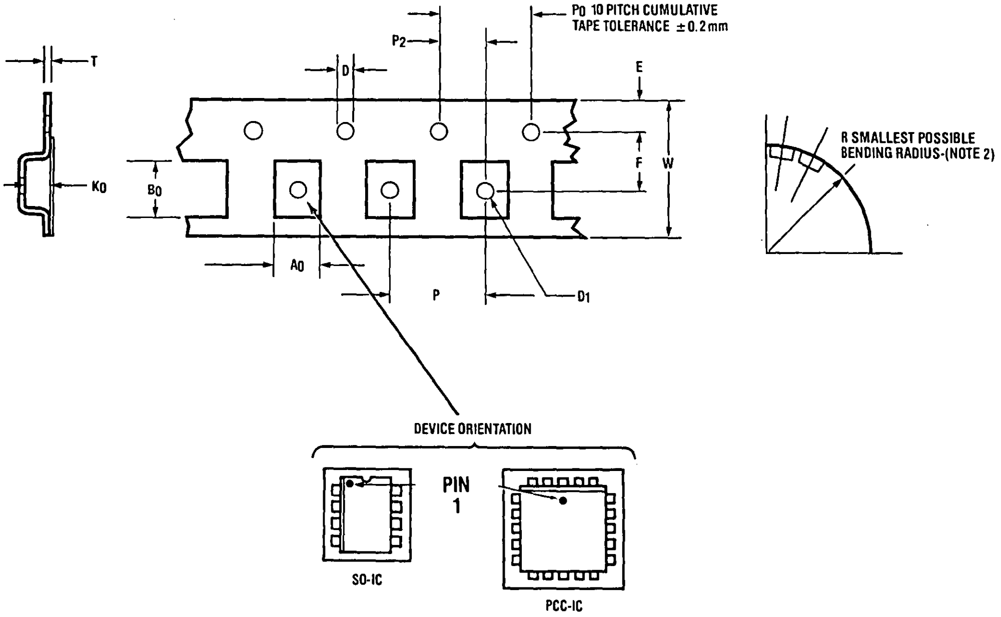


TL/XX/0026-8

MATERIALS

- Cavity Tape: Conductive PVC (less than 10^5 Ohms/Sq)
- Cover Tape: Polyester
  (1) Conductive cover available
- Reel:
  (1) Solid 80 pt fibreboard (standard)
  (2) Conductive fibreboard available
  (3) Conductive plastic (PVC) available

TAPE DIMENSIONS (24 Millimeter Tape or Less)


P0 10 PITCH CUMULATIVE TAPE TOLERANCE ±0.2mm
P2
T
D
E
F
W
K0
B0
A0
P
D1
R SMALLEST POSSIBLE BENDING RADIUS-(NOTE 2)
DEVICE ORIENTATION
PIN 1
SO-IC
PCC-IC

TL/XX/0026-9

Short-Form Procurement Specification (Continued)

	W	P	F	E	P_2	P_0	D	T	A_0	B_0	K_0	D_1	R
Small Outline IC													
SO-8 (Narrow)	12±.30	8.0±.10	5.5±.05	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	6.4±.10	5.2±.10	2.1±.10	1.55±.05	30
SO-14 (Narrow)	16±.30	8.0±.10	7.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	6.5±.10	9.0±.10	2.1±.10	1.55±.05	40
SO-14 (Wide)	16±.30	12.0±.10	7.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	10.9±.10	9.5±.10	3.0±.10	1.55±.05	40
SO-16 (Narrow)	16±.30	8.0±.10	7.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	6.5±.10	10.3±.10	2.1±.10	1.55±.05	40
SO-16 (Wide)	16±.30	12.0±.10	7.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	10.9±.10	10.76±.10	3.0±.10	1.55±.05	40
SO-20 (Wide)	24±.30	12.0±.10	11.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	10.9±.10	13.3±.10	3.0±.10	2.05±.05	50
SO-24 (Wide)	24±.30	12.0±.10	11.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	10.9±.10	15.85±.10	3.0±.10	2.05±.05	50
Plastic Chip Carrier IC													
PCC-20	16±.30	12.0±.10	7.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	9.3±.10	9.3±.10	4.9±.10	1.55±.05	40
PCC-28	24±.30	16.0±.10	11.5±.10	1.75±.10	2.0±.05	4.0±.10	1.55±.05	.30±.10	13.0±.10	13.0±.10	4.9±.10	2.05±.05	50

Note 1: A_0, B_0 and K_0 dimensions are measured 0.3 mm above the inside wall of the cavity bottom.

Note 2: Tape with components shall pass around a mandril radius R without damage.

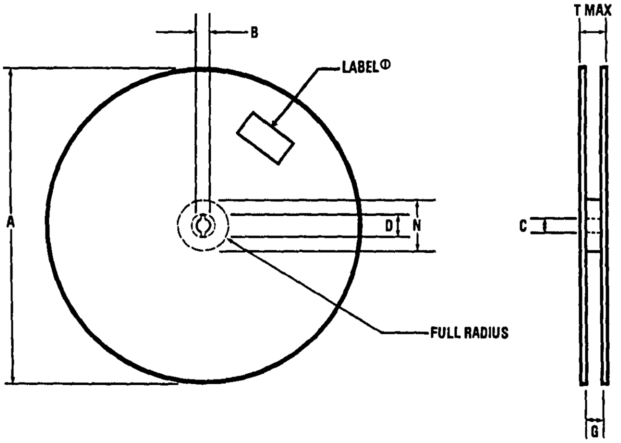
Note 3: Cavity tape material shall be PVC conductive (less than 10^5 Ohms/Sq).

Note 4: Cover tape material shall be polyester (30–65 grams peel-back force).

Note 5: D_1 Dimension is centered within cavity.

Note 6: All dimensions are in millimeters.

REEL DIMENSIONS



TL/XX/0026-10

STAR™* Surface Mount Tape and Reel

Short-Form Procurement Specifications (Continued)

		A (Max)	B (Min)	C	D (Min)	N (Min)	G	T (Max)
12 mm Tape	SO-8 (Narrow)	(13.00) (330)	.059 1.5	.512±.002 13±0.05	.795 20.2	1.969 50	$0.488^{+.078}_{-.000}$ 12.4^{+2}_{-0}	.724 18.4
16 mm Tape	SO-14 (Narrow) SO-14 (Wide) SO-16 (Narrow) SO-16 (Wide) PCC-20	(13.00) (330)	.059 1.5	.512±.002 13±0.05	.795 20.2	1.969 50	$0.646^{+.078}_{-.000}$ 16.4^{+2}_{-0}	.882 22.4
24 mm Tape	SO-20 (Wide) SO-24 (Wide) PCC-28	(13.00) (330)	.059 1.5	.512±.002 13±0.05	.795 20.2	1.969 50	$0.960^{+.078}_{-.000}$ 24.4^{+2}_{-0}	1.197 30.4
32 mm Tape	PCC-44	(13.00) (330)	.059 1.5	.512±.002 13±0.05	.795 20.2	1.969 50	$1.276^{+.078}_{-.000}$ 32.4^{+2}_{-0}	1.512 38.4

Units: $\frac{\text{Inches}}{\text{Millimeters}}$

Material: Paperboard (Non-Flaking)

LABEL

Human and Machine Readable Label is provided on reel. A variable (C.P.I) density code 39 is available. NSC STD label (7.6 C.P.I.)

FIELD

Lot Number

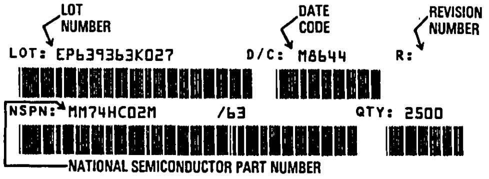
Date Code

Revision Level

National Part No. I.D.

Qty.

EXAMPLE



LOT: EP639363K027 D/C: M8644 R:

NSPN: MM74HC02M /63 QTY: 2500

TL/XX/0026-11

Fields are separated by at least one blank space.

Future Tape-and-Reel packs will also include a smaller-size bar code label (high-density code 39) at the beginning of the tape. (This tape label is not available on current production.)

National Semiconductor will also offer additional labels containing information per your specific specification.

Wave Soldering of Surface Mount Components

ABSTRACT

In facing the upcoming surge of "surface mount technology", many manufacturers of printed circuit boards have taken steps to convert some portions of their boards to this new process. However, as the availability of surface mount components is still limited, may have taken to mixing the lead-inserted standard dual-in-line packages (DIPs) with the surface mounted devices (SMDs). Furthermore, to take advantage of using both sides of the board, surface-mounted components are generally adhered to the bottom side of the board while the top side is reserved for the conventional lead-inserted packages. If processed through a wave solder machine, the semiconductor components are now subjected to extra thermal stresses (now that the components are totally immersed into the molten solder).

A discussion of the effect of wave soldering on the reliability of plastic semiconductor packages follows. This is intended to highlight the limitations which should be understood in the use of wave soldering of surface mounted components.

ROLE OF WAVE-SOLDERING IN APPLICATION OF SMDs

The generally acceptable methods of soldering SMDs are vapor phase reflow soldering and IR reflow soldering, both requiring application of solder paste on PW boards prior to placement of the components. However, sentiment still exists for retaining the use of the old wave-soldering machine.

Wave Soldering of Surface Mount Components (Continued)

The reasons being:

1) Most PC Board Assembly houses already possess wave soldering equipment. Switching to another technology such as vapor phase soldering requires substantial investment in equipment and people.
2) Due to the limited number of devices that are surface mount components, it is necessary to mix both lead inserted components and surface mount components on the same board.
3) Some components such as relays and switches are made of materials which would not be able to survive the temperature exposure in a vapor phase or IR furnace.

PW BOARD ASSEMBLY PROCEDURES

There are two considerations in which through-hole ICs may be combined with surface mount components on the PW Board:

a) Whether to mount ICs on one or both sides of the board.
b) The sequence of soldering using Vapor Phase, IR or Wave Soldering singly or combination of two or more methods.

   The various processes that may be employed are:

   A) Wave Solder before Vapor/IR reflow solder.

      1. Components on the same side of PW Board.

         Lead insert standard DIPS onto PW Board Wave solder (conventional)

         Wash and lead trim

         Dispense solder paste on SMD pads

         Pick and place SMDs onto PW Board

         Bake

         Vapor phase/IR reflow

         Clean

      2. Components on opposite side of PW Board.

         Lead insert standard DIPs onto PW Board

         Wave Solder (conventional)

         Clean and lead trim

         Invert PW Board

         Dispense solder paste on SMD pads

         Dispense drop of adhesive on SMD sites (optional for smaller components)

         Pick and place SMDs onto board

         Bake/Cure

         Invert board to rest on raised fixture

         Vapor/IR reflow soldering

         Clean

   B) Vapor/IR reflow solder then Wave Solder.

      1. Components on the same side of PW Board.

         Solder paste screened on SMD side of Printed Wire Board

         Pick and place SMDs

         Bake

         Vapor/IR reflow

         Lead insert on same side as SMDs

         Wave solder

         Clean and trim underside of PCB

   C) Vapor/IR reflow only.

      1. Components on the same side of PW Board.

         Trim and form standard DIPs in "gull wing" configuration

         Solder paste screened on PW Board

         Pick and place SMDs and DIPs

         Bake

         Vapor/IR reflow

         Clean

      2. Components on opposite sides of PW Board.

         Solder paste screened on SMD-side of Printed Wire Board

         Adhesive dispensed at central location of each component

         Pick and place SMDs

         Bake

         Solder paste screened on all pads on DIP-side or alternatively apply solder rings (performs) on leads

         Lead insert DIPs

         Vapor/IR reflow

         Clean and lead trim

   D) Wave Soldering Only

      1. Components on opposite sides of PW Board.

         Adhesive dispense on SMD side of PW Board

         Pick and place SMDs

         Cure adhesive

         Lead insert top side with DIPs

         Wave solder with SMDs down and into solder bath

         Clean and lead trim

All of the above assembly procedures can be divided into three categories for I.C. Reliability considerations:

1) Components are subjected to both a vapor phase/IR heat cycle then followed by a wave-solder heat cycle or vice versa.
2) Components are subjected to only a vapor phase/IR heat cycle.
3) Components are subjected to wave-soldering only and SMDs are subjected to heat by immersion into a solder pot.

Of these three categories, the last is the most severe regarding heat treatment to a semiconductor device. However, note that semiconductor molded packages generally possess a coating of solder on their leads as a final finish for solderability and protection of base leadframe material. Most semiconductor manufacturers solder-plate the component leads, while others perform hot solder dip. In the latter case the packages may be subjected to total immersion into a hot solder bath under controlled conditions (manual operation) or be partially immersed while in a 'pallet' where automatic wave or DIP soldering processes are used. It is, therefore, possible to subject SMDs to solder heat under certain conditions and not cause catastrophic failures.

Wave Soldering of Surface Mount Components (Continued)

THERMAL CHARACTERISTICS OF MOLDED INTEGRATED CIRCUITS

Since Plastic DIPs and SMDs are encapsulated with a thermoset epoxy, the thermal characteristics of the material generally correspond to a TMA (Thermo-Mechanical Analysis) graph. The critical parameters are (a) its Linear thermal expansion characteristics and (b) its glass transition temperature after the epoxy has been fully cured. A typical TMA graph is illustrated in *Figure 1*. Note that the epoxy changes to a higher thermal expansion once it is subjected to temperatures exceeding its glass transition temperature. Metals (as used on lead frames, for example) do not have this characteristic and generally will have a consistent Linear thermal expansion over the same temperature range.

In any good reliable plastic package, the choice of lead frame material should be such to match its thermal expansion properties to that of the encapsulating epoxy. In the event that there is a mismatch between the two, stresses can build up at the interface of the epoxy and metal. There now exists a tendency for the epoxy to separate from the metal lead frame in a manner similar to that observed on bimetallic thermal range.

In most cases when the packages are kept at temperatures below their glass transition, there is a small possibility of separation at the expoxy-metal interface. Howerver, if the package is subjected to temprature above its glass-transition temperature, the epoxy will begin to expand much faster than the metal and the probability of separation is greatly increased.

CONVENTIONAL WAVE-SOLDERING

Most wave-soldering operations occur at temperatures between 240–260°C. Conventional epoxies for encapsulation have glass-transition temperature between 140–170°C. An I.C. directly exposed to these temperatures risks its long term functionality due to epoxy/metal separation.

Fortunately, there are factors that can reduce that element of risk:

1) The PW board has a certain amount of heat-sink effect and tends to shield the components from the temperature of the solder (if they were placed on the top side of the board). In actual measurements, DIPs achieve a temperature between 120–150°C in a 5-second pass over the solder. This accounts for the fact that DIPs mounted in the conventional manner are reliable.

2) In conventional soldering, only the tip of each lead in a DIP would experience the solder temperature because the epoxy and die are standing above the PW board and out of the solder bath.

EFFECT ON PACKAGE PERFORMANCE BY EPOXY-METAL SEPARATION

In wave soldering, it is necessary to use fluxes to assist the solderability of the components and PW boards. Some facilities may even process the boards and components through some form of acid cleaning prior to the soldering temperature. If separation occurs, the flux residues and acid residues (which may be present owing to inadequate cleaning) will be forced into the package mainly by capillary action as the residues move away from the solder heat source. Once the package is cooled, these contaminants are now trapped within the package and are available to diffuse with moisture from the epoxy over time. It should be noted that electrical tests performed immediately after soldering generally will give no indication of this potential problem. In any case, the end result will be corrosion of the chip metallization over time and premature failure of the device in the field.

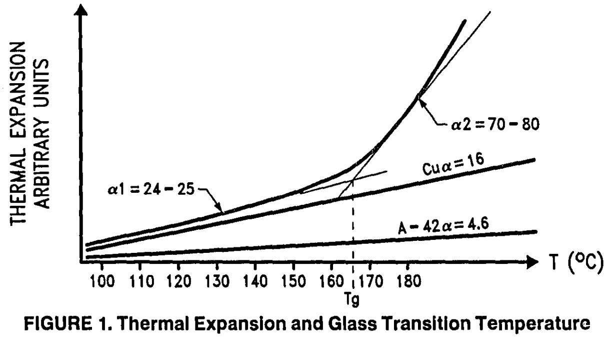
VAPOR PHASE/IR REFLOW SOLDERING

In both vapor phase and IR reflow soldering, the risk of separation between epoxy/metal can also be high. Operating temperatures are 215°C (vapor phase) or 240°C (IR) and duration may also be longer (30 sec–60 sec). On the same theoretical basis, there should also be separation. However, in both these methods, solder paste is applied to the pads of the boards; no fluxes are used. Also, the devices are not immersed into the hot solder. This reduces the possibility of solder forcing itself into the epoxy-lead frame interface. Furthermore, in the vapor phase system, the soldering environment is "oxygen-free" and considered "contaminant free". Being so, it could be visualized that as far as reliability with respect to corrosion, both of these methods are advantageous over wave soldering.

BIAS MOISTURE TEST

A bias moisture test was designed to determine the effect on package performance. In this test, the packages are pressured in a stream chamber to accelerate penetration of moisture into the package. An electrical bias is applied on the device. Should there be any contaminants trapped within the package, the moisture will quickly form an electrolyte and cause the electrodes (which are the lead fingers), the gold wire and the aluminum bond-pads of the silicon device to corrode. The aluminum bond-pads, being the weakest link of the system, will generally be the first to fail.

This proprietary accelerated bias/moisture pressure-test is significant in relation to the life test condition at 85°C and



TL/XX/0026-12

FIGURE 1. Thermal Expansion and Glass Transition Temperature

Wave Soldering of Surface Mount Components (Continued)

85% relative humidity. Once cycle of approximately 100 hours has been shown to be equivalent to 2000 hours in the 85/85 condition. Should the packages start to fail within the first cycle in the test, it is anticipated that the boards with these components in the harsh operating environment (85°C/85% RH) will experience corrosion and eventual electrical failures within its first 2000 hours of operation.

Whether this is significant to a circuit board manufacturer will obviously be dependent on the products being manufactured and the workmanship or reliability standards. Generally in systems with a long warranty and containing many components, it is advisable both on a reputation and cost basis to have the most reliable parts available.

TEST RESULTS

The comparison of vapor phase and wave-soldering upon the reliability of molded Small-Outline packages was performed using the bias moisture test (see Table IV). It is clearly seen that vapor phase reflow soldering gave more consistent results. Wave-soldering results were based on manual operation giving variations in soldering parameters such as temperature and duration.

TABLE IV. Vapor Phase vs. Wave Solder

1. Vapor phase (60 sec. exposure @ 215°C)
   = 9 failures/1723 samples
   = 0.5% (average over 32 sample lots)
2. Wave solder (2 sec total immersion @ 260°C)
   = 16 failures/1201 samples
   = 1.3% (average over 27 sample lots)

Package: SO-14 lead
Test: Bias moisture test 85% R.H.,
85°C for 2000 hours
Device: LM324M

In Table V we examine the tolerance of the Small-Outlined (SOIC) package to varying immersion time in a hot solder pot. SO-14 lead molded packages were subjected to the bias moisture test after being treated to the various soldering conditions and repeated four (4) times. End point was an electrical test after an equivalent of 4000 hours 85/85 test. Results were compared for packages by itself against packages which were surface-mounted onto a FR-4 printed wire board.

TABLE V. Summary of Wave Solder Results (85% R.H./85°C Bias Moisture Test, 2000 hours) (# Failures/Total Tested)

	Unmounted	Mounted
Control/Vapor Phase 15 sec @ 215°C	0/114	0/84
Solder Dip 2 sec @ 260°C	2/144 (1.4%)	0/85
Solder Dip 4 sec @ 260°C	—	0/83
Solder Dip 6 sec @ 260°C	13/248 (5.2%)	1/76 (1.3%)
Solder Dip 10 sec @ 260°C	14/127 (11.0%)	3/79 (3.8%)
Package: SO-14 lead Device: LM324M		

Since the package is of very small mass and experiences a rather sharp thermal shock followed by stresses created by the mismatch in expansion, the results show the package being susceptible to failures after being immersed in excess of 6 seconds in a solder pot. In the second case where the packages were mounted, the effect of severe temperature excursion was reduced. In the second case where the packages were mounted, the effect of severe temperature excursion was reduced. In any case, because of the repeated treatment, the package had failures when subjected in excess of 6 seconds immersion in hot solder. The safety margin is therefore recommended as maximum 4 seconds immersion. If packages were immersed longer than 4 seconds, there is a probable chance of finding some long term reliability failures even though the immediate electrical test data could be acceptable.

Finally, Table VI examines the bias moisture test performed on surface mount (SOIC) components manufactured by various semiconductor houses. End point was an electrical test after an equivalent of 6000 hours in a 85/85 test. Failures were analyzed and corrosion was checked for in each case to detect flaws in package integrity.

TABLE VI. U.S. Manufacturers Integrated Circuits Reliability in Various Solder Environments (# Failure/Total Tested)

Package SO-8	Vapor Phase 30 sec	Wave Solder 2 sec	Wave Solder 4 sec	Wave Solder 6 sec	Wave Solder 10 sec
Manuf A	8/30*	1/30*	0.30	12/30*	16/30*
Manuf B	2/30*	8/30*	2/30*	22/30*	20/30*
Manuf C	0/30	0/29	0/29	0/30	0/30
Manuf D	1/30*	0/30	12/30*	14/30*	2/30*
Manuf E	1/30**	0/30	0/30	0/30	0/30
Manuf F	0/30	0/30	0/30	0/30	0/30
Manuf G	0/30	0/30	0/30	0/30	0/30

*Corrosion-failures

**No Visual Defects—Non-corrosion failures

Test: Accelerated Bias Moisture Test; 85% R.H./85°C, 6000 equivalent hours.

SUMMARY

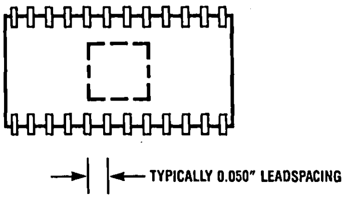
Based on the results presented, it is noted that surface-mounted components are as reliable as standard molded DIP packages. Whereas DIPs were never processed by being totally immersed in a hot solder wave during printed circuit board soldering, surface mounted components such as SOICs (Small Outline) are expected to survive a total immersion in the hot solder in order to capitalize on maximum population on boards. Being constructed from a thermoset plastic of relatively low Tg compared to the soldering temperature, the ability of the package to survive is dependent on the time of immersion and also the cleanliness of material. The results indicate that one should limit the immersion time of package in the solder wave to a maximum of 4 seconds in order to truly duplicate the reliability of a DIP. As the package size is reduced, as in a SO-8 lead, the requirement becomes even more critical. This is shown by the various manufacturers' performance. Results indicate there is room for improvement since not all survived the hot solder immersion without compromise to lower reliability.

Small Outline (SO) Package Surface Mounting Methods— Parameters and Their Effect on Product Reliability

The SO (small outline) package has been developed to meet customer demand for ever-increasing miniaturization and component density.

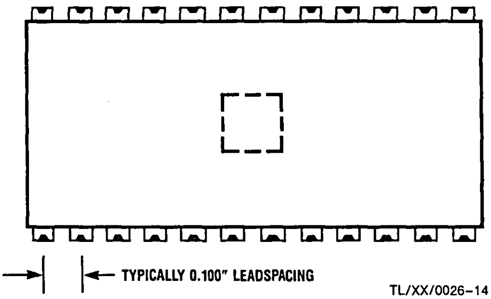
COMPONENT SIZE COMPARISON

S.O. Package



TL/XX/0026-13

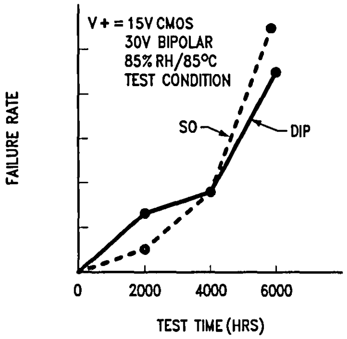
Standard DIP Package



TL/XX/0026-14

Because of its small size, reliability of the product assembled in SO packages needs to be carefully evaluated.

SO packages at National were internally qualified for production under the condition that they be of comparable reliability performance to a standard dual in line package under all accelerated environmental tests. *Figure A* is a summary of accelarated bias moisture test performance on 30V bipolar and 15V CMOS product assembled in SO and DIP (control) packages.



TL/XX/0026-15

FIGURE A

In order to achieve reliability performance comparable to DIPs—SO packages are designed and built with materials and processes that effectively compensate for their small size.

All SO packages tested on 85%RA, 85°C were assembled on PC conversion boards using vapor-phase reflow soldering. With this approach we are able to measure the effect of surface mounting methods on reliability of the process. As illustrated in *Figure A* no significant difference was detected between the long term reliability performance of surface mounted S.O. packages and the DIP control product for up to 6000 hours of accelerated 85%/85°C testing.

SURFACE-MOUNT PROCESS FLOW

The standard process flowcharts for basic surface-mount operation and mixed-lead insertion/surface-mount operations, are illustrated on the following pages.

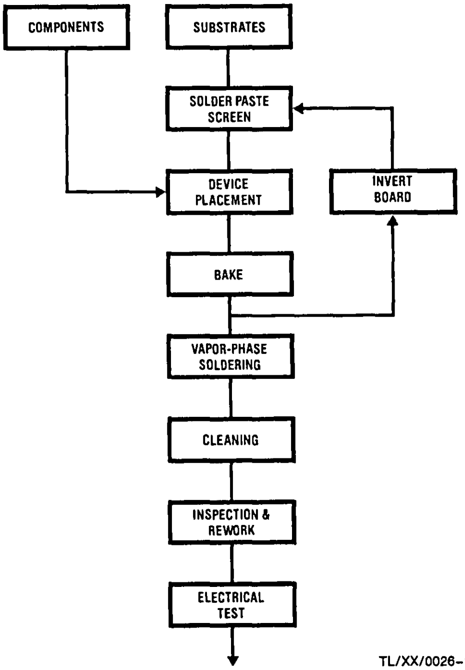
Usual variations encountered by users of SO packages are:

- Single-sided boards, surface-mounted components only.
- Single-sided boards, mixed-lead inserted and surface-mounted components.
- Double-sided boards, surface-mounted components only.
- Double-sided boards, mixed-lead inserted and surface-mounted components.

In consideration of these variations, it became necessary for users to utilize techniques involving wave soldering and adhesive applications, along with the commonly-used vapor-phase solder reflow soldering technique.

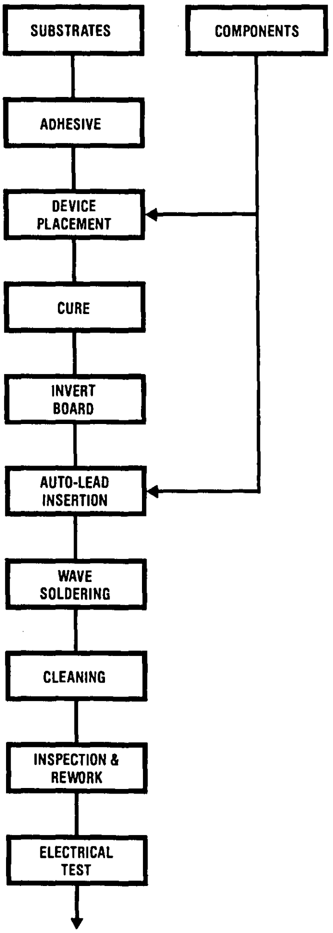
PRODUCTION FLOW

Basic Surface-Mount Production Flow



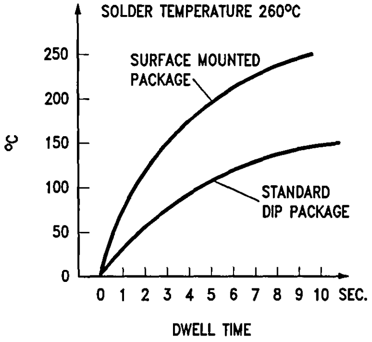
TL/XX/0026-16

Mixed Surface-Mount and Axial-Leaded Insertion Components Production Flow



TL/XX/0026-17

Thermal stress of the packages during surface-mounting processing is more severe than during standard DIP PC board mounting processes. *Figure B* illustrates package temperature versus wave soldering dwell time for surface mounted packages (components are immersed into the molten solder) and the standard DIP wave soldering process. (Only leads of the package are immersed into the molten solder).

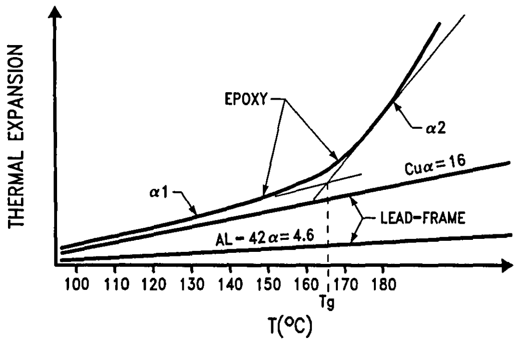


TL/XX/0026-18

FIGURE B

For an ideal package, the thermal expansion rate of the encapsulant should match that of the leadframe material in order for the package to maintain mechanical integrity during the soldering process. Unfortunately, a perfect matchup of thermal expansion rates with most presently used packaging materials is scarce. The problem lies primarily with the epoxy compound.

Normally, thermal expansion rates for epoxy encapsulant and metal lead frame materials are linear and remain fairly close at temperatures approaching 160°C, *Figure C.* At lower temperatures the difference in expansion rate of the two materials is not great enough to cause interface separation. However, when the package reaches the glass-transition temperature (T_g) of epoxy (typically 160–165°C), the thermal expansion rate of the encapsulant increases sharply, and the material undergoes a transition into a plastic state. The epoxy begins to expand at a rate three times or more greater than the metal leadframe, causing a separation at the interface.



TL/XX/0026-19

FIGURE C

When this happens during a conventional wave soldering process using flux and acid cleaners, process residues and even solder can enter the cavity created by the separation and become entrapped when the material cools. These contaminants can eventually diffuse into the interior of the package, especially in the presence of moisture. The result is die contamination, excessive leakage, and even catastrophic failure. Unfortunately, electrical tests performed immediately following soldering may not detect potential flaws.

Most soldering processes involve temperatures ranging up to 260°C, which far exceeds the glass-transition temperature of epoxy. Clearly, circuit boards containing SMD packages require tighter process controls than those used for boards populated solely by DIPs.

Figure D is a summary of accelerated bias moisture test performance on the 30V bipolar process.

Group 1 — Standard DIP package

Group 2 — SO packages vapor-phase reflow soldered on PC boards

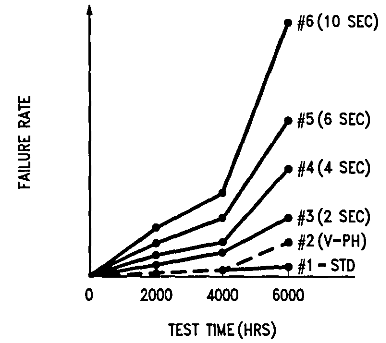
Group 3-6 SO packages wave soldered on PC boards

Group 3 — dwell time 2 seconds

4 — dwell time 4 seconds

5 — dwell time 6 seconds

6 — dwell time 10 seconds



TL/XX/0026-20

FIGURE D

It is clear based on the data presented that SO packages soldered onto PC boards with the vapor phase reflow process have the best long term bias moisture performance and this is comparable to the performance of standard DIP packages. The key advantage of reflow soldering methods is the clean environment that minimized the potential for contamination of surface mounted packages, and is preferred for the surface-mount process.

When wave soldering is used to surface mount components on the board, the dwell time of the component under molten solder should be no more than 4 seconds, preferrably under 2 seconds in order to prevent damage to the component. Non-Halide, or (organic acid) fluxes are highly recommended.

PICK AND PLACE

The choice of automatic (all generally programmable) pick-and-place machines to handle surface mounting has grown considerably, and their selection is based on individual needs and degree of sophistication.

The basic component-placement systems available are classified as:

(a) In-line placement

- Fixed placement stations
- Boards indexed under head and respective components placed

(b) Sequential placement

- Either a X-Y moving table system or a θ, X-Y moving pickup system used
- Individual components picked and placed onto boards

(c) Simultaneous placement

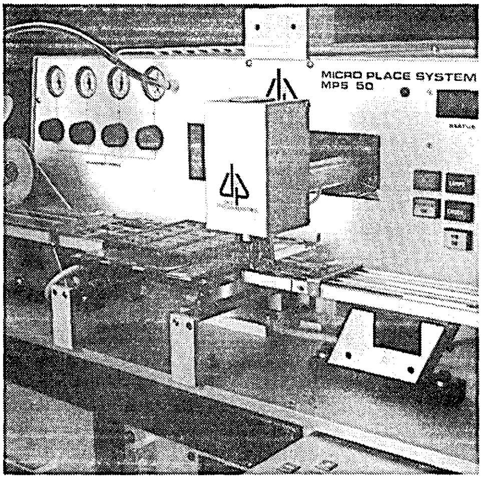
- Multiple pickup heads
- Whole array of components placed onto the PCB at the same time

(d) Sequential/simultaneous placement

- X–Y moving table, multiple pickup heads system
- Components placed on PCB by successive or simultaneous actuation of pickup heads

The SO package is treated almost the same as surface-mount, passive components requiring correct orientation in placement on the board.

Pick and Place Action



TL/XX/0026-21

BAKE

This is recommended, despite claims made by some solder paste suppliers that this step be omitted.

The functions of this step are:

- Holds down the solder globules during subsequent reflow soldering process and prevents expulsion of small solder balls.
- Acts as an adhesive to hold the components in place during handling between placement to reflow soldering.
- Holds components in position when a double-sided surface-mounted board is held upside down going into a vapor-phase reflow soldering operation.
- Removes solvents which might otherwise contaminate other equipment.
- Initiates activator cleaning of surfaces to be soldered.
- Prevents moisture absorption.

The process is moreover very simple. The usual schedule is about 20 minutes in a 65°C–95°C (dependent on solvent system of solder paste) oven with adequate venting. Longer bake time is not recommended due to the following reasons:

- The flux will degrade and affect the characteristics of the paste.
- Solder globules will begin to oxidize and cause solderability problems.
- The paste will creep and after reflow, may leave behind residues between traces which are difficult to remove and vulnerable to electro-migration problems.

REFLOW SOLDERING

There are various methods for reflowing the solder paste, namely:

- Hot air reflow
- Infrared heating (furnaces)
- Convectional oven heating
- Vapor-phase reflow soldering
- Laser soldering

For SO applications, hot air reflow/infrared furnace may be used for low-volume production or prototype work, but vapor-phase soldering reflow is more efficient for consistency and speed. Oven heating is not recommended because of "hot spots" in the oven and uneven melting may result. Laser soldering is more for specialized applications and requires a great amount of investment.

HOT GAS REFLOW/INFRARED HEATING

A hand-held or table-mount air blower (with appropriate orifice mask) can be used.

The boards are preheated to about 100°C and then subjected to an air jet at about 260°C. This is a slow process and results may be inconsistent due to various heat-sink properties of passive components.

Use of an infrared furnace is the next step to automating the concept, except that the heating is promoted by use of IR lamps or panels. The main objection to this method is that certain materials may heat up at different rates under IR radiation and may result in damage to these components (usually sockets and connectors). This could be minimized by using far-infrared (non-focused) system.

VAPOR-PHASE REFLOW SOLDERING

Currently the most popular and consistent method, vapor-phase soldering utilizes a fluoroinert fluid with excellent heat-transfer properties to heat up components until the solder paste reflows. The maximum temperature is limited by the vapor temperature of the fluid.

The commonly used fluids (supplied by 3M Corp) are:

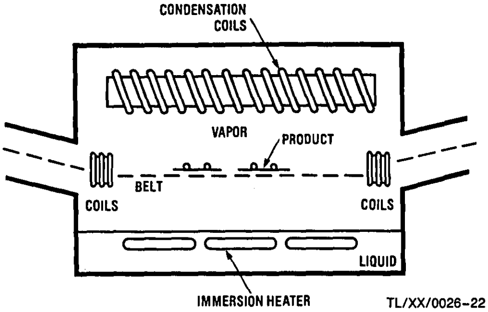
- FC-70, 215°C vapor (most applications) or FX-38
- FC-71, 253°C vapor (low-lead or tin-plate)

HTC, Concord, CA, manufactures equipment that utilizes this technique, with two options:

- Batch systems, where boards are lowered in a basket and subjected to the vapor from a tank of boiling fluid.
- In-line conveyorized systems, where boards are placed onto a continuous belt which transports them into a concealed tank where they are subjected to an environment of hot vapor.

Dwell time in the vapor is generally on the order of 15–30 seconds (depending on the mass of the boards and the loading density of boards on the belt).

In-Line Conveyorized Vapor-Phase Soldering



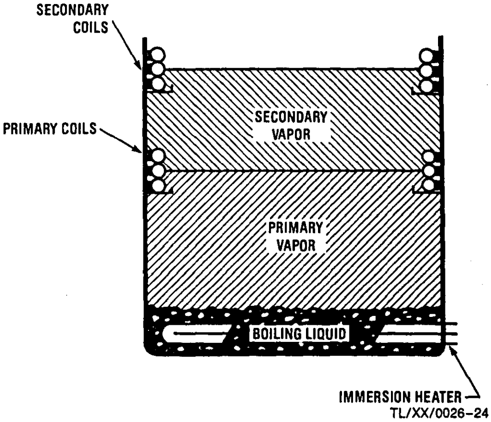
TL/XX/0026–22

The question of thermal shock is asked frequently because of the relatively sharp increase in component temperature from room temperature to 215°C. SO packages mounted on representative boards have been tested and have shown little effect on the integrity of the packages. Various packages, such as cerdips, metal cans and TO-5 cans with glass seals, have also been tested.

Vapor-Phase Furnace

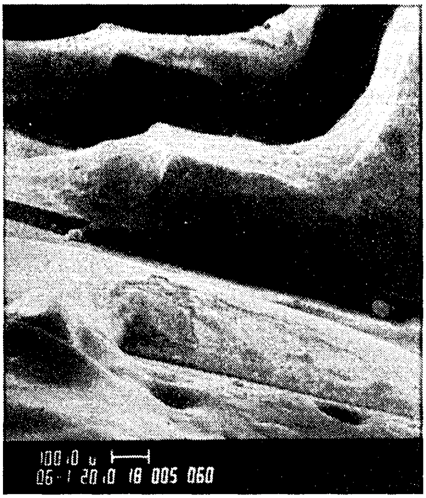
TL/XX/0026–23

Batch-Fed Production Vapor-Phase Soldering Unit



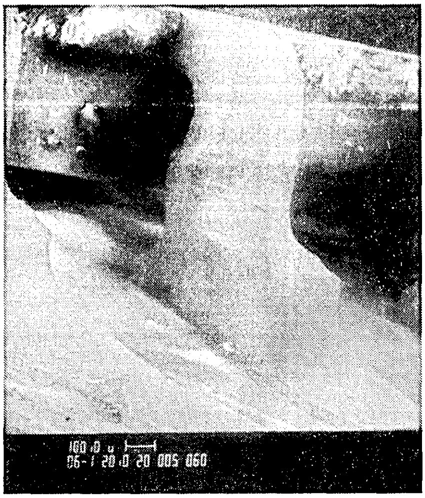
TL/XX/0026–24

Solder Joints on a SO-14 Package on PCB



TL/XX/0026-25

Solder Joints on a SO-14 Package on PCB



TL/XX/0026-26

PRINTED CIRCUIT BOARD

The SO package is molded out of clean, thermoset plastic compound and has no particular compatibility problems with most printed circuit board substrates.

The package can be reliably mounted onto substrates such as:

- G10 or FR4 glass/resin
- FR5 glass/resin systems for high-temperature applications
- Polymide boards, also high-temperature applications
- Ceramic substrates

General requirements for printed circuit boards are:

- Mounting pads should be solder-plated whenever applicable.
- Solder masks are commonly used to prevent solder bridging of fine lines during soldering.

  The mask also protects circuits from processing chemical contamination and corrosion.

  If coated over pre-tinned traces, residues may accumulate at the mask/trace interface during subsequent reflow, leading to possible reliability failures.

  Recommended application of solder resist on bare, clean traces prior to coating exposed areas with solder.

  General requirements for solder mask:

  — Good pattern resolution.

  — Complete coverage of circuit lines and resistance to flaking during soldering.

  — Adhesion should be excellent on substrate material to keep off moisture and chemicals.

  — Compatible with soldering and cleaning requirements.

SOLDER PASTE SCREEN PRINTING

With the initial choice of printed circuit lithographic design and substrate material, the first step in surface mounting is the application of solder paste.

The typical lithographic "footprints" for SO packages are illustrated below. Note that the 0.050″ lead center-center spacing is not easily managed by commercially-available air pressure, hand-held dispensers.

Using a stainless-steel, wire-mesh screen stencilled with an emulsion image of the substrate pads is by far the most common and well-tried method. The paste is forced through the screen by a V-shaped plastic squeegee in a sweeping manner onto the board placed beneath the screen.

The setup for SO packages has no special requirement from that required by other surface-mounted, passive components. Recommended working specifications are:

- Use stainless-steel, wire-mesh screens, #80 or #120, wire diameter 2.6 mils. Rule of thumb: mesh opening should be approximately 2.5–5 times larger than the average particle size of paste material.
- Use squeegee of Durometer 70.
- Experimentation with squeegee travel speed is recommended, if available on machine used.
- Use solder paste of mesh 200–325.
- Emulsion thickness of 0.005″ usually used to achieve a solder paste thickness (wet) of about 0.008″ typical.
- Mesh pattern should be 90 degrees, square grid.
- Snap-off height of screen should not exceed ⅛″, to avoid damage to screens and minimize distortion.

SOLDER PASTE

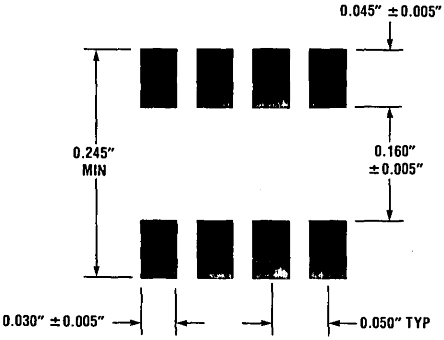
Selection of solder paste tends to be confusing, due to numerous formulations available from various manufacturers. In general, the following guidelines are sufficient to qualify a particular paste for production:

- Particle sizes (see photographs below). Mesh 325 (approximately 45 microns) should be used for general purposes, while larger (solder globules) particles are preferred for leadless components (LCC). The larger particles can easily be used for SO packages.

- Uniform particle distribution. Solder globules should be spherical in shape with uniform diameters and minimum amount of elongation (visual under 100/200 × magnification). Uneven distribution causes uneven melting and subsequent expulsion of smaller solder balls away from their proper sites.
- Composition, generally 60/40 or 63/37 Sn/Pb. Use 62/36 Sn/Pb with 2% Ag in the presence of Au on the soldering area. This formulation reduces problems of metal leaching from soldering pads.
- RMA flux system usually used.
- Use paste with aproximately 88–90% solids.

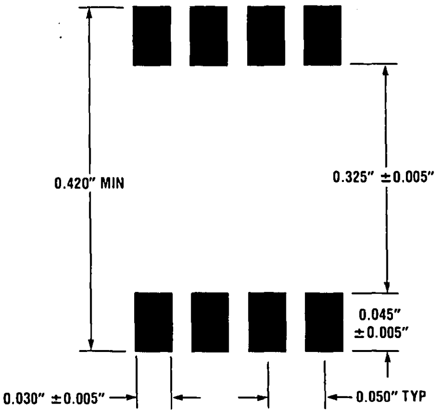
RECOMMENDED SOLDER PADS FOR SO PACKAGES

SO-8, SO-14, SO-16



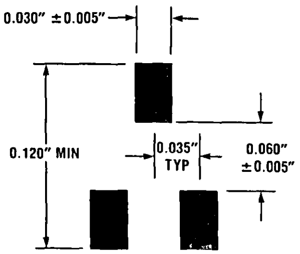
TL/XX/0026-27

SO-16L, SO-20



TL/XX/0026-28

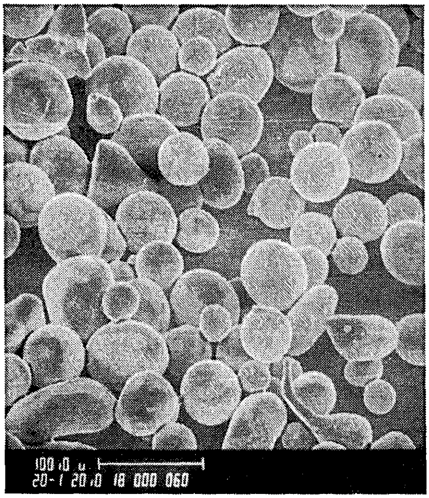
SOT-23



TL/XX/0026-29

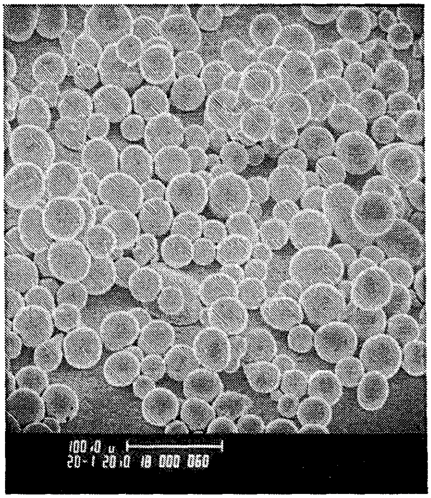
Comparison of Particle Size/Shape of Various Solder Pastes

200 × Alpha (62/36/2)



TL/XX/0026-30

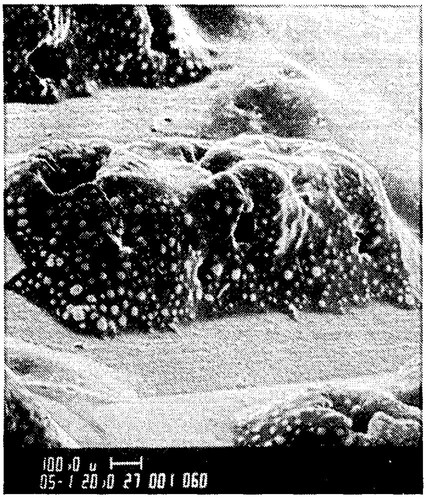
200 × Kester (63/37)



TL/XX/0026-31

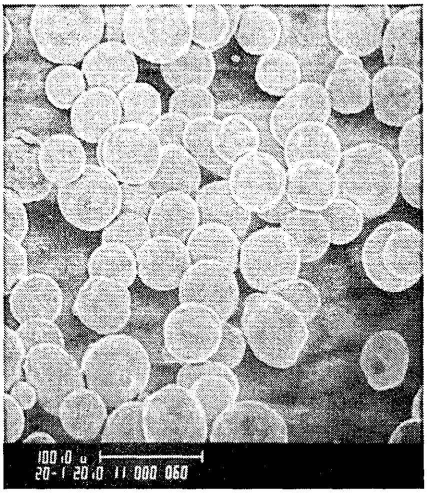
Comparison of Particle Size/Shape of Various Solder Pastes (Continued)

Solder Paste Screen on Pads



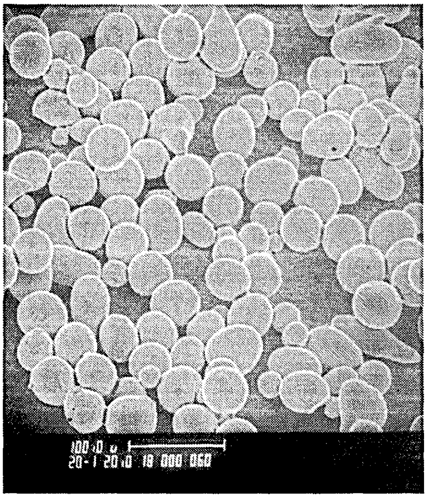
TL/XX/0026-32

200 × Fry Metal (63/37)



TL/XX/0026-33

200 ESL (63/37)



TL/XX/0026-34

CLEANING

The most critical process in surface mounting SO packages is in the cleaning cycle. The package is mounted very close to the surface of the substrate and has a tendency to collect residue left behind after reflow soldering.

Important considerations in cleaning are:

- Time between soldering and cleaning to be as short as possible. Residue should not be allowed to solidify on the substrate for long periods of time, making it difficult to dislodge.
- A low surface tension solvent (high penetration) should be employed. Solvents commercially available are:

  Freon TMS (general purpose)
  Freon TE35/TP35 (cold-dip cleaning)
  Freon TES (general purpose)

  It should also be noted that these solvents generally will leave the substrate surface hydrophobic (moisture repellent), which is desirable.

  Prelete or 1,1,1-Trichloroethane
  Kester 5120/5121
- A defluxer system which allows the workpiece to be subjected to a solvent vapor, followed by a rinse in pure solvent and a high-pressure spray lance are the basic requirments for low-volume production.
- For volume production, a conveyorized, multiple hot solvent spray/jet system is recommended.
- Rosin, being a natural occurring material, is not readily soluble in solvents, and has long been a stumbling block to the cleaning process. In recent developments, synthetic flux (SA flux), which is readily soluble in Freon TMS solvent, has been developed. This should be explored where permissible.

The dangers of an inadequate cleaning cycle are:

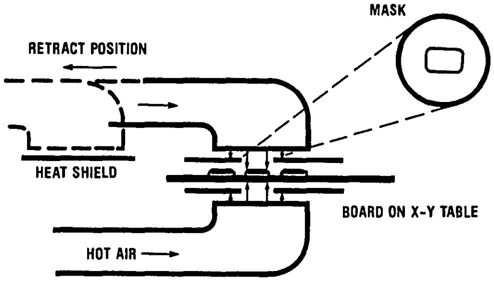
- Ion contamination, where ionic residue left on boards would cause corrosion to metallic components, affecting the performance of the board.
- Electro-migration, where ionic residue and moisture present on electrically-biased boards would cause dentritic growth between close spacing traces on the substrate, resulting in failures (shorts).

REWORK

Should there be a need to replace a component or re-align a previously disturbed component, a hot air system with appropriate orifice masking to protect surrounding components may be used.

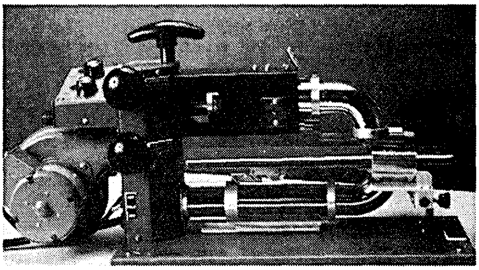
When rework is necessary in the field, specially-designed tweezers that thermally heat the component may be used to remove it from its site. The replacement can be fluxed at the lead tips or, if necessary, solder paste can be dispensed onto the pads using a varimeter. After being placed into position, the solder is reflowed by a hot-air jet or even a standard soldering iron.

Hot-Air Solder Rework Station



TL/XX/0026-35

Hot-Air Rework Machine



TL/XX/0026-36

WAVE SOLDERING

In a case where lead insertions are made on the same board as surface-mounted components, there is a need to include a wave-soldering operation in the process flow.

Two options are used:

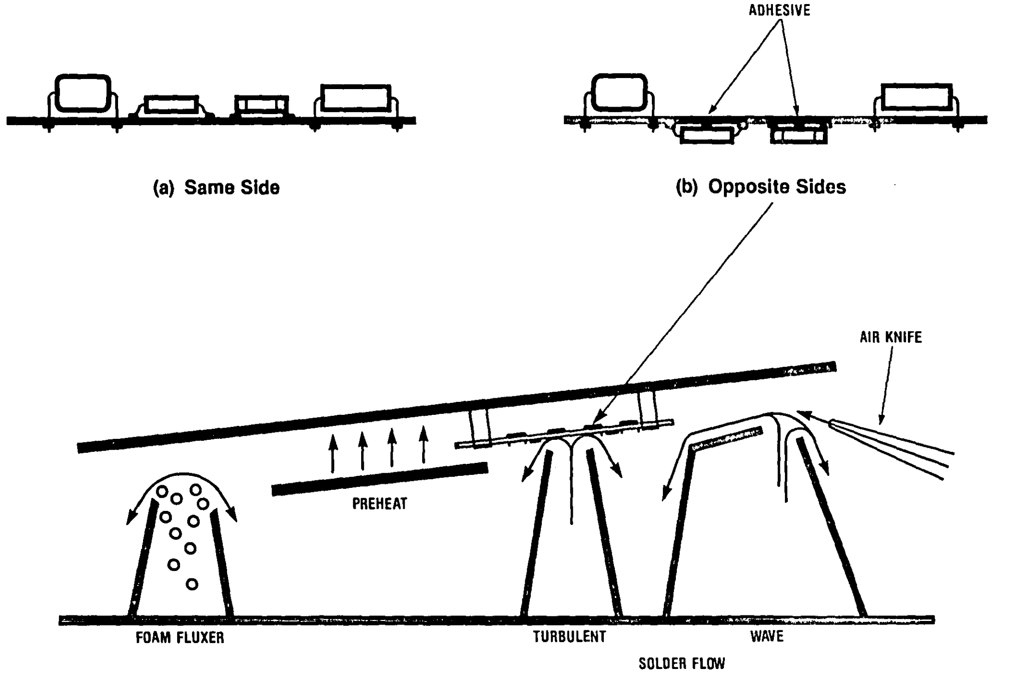
- Surface mounted components are placed and vapor phase reflowed before auto-insertion of remaining components. The board is carried over a standard wave-solder system and the underside of the board (only lead-inserted leads) soldered.
- Surface-mounted components are placed in position, but no solder paste is used. Instead, a drop of adhesive about 5 mils maximum in height with diameter not exceeding 25% width of the package is used to hold down the package. The adhesive is cured and then proceeded to auto-insertion on the reverse side of the board (surface-mounted side facing down). The assembly is then passed over a "dual wave" soldering system. Note that the surface-mounted components are immersed into the molten solder.

Lead trimming will pose a problem after soldering in the latter case, unless the leads of the insertion components are pre-trimmed or the board specially designed to localize certain areas for easy access to the trim blade.

The controls required for wave soldering are:

- Solder temperature to be 240–260°C. The dwell time of components under molten solder to be short (preferably kept under 2 seconds), to prevent damage to most components and semiconductor devices.
- RMA (Rosin Mildly Activated) flux or more aggressive OA (Organic Acid) flux are applied by either dipping or foam fluxing on boards prior to preheat and soldering. Cleaning procedures are also more difficult (aqueous, when OA flux is used), as the entire board has been treated by flux (unlike solder paste, which is more or less localized). Non-halide OA fluxes are highly recommended.
- Preheating of boards is essential to reduce thermal shock on components. Board should reach a temperature of about 100°C just before entering the solder wave.
- Due to the closer lead spacings (0.050″ vs 0.100″ for dual-in-line packages), bridging of traces by solder could occur. The reduced clearance between packages also causes "shadowing" of some areas, resulting in poor solder coverage. This is minimized by dual-wave solder systems.

Mixed Surface Mount and Lead Insertion



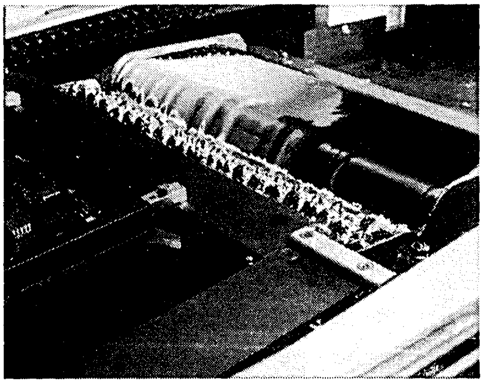
TL/XX/0026-37

A typical dual-wave system is illustrated below, showing the various stages employed. The first wave typically is in turbulence and given a transverse motion (across the motion of the board). This covers areas where "shadowing" occurs. A second wave (usually a broad wave) then proceeds to perform the standard soldering. The departing edge from the solder is such to reduce "icicles," and is still further reduced by an air knife placed close to the final soldering step. This air knife will blow off excess solder (still in the fluid stage) which would otherwise cause shorts (bridging) and solder bumps.

AQUEOUS CLEANING

- For volume production, a conveyorized system is often used with a heated recirculating spray wash (water temperature 130°C), a final spray rinse (water temperature 45–55°C), and a hot (120°C) air/air-knife drying section.
- For low-volume production, the above cleaning can be done manually, using several water rinses/tanks. Fast-drying solvents, like alcohols that are miscible with water, are sometimes used to help the drying process.
- Neutralizing agents which will react with the corrosive materials in the flux and produce material readily soluble in water may be used; the choice depends on the type of flux used.
- Final rinse water should be free from chemicals which are introduced to maintain the biological purity of the water. These materials, mostly chlorides, are detrimental to the assemblies cleaned because they introduce a fresh amount of ionizable material.

Dual Wave



TL/XX/0026-38

CONFORMAL COATING

Conformal coating is recommended for high-reliability PCBs to provide insulation resistance, as well as protection against contamination and degradation by moisture.

Requirements:

- Complete coating over components and solder joints.
- Thixotropic material which will not flow under the packages or fill voids, otherwise will introduce stress on solder joints on expansion.
- Compatibility and possess excellent adhesion with PCB material/components.
- Silicones are recommended where permissible in application.

SMD Lab Support

FUNCTIONS

Demonstration—Introduce first-time users to surface-mounting processes.

Service—Investigate problems experienced by users on surface mounting.

Reliability Builds—Assemble surface-mounted units for reliability data acquisition.

Techniques—Develop techniques for handling different materials and processes in surface mounting.

Equipment—In conjunction with equipment manufacturers, develop customized equipments to handle high density, new technology packages developed by National.

In-House Expertise—Availability of in-house expertise on semiconductor research/development to assist users on packaging queries.

National Semiconductor

Plastic Leaded Chip Carrier (PLCC) Packaging

General Description

The Plastic Leaded Chip Carrier (PLCC) is a miniaturized low cost semiconductor package designed to replace the Plastic Dual-In-Line Package (P-DIP) in high density applications. The PLCC utilizes a smaller lead-to-lead spacing—0.050″ versus 0.100″ - and leads on all four sides to achieve a significant footprint reduction over the P-DIP. The rolled under J-bend leadform separates this package style from other plastic quad packages with flat or gull wing lead forms. As with virtually all packages of 0.050″ or less lead spacing, the PLCC requires surface mounting to printed circuit boards as opposed to the more conventional thru-hole mounting of the P-DIP.

History

The Plastic Leaded Chip Carrier with J-bend leadform was first introduced in 1976 as a premolded plastic package. The premolded version has yet to become popular but the quad format with J-Bend leads has been adapted to traditional post molded packaging technology (the same technology used to manufacture the P-DIP). In 1980 National Semiconductor developed a post molded version of the PLCC. The J-bend leadform allowed them to adopt the footprint connection pattern already registered with JEDEC for the leadless chip carrier (LCC). In 1981 a task force was organized within JEDEC to develop a PLCC registration for package I/O counts of 20, 28, 44, 52, 68, 84, 100, and 124. A registered outline was completed in 1984 (JEDEC Outline MO-047) after many changes and improvements over the original proposals. This first PLCC registration covers square packages with an equal number of leads on all sides. A second registration, MO-052, was completed in 1985 for rectangular packages with I/O counts of 18, 22, 28 and 32.

Since 1980 many additional semiconductor manufacturers and packaging subcontractors have developed PLCC capability. There are now well over 20 sources with the number growing steadily.

Surface Mounting

Surface mounting refers to component attachment whereby the component leads or pads rest on the surface of the PCB instead of the traditional approach of inserting the leads into through-holes which go through the board. With surface mounting there are solder pads on the PCB which align with the leads or pads on the component. The resulting solder joint forms both the mechanical and electrical connection.

ADVANTAGES

The primary reason for surface mounting is to allow leads to be placed closer together than the 0.100″ standard for DIPs with through-hole mounting. Through-hole mounting on smaller than 0.100″ spacing is difficult to achieve in production and generally avoided. The move to 0.050″ lead spacing offered with the current generation of surface mounted components, along with a switch from a dual-in-line format to a quad format, has achieved a threefold increase in component mounting density. A need to achieve greater density is a major driving force in today's marketplace.

MANUFACTURING TECHNIQUES

Learning how to surface mount components to printed circuit boards requires the user to become educated in new assembly processes not typically associated with through-hole insertion/wave soldering assembly methods.

Surface mounting involves three basic process steps:

1) Application of solder or solder paste to the printed circuit board.

2) Positioning of the component onto the printed circuit board

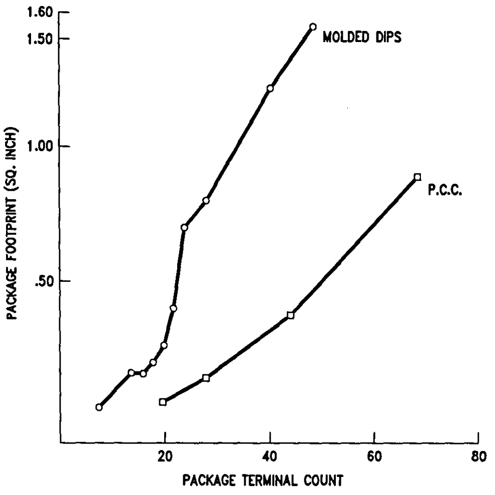
3) Reflowing of the solder or solder paste.

As with any process, there are many details involved to achieve acceptable throughput and acceptable quality. National Semiconductor offers a surface mounting guide which deals with the specifics of successful surface mounting. We encourage the user to review this document and to contact us if further information on surface mounting is desired.

Benefits of the PLCC

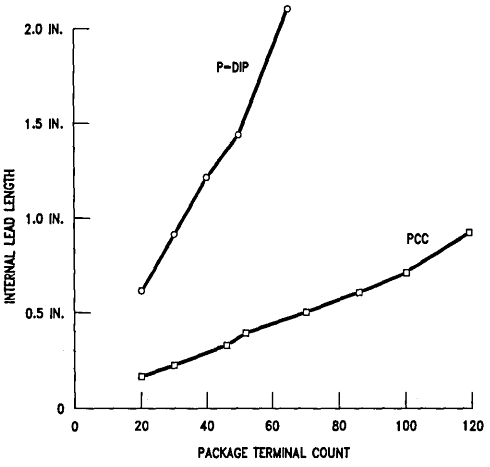
There are four principle advantages offered the user by switching from P-DIP to PLCC. These four advantages are outlined below as follows:

1. Increased Density—
   - — Typically 3-to-1 size reduction of printed circuit boards. See *Figure 1* for a footprint comparison between PLCC and P-DIP. This can be as high as 6-to-1 in certain applications.
   - — Surface mounting allows components to be placed on both sides of the board.
   - — Surface mount and thru-hole mount components can be placed on the same board.
   - — The large diameter thru-holes can be reduced in number, entirely eliminated, or reduced in size (if needed for via connection).
2. Increased Performance—
   - — Shorter traces on printed circuit boards.
   - — Better high frequency operation.
   - — Shorter leads in package. *Figure 2* and Table I compare PLCC and P-DIP mechanical and electrical characteristics.
3. Increased Reliability—
   - — Leads are well protected.
   - — Fewer connectors.
   - — Simplified rework.
   - — Vibration and shock resistant.
4. Reduced Cost—
   - — Fewer or smaller printed circuit boards.
   - — Less hardware.
   - — Same low cost printed circuit board material.
   - — Plastic packaging material.
   - — Reduced number of costly plated-through-holes.
   - — Fewer circuit layers.



TL/ZZ/0001-1

FIGURE 1. Footprint Area of PLCC vs. P-DIP

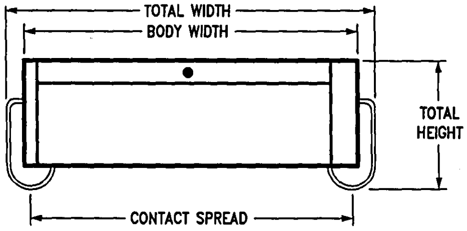


TL/ZZ/0001-2

FIGURE 2. Longest Internal Lead PLCC vs. P-DIP

TABLE I. Electrical Performance of PLCC vs. P-DIP (44 I/O PLCC vs. 40 I/O P-DIP, both with Copper Leads)

Criteria	Shortest Lead		Longest Lead	
	PLCC	P-DIP	PLCC	P-DIP
Lead Resistance (Measured)	3Ω	4Ω	6Ω	7Ω
Lead-to-Lead Capacitance (Measured on Adjacent Leads)	0.1 pF	0.1 pF	0.3 pF	3.0 pF
Lead Self-Inductance (Calculated)	3.2 nH	1.4 nH	3.5 nH	19.1 nH



TL/ZZ/0001-3

FIGURE 3. Package Outline

TABLE II. Principle Dimensions Inches/(Millimeters) (Refer to *Figure 3*)

Lead Count	Total Width		Total Height		Body Width		Contact Spread	
	Min	Max	Min	Max	Min	Max	Min	Max
20	0.385 sq. (9.779)	0.395 sq. (10.03)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.345 sq. (8.763)	0.355 sq. (9.017)	0.310 sq. (7.874)	0.330 sq. (8.382)
28	0.485 sq. (12.32)	0.495 sq. (12.57)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.445 sq. (11.30)	0.455 sq. (11.56)	0.410 sq. (10.41)	0.430 sq. (10.92)
44	0.685 sq. (17.40)	0.695 sq. (17.65)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.645 sq. (16.38)	0.655 sq. (16.64)	0.610 sq. (15.49)	0.630 sq. (16.00)

TABLE II. Principle Dimensions Inches/(Millimeters) (Refer to *Figure 3*) (Continued)

Lead Count	Total Width		Total Height		Body Width		Contact Spread	
	Min	Max	Min	Max	Min	Max	Min	Max
68	0.985 sq. (25.02)	0.995 sq. (25.27)	0.165 sq. (4.191)	0.180 sq. (4.572)	0.945 sq. (24.00)	0.955 sq. (24.26)	0.910 sq. (23.11)	0.930 sq. (23.62)
84	1.185 sq. (30.10)	1.195 sq. (30.36)	0.165 sq. (4.191)	0.180 sq. (4.572)	1.150 sq. (29.21)	1.158 sq. (29.41)	1.110 sq. (28.20)	1.130 sq. (28.70)
124	1.685 sq. (49.13)	1.695 sq. (49.39)	0.180 sq. (4.572)	0.200 sq. (5.080)	1.650 sq. (41.91)	1.658 sq. (42.11)	1.610 sq. (40.90)	1.630 sq. (41.40)

TABLE III. Package Thermal Resistance
(Deg. C/Watt, Junction-to-Ambient, Board Mount)

Lead Count	Device Size		
	1,000 Mil^2	10,000 Mil^2	100,000 Mil^2
20	102	85	67
28	95	73	55
44	54	47	40
68	44	40	38
84*	40	35	30
124*	40	35	30

*Estimated values

Package Design Criteria

Experience has taught us there are certain criteria to the PLCC design which must be followed to provide the user with the proper mechanical and thermal performance. These requirements should be carefully reviewed by the user when selecting suppliers for devices in PLCC. Some of these are covered by the JEDEC registration and some are not. These important requirements are listed in Table IV.

Reliability

National Semiconductor utilizes an assembly process for the PLCC which is similar to our P-DIP assembly process. We also utilize identical materials. This is a very important point when considering reliability. Many years of research and development have gone into steadily improving our P-DIP quality and maintaining a leadership position in plastic package reliability. All of this technology can be directly applied to the PLCC. Table V shows the results of applying this technology to the PLCC. As we make further advances in plastic package reliability, these will also be applied to the PLCC.

Sockets

There are several manufacturers currently offering sockets for the plastic chip carrier. Following is a listing of those manufacturers. The listing is divided into test/burn-in and production categories. There may be some individual sockets that will cover both requirements.

TABLE IV. Package Design Criteria

Criteria	Required to Comply with JEDEC Registration
Minimum Inside Bend Radius of Lead at Shoulder Equal or Greater than Lead Thickness—to Prevent Lead Cracking/Fatigue	Not Required
Minimum One Mil Clearance Between Lead and Plastic Body at all Points—to Provide Lead Compliancy and Prevent Shoulder Joint Cracking/Fatigue	Not Required
Copper Leads for Low Thermal Resistance	Not Required
Minimum 10 Mil Lead Thickness for Low Thermal Resistance and Good Handling Properties	Not Required
Minimum 26 Mil Lead Shoulder Width to Prevent Interlocking of Devices During Handling	Yes
Maximum 4 Mils coplanarity Across Seating Plane of all Leads	Yes

TABLE V. Reliability Test Data
(Expressed as Failures per Units Tested)

Device/Package	OPL	TMCL	TMSK	BHTL	ACLV
LM324/20 Lead	0/96	0/199	0/50	0/97	0/300
LF353/20 Lead	0/50	0/50	—	0/45	0/100
DS75451/20 Lead	0/47	—	0/50	0/93	0/179
DM875191/28 Lead	0/154	0/154	0/154	0/154	0/154
DM875181/28 Lead	0/77	0/77	0/77	0/77	0/77

OPL = Dynamic high temperature operating life at 125°C or 150°C, 1,000 hours.
TMCL = Temperature cycle, Air-to-Air, −40°C to +125°C or −65°C to +150°C, 2,000 cycles.
TMSK = Thermal shock, Liquid-to-Liquid, −65°C to +150°C, 100 cycles.
BHTL = Biased humidity temperature life, 85°C, 85% humidity, 1,000 hours.
ACLV = Autoclave, 15 psi, 121°C, 100% humidity, 1,000 hours.

Production Sockets

AMP
Harrisburg, PA
(715) 564-0100

Augat
Attleboro, MA
(617) 222-2202

Burndy
Norwalk, CT
(203) 838-4444

Methode
Rolling Meadows, IL
(312) 392-3500

Textool
Irving, TX
(214) 259-2676

Thomas & Betts
Raritan, NJ
(201) 469-4000

Test/Burn-In Sockets

Plastronics
Irving, TX
(214) 258-1906

Textool
Irving, TX
(214) 259-2676

Yamaichi
c/o Nepenthe Dist.
(415) 856-9332

ADDITIONAL INFORMATION AND SERVICES

National Semiconductor offers additional Databooks which cover surface mount technology in much greater detail. We also have a surface mount laboratory to provide demonstrations and customer support, as well as technology development. Feel free to contact us about these additional resources.

National Semiconductor

TapePak®

The latest generation in VLSI packaging, TapePak is the package of the future—low-cost, reliable, high-leadcount packaging that's easy to handle, easy to test, and easy to mount. It's also compatible with existing surface-mount technology.

TapePak uses tape-automated bonding technology and a unique, patented outer ring to protect the leads and, at the same time, provide an effective test interface.

This outer ring is molded at the same time as the body of the package and creates test points outside the package leads. The test ring is discarded along with the tape as the package is excised by the automatic pick-and-place machine at the point of assembly.

During testing, the leads themselves never come in contact with the test socket, so lead damage and coplanarity problems are eliminated. The test ring also allows burn-in to be performed on each device.

Not only does this ring protect the leads during handling, testing and assembly, but it also allows leads to be placed on centers of 0.012 inch–0.020 inch (0.3 mm–0.5 mm), while the test points are placed on standard centers of 0.020 inch (0.5 mm), 0.025 inch (0.65 mm) or 0.050 inch (1.27 mm). That way, the test points are compatible with existing automatic test equipment.

TapePak packages are significantly smaller than conventional and alternative surface-mount packages. TapePak lead counts range from 40 to greater than 360, yet the largest package measures only 1.1 inches (28 mm) square.

Comparison of TapePak and Conventional Packages

	40L DIP		44L PLCC		40L TapePak	
Lead thickness (mils)	10.0		10.0		2.8	
Lead pitch (mils)	100		50		20	
Package length (mils)	2050		650		350	
Package width (mils)	600		650		350	
Package thickness (mils)*	175		180		71	
Volume ratio	24.4		9.1		1	
	40L DIP		44L PLCC		40L TapePak	
	Long	Short	Long	Short	Long	Short
Lead length (in)	1.0	0.3	0.35	0.25	0.1	0.1
Resistance (mOhm)	7	4	4	3	2.4	2.4
Inductance (nH)	22	6.0	6.5	5	1.2	1.2
Capacitance (pF) (lead to lead)	0.5	0.2	0.3	0.2	0.2	0.1

*Measured from seating plane to the top of the package.

A TapePak device can be less than 1/10 the size of a traditional DIP and 1/3 the size of other surface-mount packages such as a PLCC.

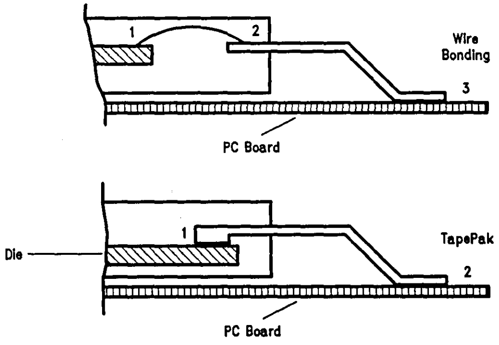
TapePak technology was designed to take full advantage of automatic assembly systems with their high speed and precision. It can be used with existing precision surface-mount assembly equipment with minimal modification. The only requirement is an accessory for removing the test ring and forming the leads at the point of assembly.

TapePak also provides a significant improvement in the electrical characteristics of each package. Lead capacitance and inductance, for example, can be reduced up to ten times that of other packages. Signal propagation time is also reduced, and thermal characteristics are improved. Because of the tremendous space savings, TapePak technology offers much greater power density per unit area as compared to DIP or alternative surface-mount packages.

Performance and reliability are improved because there are one-third fewer connections between the die and the PC board. Low-stress molding compounds also improve package reliability. TapePak devices pass stringent environmental tests, including autoclaving at 121°C at 15 psi and thermal shock from −65°C to +150°C for 1000 cycles.

No other package takes similar advantage of materials technology to provide the combination of low cost, high density, testability, damage resistance, and reliability.

TapePak has been accepted as an industry standard by the Joint Electronic Device and Engineering Council (JEDEC) and registration is in progress with the Electronic Industries Association of Japan (EIAJ). TapePak technology has also been licensed to other manufacturers for their own proprietary devices.



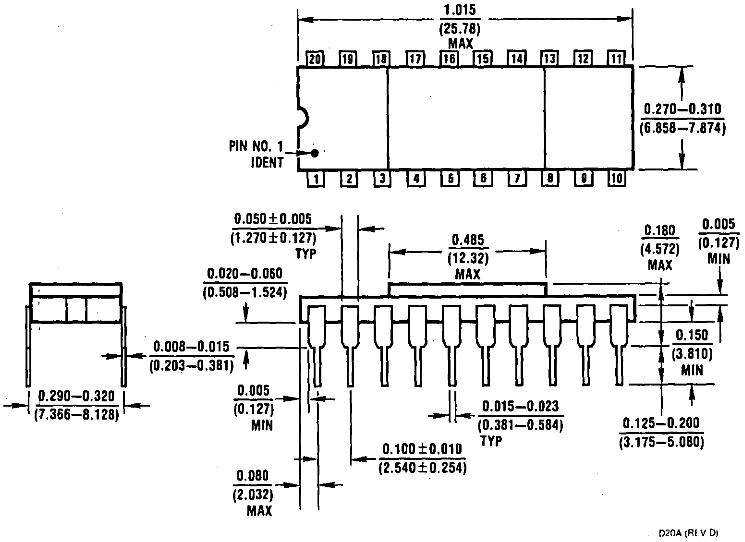
TL/XX/0077-1

With TapePak, there are one-third fewer connections between die and board than with traditional wire bonding.

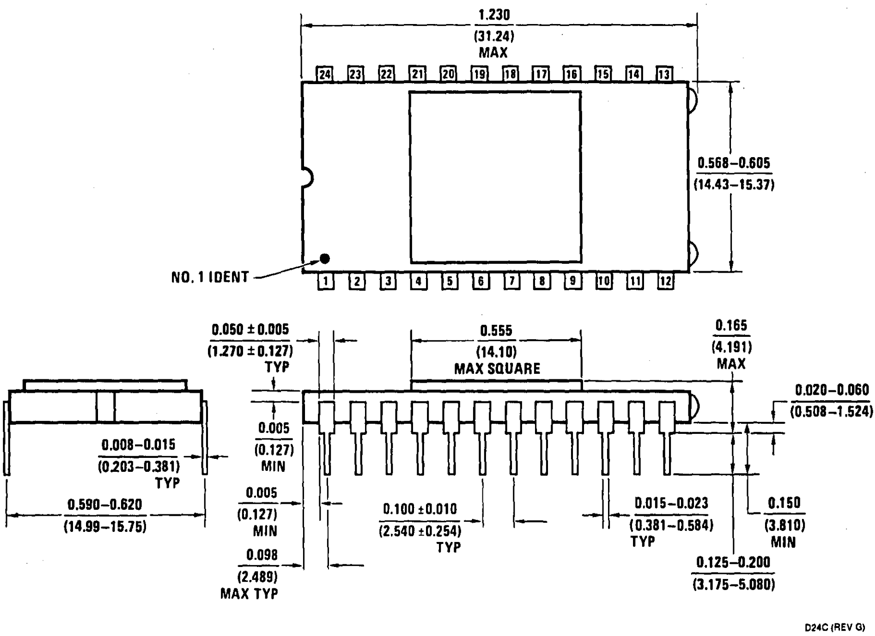
National Semiconductor

All dimensions are in inches (millimeters)

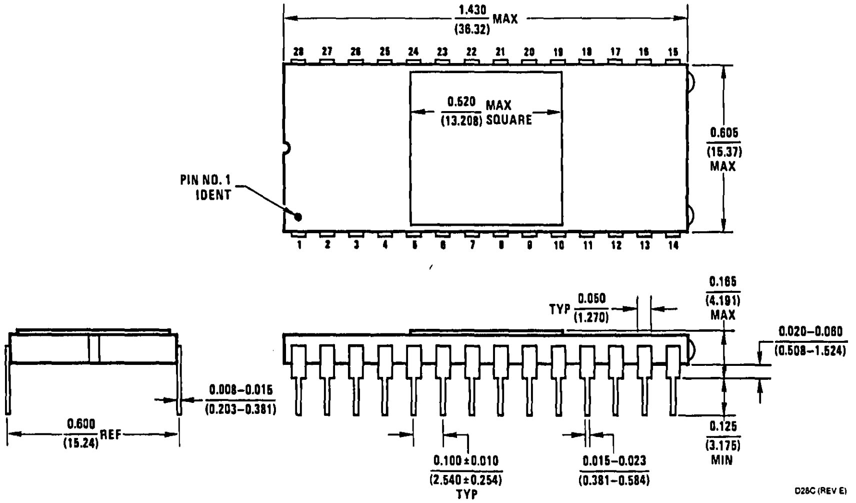
20 Lead Hermetic Dual-In-Line Package (D)
NS Package Number D20A



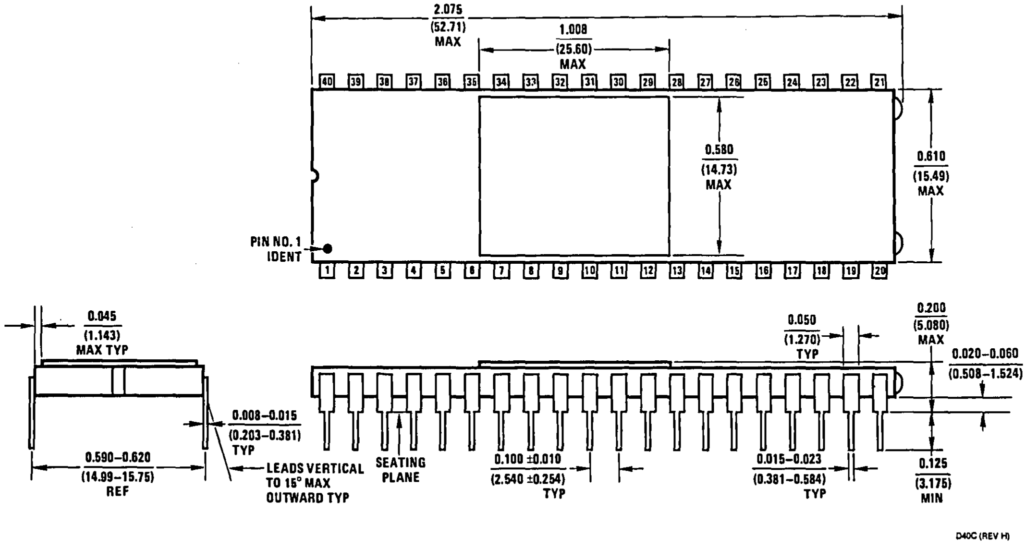
24 Lead Hermetic Dual-In-Line Package (D)
NS Package Number D24C



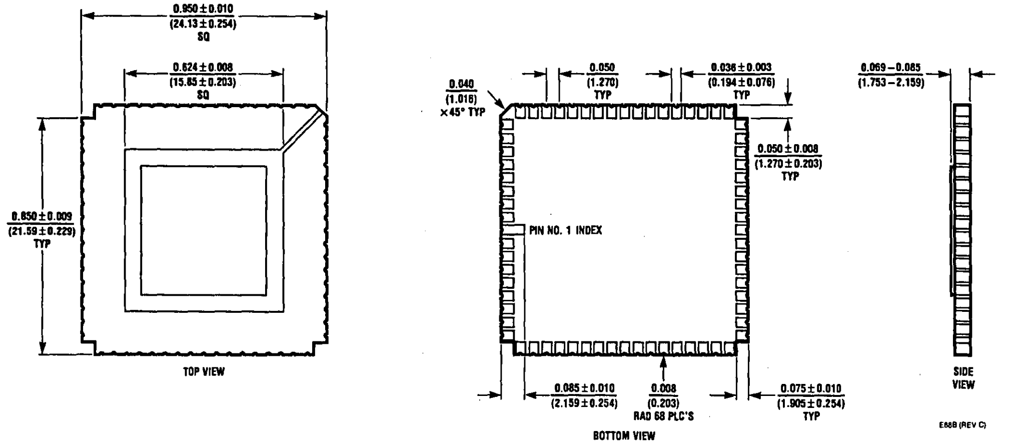
28 Lead Hermetic Dual-In-Line Package (D)
NS Package Number D28C



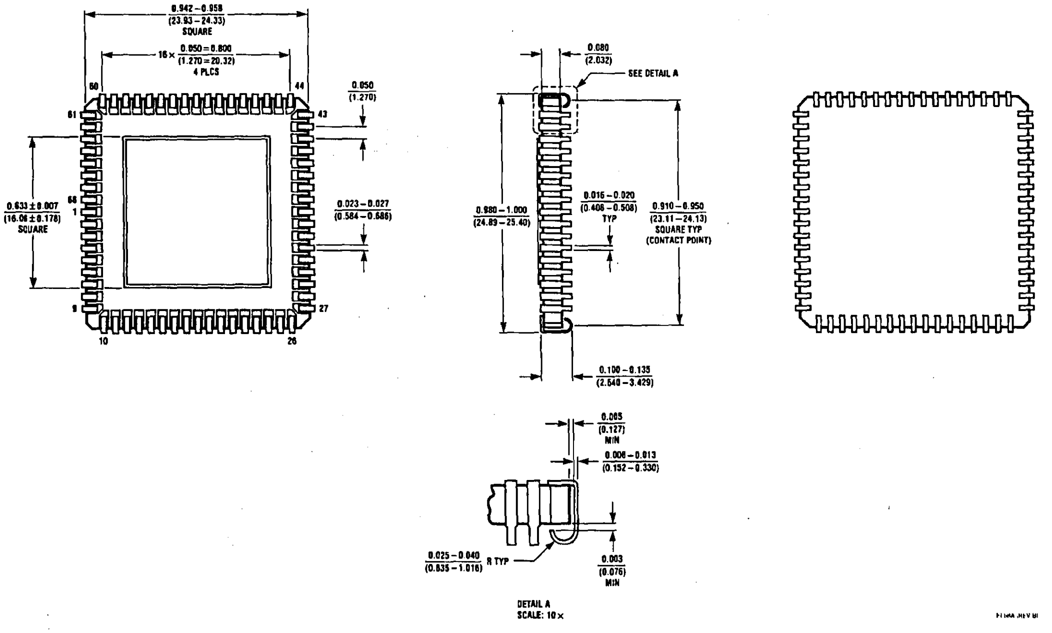
40 Lead Hermetic Dual-In-Line Package (D)
NS Package Number D40C



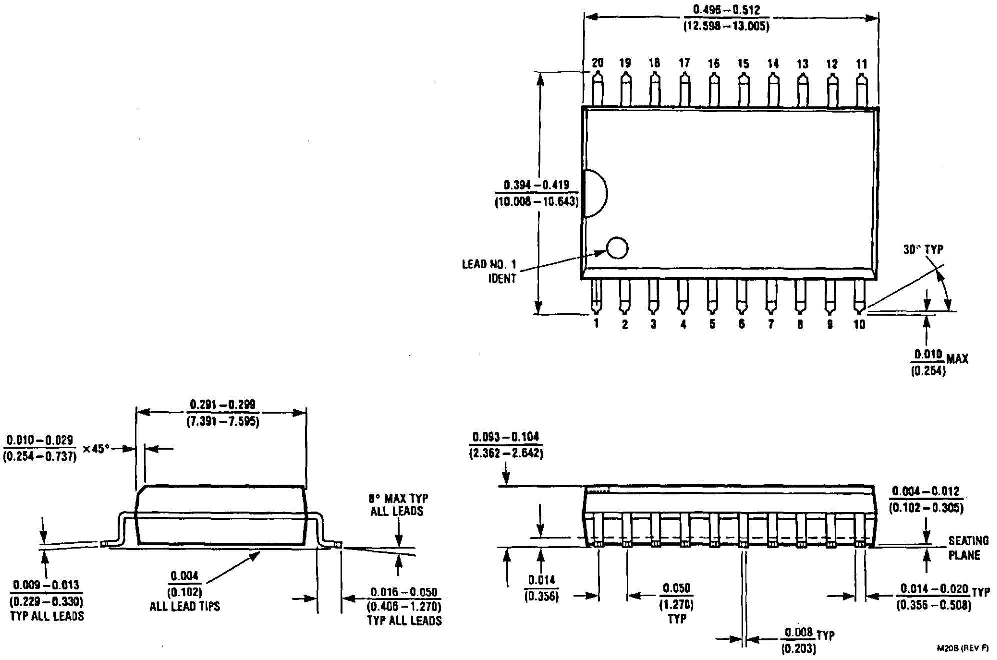
68 Pin Chip Carrier, Type B (E)
NS Package Number E68B



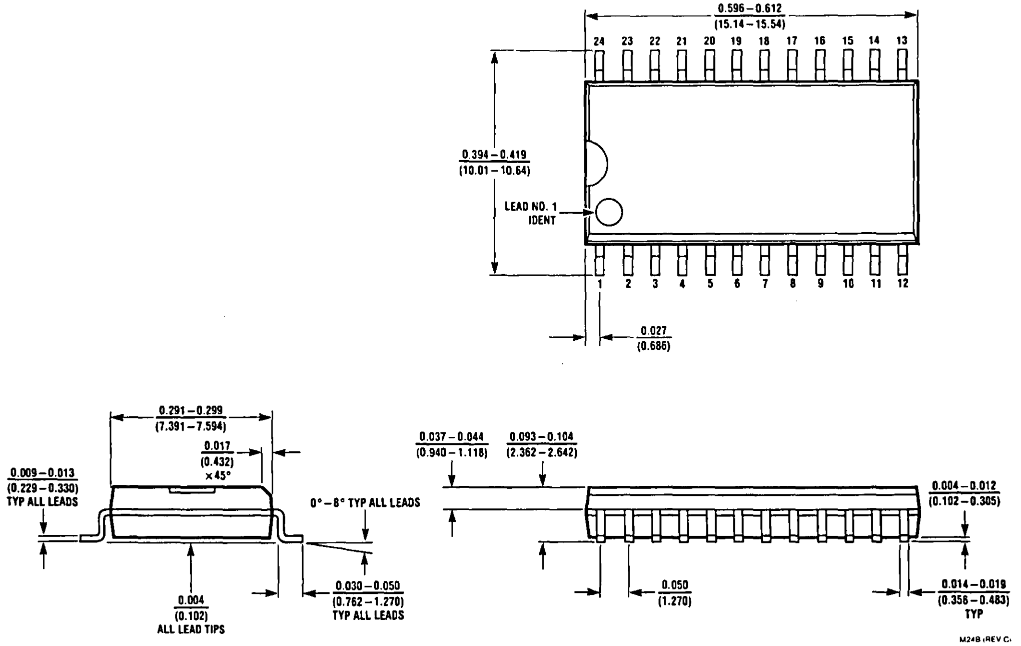
68 Lead Chip Carrier (E)
NS Package Number EL68A



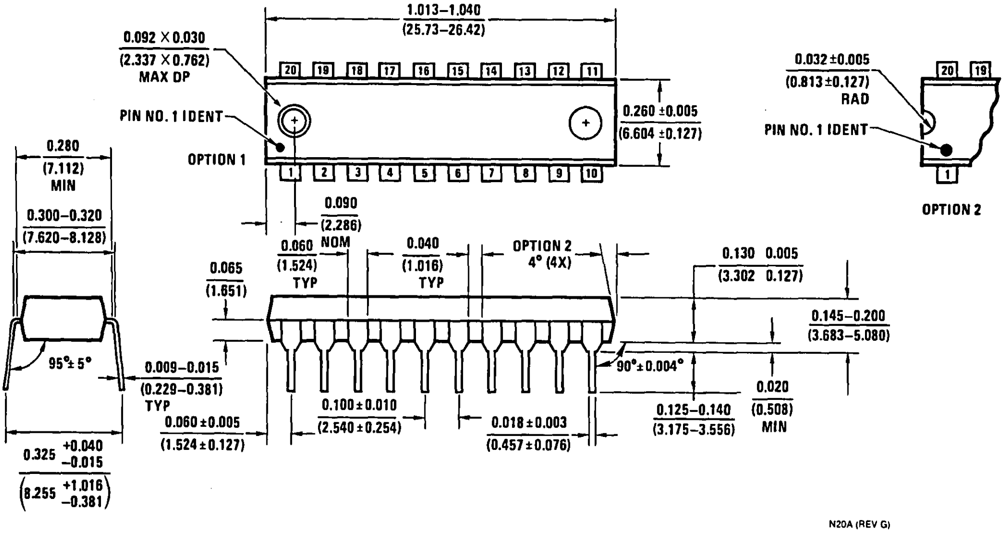
20 Lead (0.300″ Wide) Molded Small Outline Package (M)
NS Package Number M20B



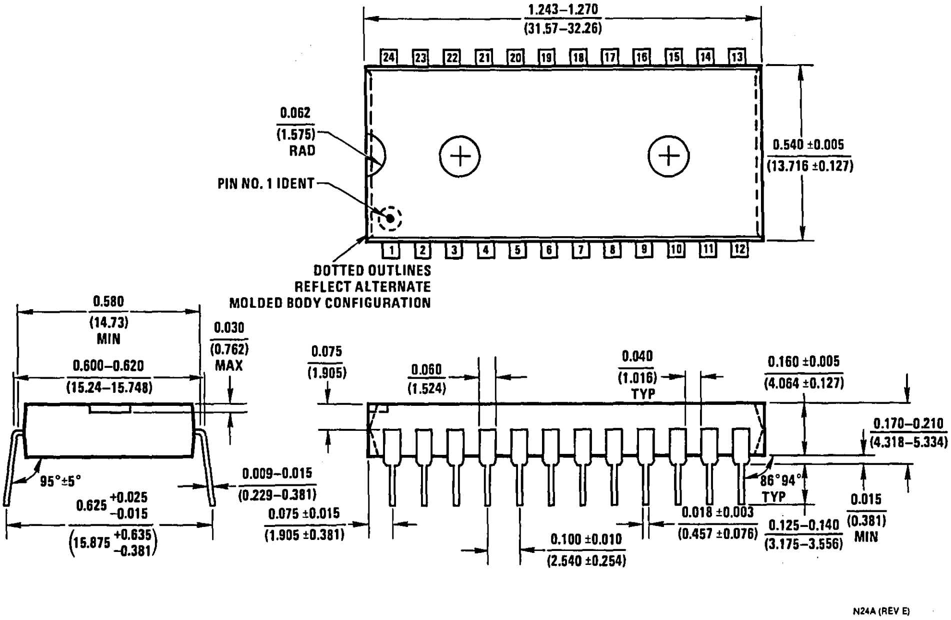
24 Lead (0.300″ Wide) Molded Small Outline Package (M)
NS Package Number M24B



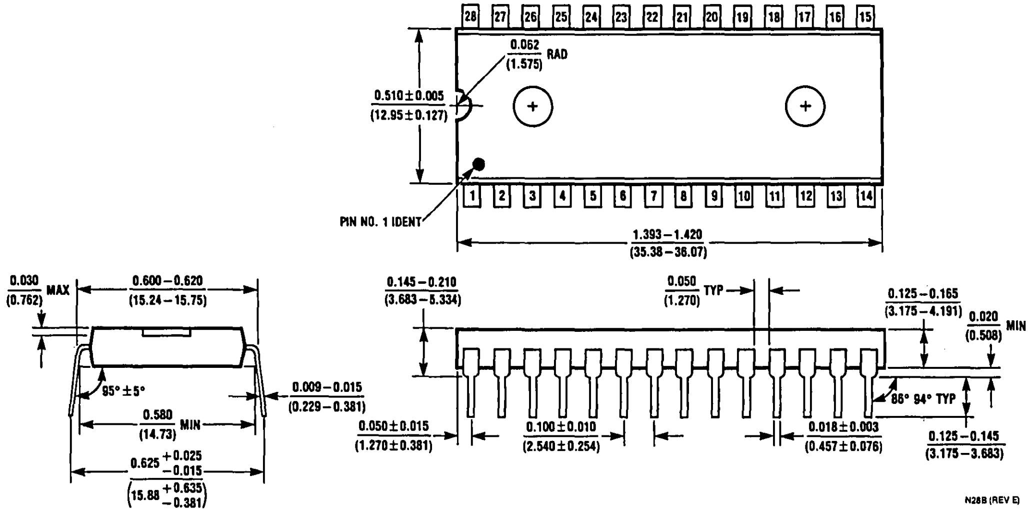
20 Lead Molded Dual-In-Line Package (N)
NS Package Number N20A



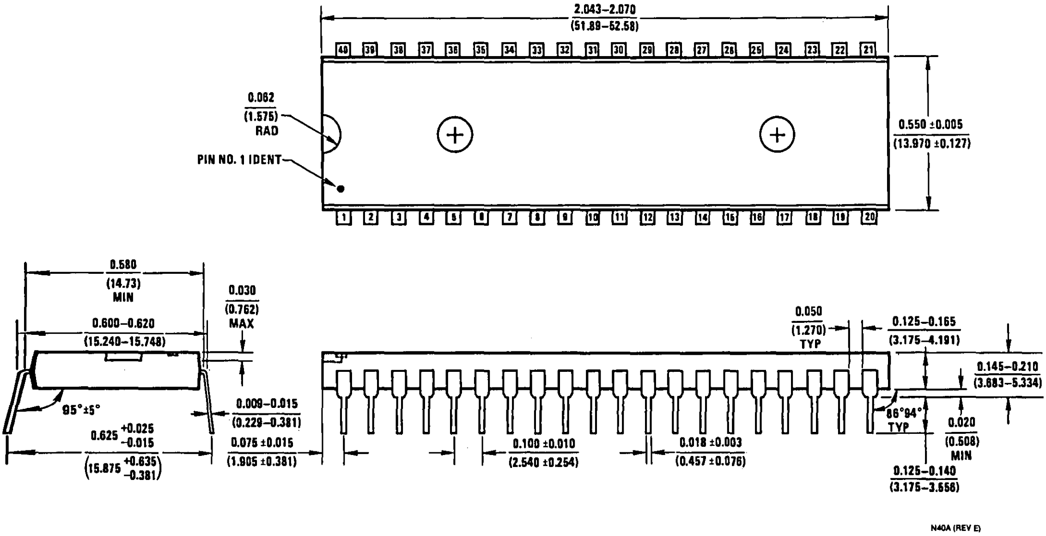
24 Lead Molded Dual-In-Line Package (N)
NS Package Number N24A



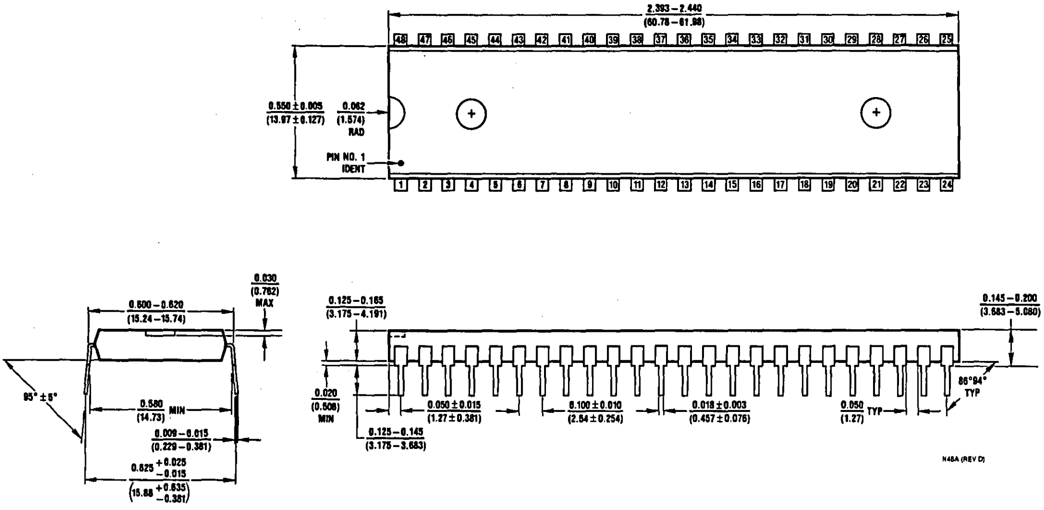
28 Lead Molded Dual-In-Line Package (N)
NS Package Number N28B



40 Lead Molded Dual-In-Line Package (N)
NS Package Number N40A

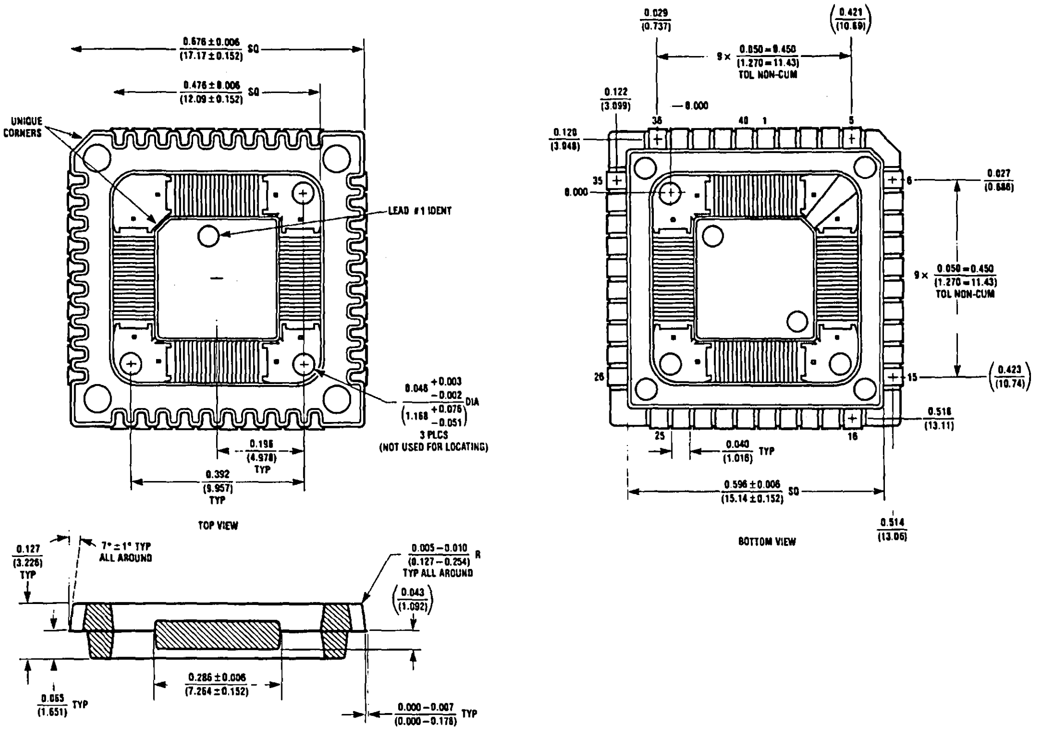


48 Lead Molded Dual-In-Line Package (N)
NS Package Number N48A



40 Lead TapePak® Package (TP)
NS Package Number TP40A

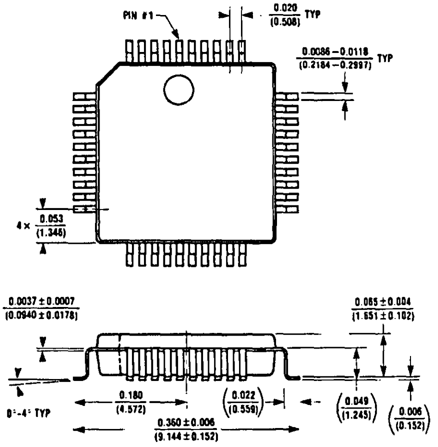
PACKAGE CONFIGURATION AS SHIPPED



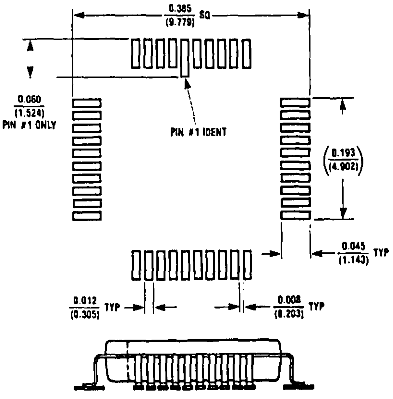
TOP VIEW

BOTTOM VIEW

RECOMMENDED FORMED AND EXCISED PACKAGE OUTLINE

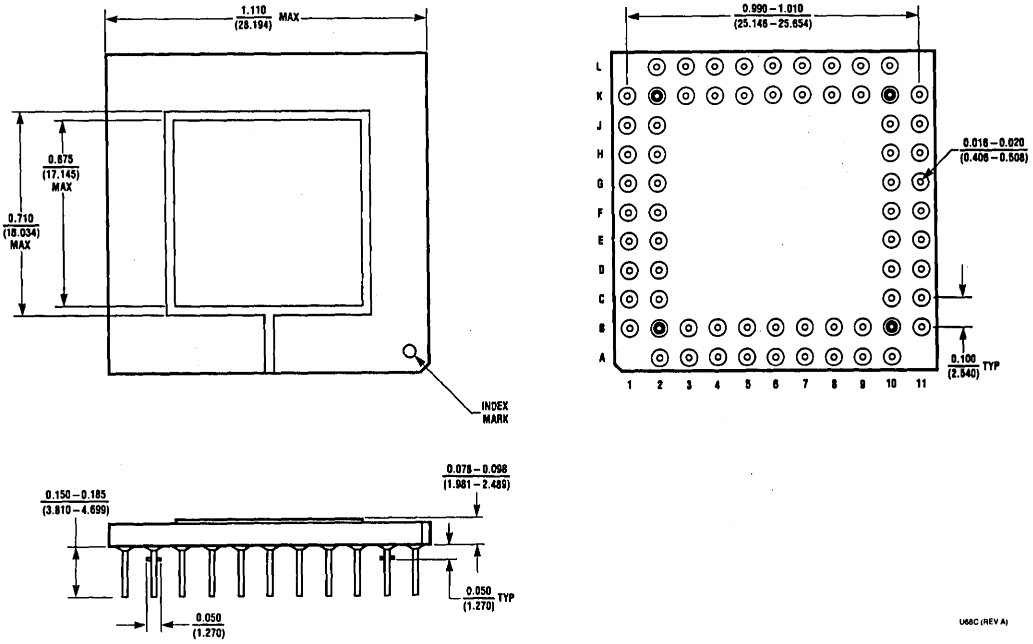


RECOMMENDED FOOTPRINT

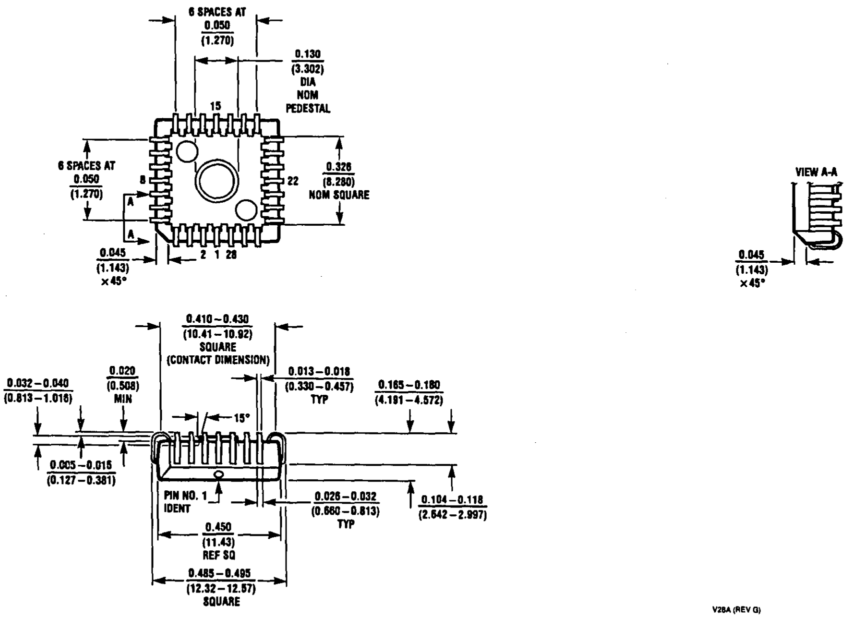


TP40A (REV A)

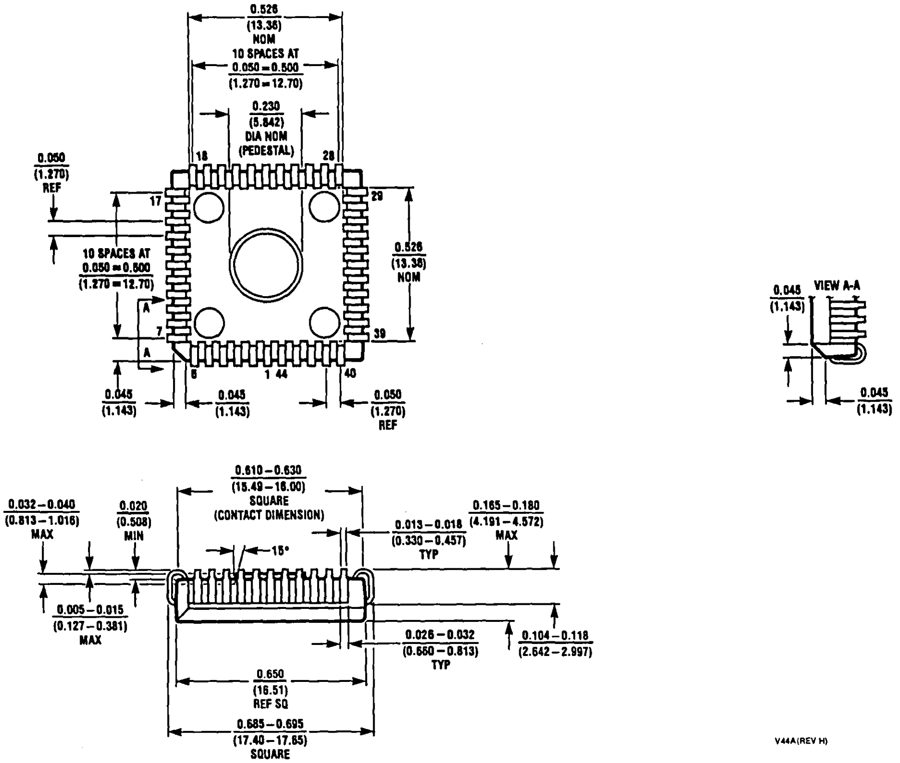
68 Lead Pin Grid Array (U)
NS Package Number U68C



28 Lead Plastic Chip Carrier (V)
NS Package Number V28A

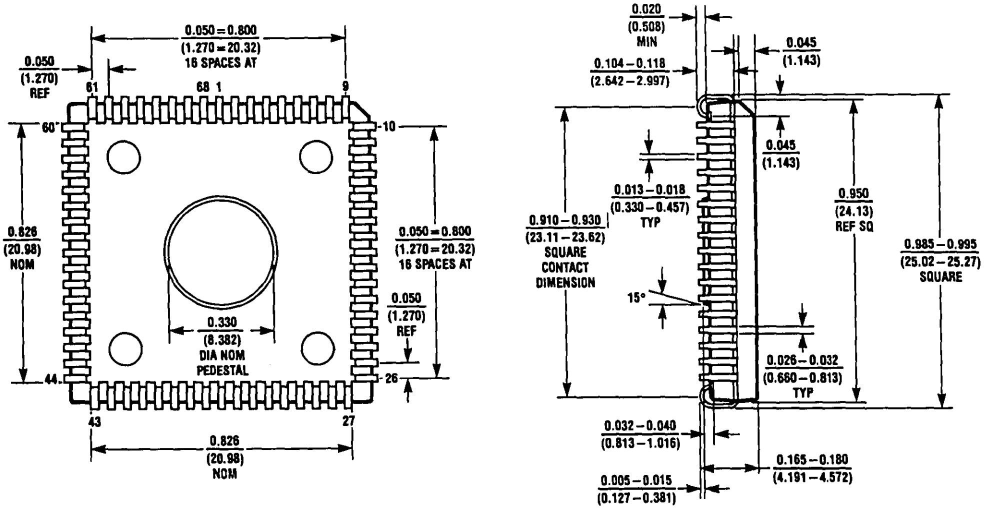


44 Lead Plastic Chip Carrier (V)
NS Package Number V44A



V44A (REV H)

68 Lead Plastic Chip Carrier (V)
NS Package Number V68A



V68A (REV G)

NOTES

National Semiconductor

Bookshelf of Technical Support Information

National Semiconductor Corporation recognizes the need to keep you informed about the availability of current technical literature.

This bookshelf is a compilation of books that are currently available. The listing that follows shows the publication year and section contents for each book.

Please contact your local National sales office for possible complimentary copies. A listing of sales offices follows this bookshelf.

We are interested in your comments on our technical literature and your suggestions for improvement.

Please send them to:

Technical Communications Dept. M/S 16300
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

ALS/AS LOGIC DATABOOK—1987

Introduction to Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky

ASIC DESIGN MANUAL/GATE ARRAYS & STANDARD CELLS—1987

SSI/MSI Functions • Peripheral Functions • LSI/VLSI Functions • Design Guidelines • Packaging

CMOS LOGIC DATABOOK—1988

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • Surface Mount

DATA ACQUISITION LINEAR DEVICES—1989

Active Filters • Analog Switches/Multiplexers • Analog-to-Digital Converters • Digital-to-Analog Converters Sample and Hold • Temperature Sensors • Voltage Regulators • Surface Mount

DATA COMMUNICATION/LAN/UART DATABOOK—Rev. 1—1988

LAN IEEE 802.3 • High Speed Serial/IBM Data Communications • ISDN Components • UARTs Modems • Transmission Line Drivers/Receivers

DISCRETE SEMICONDUCTOR PRODUCTS DATABOOK—1989

Selection Guide and Cross Reference Guides • Diodes • Bipolar NPN Transistors Bipolar PNP Transistors • JFET Transistors • Surface Mount Products • Pro-Electron Series Consumer Series • Power Components • Transistor Datasheets • Process Characteristics

DRAM MANAGEMENT HANDBOOK—1988

Dynamic Memory Control • Error Detection and Correction • Microprocessor Applications for the DP8408A/09A/17/18/19/28/29 • Microprocessor Applications for the DP8420A/21A/22A

F100K DATABOOK—1989

Family Overview • F100K Datasheets • 11C Datasheets • 10K and 100K Memory Datasheets Design Guide • Circuit Basics • Logic Design • Transmission Line Concepts • System Considerations Power Distribution and Thermal Considerations • Testing Techniques • Quality Assurance and Reliability

FACT™ ADVANCED CMOS LOGIC DATABOOK—1989

Description and Family Characteristics • Ratings, Specifications and Waveforms Design Considerations • 54AC/74ACXXX • 54ACT/74ACTXXX

FAST® ADVANCED SCHOTTKY TTL LOGIC DATABOOK—1988

Circuit Characteristics • Ratings, Specifications and Waveforms • Design Considerations • 54F/74FXXX

FAST® APPLICATIONS HANDBOOK—REPRINT

Reprint of 1987 Fairchild FAST Applications Handbook

Contains application information on the FAST family: Introduction • Multiplexers • Decoders • Encoders
Operators • FIFOs • Counters • TTL Small Scale Integration • Line Driving and System Design
FAST Characteristics and Testing • Packaging Characteristics • Index

GENERAL PURPOSE LINEAR DEVICES DATABOOK—1989

Continuous Voltage Regulators • Switching Voltage Regulators • Operational Amplifiers • Buffers • Voltage Comparators
Instrumentation Amplifiers • Surface Mount

GRAPHICS HANDBOOK—1989

Advanced Graphics Chipset • DP8500 Development Tools • Application Notes

INTERFACE DATABOOK—1988

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral Power Drivers • Display Drivers
Memory Support • Microprocessor Support • Level Translators and Buffers • Frequency Synthesis • Hi-Rel Interface

LINEAR APPLICATIONS HANDBOOK—1986

The purpose of this handbook is to provide a fully indexed and cross-referenced collection of linear integrated circuit applications using both monolithic and hybrid circuits from National Semiconductor.

Individual application notes are normally written to explain the operation and use of one particular device or to detail various methods of accomplishing a given function. The organization of this handbook takes advantage of this innate coherence by keeping each application note intact, arranging them in numerical order, and providing a detailed Subject Index.

LS/S/TTL DATABOOK—1987

Introduction to Bipolar Logic • Low Power Schottky • Schottky • TTL • Low Power

MASS STORAGE HANDBOOK—1989

Rigid Disk Pulse Detectors • Rigid Disk Data Separators/Synchronizers and ENDECs
Rigid Disk Data Controller • SCSI Bus Interface Circuits • Floppy Disk Controllers • Disk Drive Interface Circuits
Rigid Disk Preamplifiers and Servo Control Circuits • Rigid Disk Microcontroller Circuits • Disk Interface Design Guide

MEMORY DATABOOK—1988

PROMs, EPROMs, EEPROMs • Flash EPROMs and EEPROMs • TTL I/O SRAMs
ECL I/O SRAMs • ECL I/O Memory Modules

MICROCONTROLLER DATABOOK—1989

COP400 Family • COP800 Family • COPS Applications • HPC Family • HPC Applications
MICROWIRE and MICROWIRE/PLUS Peripherals • Microcontroller Development Tools

PROGRAMMABLE LOGIC DATABOOK & DESIGN MANUAL—1989

Product Line Overview • Datasheets • Designing with PLDs • PLD Design Methodology • PLD Design Development Tools
Fabrication of Programmable Logic • Application Examples

REAL TIME CLOCK HANDBOOK—1989

Real Time Clocks and Timer Clock Peripherals • Application Notes

RELIABILITY HANDBOOK—1986

Reliability and the Die • Internal Construction • Finished Package • MIL-STD-883 • MIL-M-38510
The Specification Development Process • Reliability and the Hybrid Device • VLSI/VHSIC Devices
Radiation Environment • Electrostatic Discharge • Discrete Device • Standardization
Quality Assurance and Reliability Engineering • Reliability and Documentation • Commercial Grade Device
European Reliability Programs • Reliability and the Cost of Semiconductor Ownership
Reliability Testing at National Semiconductor • The Total Military/Aerospace Standardization Program
883B/RETS™ Products • MILS/RETS™ Products • 883/RETS™ Hybrids • MIL-M-38510 Class B Products
Radiation Hardened Technology • Wafer Fabrication • Semiconductor Assembly and Packaging
Semiconductor Packages • Glossary of Terms • Key Government Agencies • AN/ Numbers and Acronyms
Bibliography • MIL-M-38510 and DESC Drawing Cross Listing

SERIES 32000 MICROPROCESSORS DATABOOK—1988

Series 32000 Overview • Central Processing Units • Slave Processors • Peripherals • Board Level Products
Development Systems and Tools • Software Support • Application Notes • NSC800 Family

SPECIAL PURPOSE LINEAR DEVICES DATABOOK—1989

Audio Circuits • Radio Circuits • Video Circuits • Motion Control Circuits • Special Function Circuits
Surface Mount

TELECOMMUNICATIONS—1987

Line Card Components • Integrated Services Digital Network Components • Modems
Analog Telephone Components • Application Notes

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS

ALABAMA
Huntsville
Arrow Electronics
(205) 837-6955
Bell Industries
(205) 837-1074
Hamilton/Avnet
(205) 837-7210
Pioneer Technology
(205) 837-9300

ARIZONA
Chandler
Hamilton/Avnet
(602) 231-5100
Phoenix
Arrow Electronics
(602) 437-0750
Tempe
Anthem Electronics
(602) 966-6600
Bell Industries
(602) 966-7800

CALIFORNIA
Agora Hills
Zeus Components
(818) 889-3838
Anaheim
Time Electronics
(714) 934-0911
Chatsworth
Anthem Electronics
(818) 700-1000
Arrow Electronics
(818) 701-7500
Hamilton Electro Sales
(818) 700-6500
Time Electronics
(818) 998-7200
Costa Mesa
Avnet Electronics
(714) 754-6050
Hamilton Electro Sales
(714) 641-4159
Garden Grove
Bell Industries
(714) 895-7801
Gardena
Bell Industries
(213) 515-1800
Hamilton/Avnet
(213) 217-6751
Irvine
Anthem Electronics
(714) 768-4444
Ontario
Hamilton/Avnet
(714) 989-4602
Rocklin
Anthem Electronics
(916) 624-9744
Bell Industries
(916) 652-0414
Sacramento
Hamilton/Avnet
(916) 925-2216
San Diego
Anthem Electronics
(619) 453-9005
Arrow Electronics
(619) 565-4800
Hamilton/Avnet
(619) 571-7510
Timo Electronics
(619) 586-1331
San Jose
Anthem Electronics
(408) 453-1200
Pioneer Technology
(408) 954-9100
Zous Components
(408) 998-5121
Sunnyvale
Arrow Electronics
(408) 745-6600
Bell Industries
(408) 734-8570
Hamilton/Avnet
(408) 743-3355
Time Electronics
(408) 734-9888
Thousand Oaks
Bell Industries
(805) 499-6821
Torrance
Time Electronics
(213) 320-0880
Tustin
Arrow Electronics
(714) 838-5422
Yorba Linda
Zeus Components
(714) 921-9000

COLORADO
Englewood
Anthem Electronics
(303) 790-4500
Arrow Electronics
(303) 790-4444
Hamilton/Avnet
(303) 799-7800
Wheatridge
Bell Industries
(303) 424-1985

CONNECTICUT
Cheshire
Time Electronics
(203) 271-3200
Danbury
Hamilton/Avnet
(203) 797-2800
Meriden
Anthem Electronics
(203) 237-2282
Norwalk
Pioneer Standard
(203) 853-1515
Wallingford
Arrow Electronics
(203) 265-7741

FLORIDA
Altamonte Springs
Bell Industries
(407) 339-0078
Pioneer Technology
(407) 834-9090
Clearwater
Pioneer Technology
(813) 536-0445
Deerfield Beach
Arrow Electronics
(305) 429-8200
Bell Industries
(305) 421-1997
Pioneer Technology
(305) 428-8877
Fort Lauderdale
Hamilton/Avnet
(305) 971-2900
Lake Mary
Arrow Electronics
(407) 333-9300
Largo
Bell Industries
(813) 541-4434
Oviedo
Zeus Components
(407) 365-3000
St. Petersburg
Hamilton/Avnet
(813) 576-3930
Winter Park
Hamilton/Avnet
(407) 628-3888

GEORGIA
Norcross
Arrow Electronics
(404) 449-8252
Bell Industries
(404) 662-0923
Hamilton/Avnet
(404) 447-7500
Pioneer Technology
(404) 448-1711

ILLINOIS
Addison
Pioneer Electronics
(312) 437-9680
Bensenville
Hamilton/Avnet
(312) 860-7780
Elk Grove Village
Anthem Electronics
(312) 640-6066
Bell Industries
(312) 640-1910
Itasca
Arrow Electronics
(312) 250-0500
Urbana
Bell Industries
(217) 328-1077
Wood Dale
Time Electronics
(312) 350-0610

INDIANA
Carmel
Hamilton/Avnet
(317) 844-9333
Fort Wayne
Bell Industries
(219) 423-3422
Indianapolis
Advent Electronics Inc.
(317) 872-4910
Arrow Electronics
(317) 243-9353
Bell Industries
(317) 634-8200
Pioneer Standard
(317) 849-7300

IOWA
Cedar Rapids
Advent Electronics
(319) 363-0221
Arrow Electronics
(319) 395-7230
Bell Industries
(319) 395-0730
Hamilton/Avnet
(319) 362-4757

KANSAS
Lenexa
Arrow Electronics
(913) 541-9542
Hamilton/Avnet
(913) 888-8900
Pioneer Standard
(913) 492-0500

MARYLAND
Columbia
Anthem Electronics
(301) 995-6640
Arrow Electronics
(301) 995-0003
Hamilton/Avnet
(301) 995-3500
Time Electronics
(301) 964-3090
Zeus Components
(301) 997-1118
Gaithersburg
Pioneer Technology
(301) 921-0660

MASSACHUSETTS
Andover
Bell Industries
(508) 474-8880
Lexington
Pioneer Standard
(617) 861-9200
Zeus Components
(617) 863-8800
Norwood
Gerber Electronics
(617) 769-6000
Peabody
Hamilton/Avnet
(508) 531-7430
Time Electronics
(508) 532-6200
Wilmington
Anthem Electronics
(508) 657-5170
Arrow Electronics
(508) 658-0900

MICHIGAN
Ann Arbor
Arrow Electronics
(313) 971-8220
Bell Industries
(313) 971-9093
Grand Rapids
Arrow Electronics
(616) 243-0912
Hamilton/Avnet
(616) 243-8805
Pioneer Standard
(616) 698-1800
Livonia
Pioneer Standard
(313) 525-1800
Novi
Hamilton/Avnet
(313) 347-4720
Wyoming
R. M. Electronics, Inc.
(616) 531-9300

MINNESOTA
Eden Prairie
Anthem Electronics
(612) 944-5454
Pioneer Standard
(612) 944-3355
Edina
Arrow Electronics
(612) 830-1800
Minnetonka
Hamilton/Avnet
(612) 932-0600

MISSOURI
Chesterfield
Hamilton/Avnet
(314) 537-1600
St. Louis
Arrow Electronics
(314) 567-6888
Time Electronics
(314) 391-6444

NEW HAMPSHIRE
Hudson
Bell Industries
(603) 882-1133
Manchester
Arrow Electronics
(603) 668-6968
Hamilton/Avnet
(603) 624-9400

NATIONAL SEMICONDUCTOR CORPORATION DISTRIBUTORS (Continued)

NEW JERSEY
Cherry Hill
Hamilton/Avnet
(609) 424-0100
Fairfield
Anthem Electronics
(201) 227-7960
Hamilton/Avnet
(201) 575-3390
Marlton
Arrow Electronics
(609) 596-8000
Parsippany
Arrow Electronics
(201) 538-0900
Pine Brook
Nu Horizons Electronics
(201) 882-8300
Pioneer Standard
(201) 575-3510
Time Electronics
(201) 882-4611

NEW MEXICO
Albuquerque
Alliance Electronics Inc.
(505) 292-3360
Arrow Electronics
(505) 243-4566
Bell Industries
(505) 292-2700
Hamilton/Avnet
(505) 765-1500

NEW YORK
Amityville
Nu Horizons Electronics
(516) 226-6000
Binghamton
Pioneer
(607) 722-9300
Buffalo
Summit Electronics
(716) 887-2800
Fairport
Pioneer Standard
(716) 381-7070
Time Electronics
(716) 383-8853
Hauppauge
Anthem Electronics
(516) 273-1660
Arrow Electronics
(516) 231-1000
Hamilton/Avnet
(516) 434-7413
Time Electronics
(516) 273-0100
Port Chester
Zeus Components
(914) 937-7400
Rochester
Arrow Electronics
(716) 427-0300
Hamilton/Avnet
(716) 475-9130
Summit Electronics
(716) 334-8110
Ronkonkoma
Zeus Components
(516) 737-4500
Syracuse
Hamilton/Avnet
(315) 437-2641
Time Electronics
(315) 432-0355
Westbury
Hamilton/Avnet Export Div.
(516) 997-6868
Woodbury
Pioneer Electronics
(516) 921-8700

NORTH CAROLINA
Charlotte
Pioneer Technology
(704) 527-8188
Time Electronics
(704) 522-7600
Durham
Pioneer Technology
(919) 544-5400
Raleigh
Arrow Electronics
(919) 876-3132
Hamilton/Avnet
(919) 878-0810
Winston-Salem
Arrow Electronics
(919) 725-8711

OHIO
Centerville
Arrow Electronics
(513) 435-5563
Bell Industries
(513) 435-8660
Bell Industries-Military
(513) 434-8231
Cleveland
Pioneer
(216) 587-3600
Dayton
Hamilton/Avnet
(513) 439-6700
Pioneer Standard
(513) 236-9900
Zeus Components
(914) 937-7400
Solon
Arrow Electronics
(216) 248-3990
Hamilton/Avnet
(216) 831-3500
Westerville
Hamilton/Avnet
(614) 882-7004

OKLAHOMA
Tulsa
Arrow Electronics
(918) 252-7537
Hamilton/Avnet
(918) 252-7297
Radio Inc.
(918) 587-9123

OREGON
Beaverton
Almac-Stroum Electronics
(503) 629-8090
Anthem Electronics
(503) 643-1114
Arrow Electronics
(503) 645-6456
Hamilton/Avnet
(503) 627-0201
Lake Oswego
Bell Industries
(503) 635-6500

PENNSYLVANIA
Horsham
Anthem Electronics
(215) 443-5150
Pioneer Technology
(215) 674-4000
King of Prussia
Time Electronics
(215) 337-0900
Monroeville
Arrow Electronics
(412) 856-7000
Pittsburgh
Hamilton/Avnet
(412) 281-4150
Pioneer
(412) 782-2300

TEXAS
Austin
Arrow Electronics
(512) 835-4180
Hamilton/Avnet
(512) 837-8911
Pioneer Standard
(512) 835-4000
Time Electronics
(512) 399-3051
Carrollton
Arrow Electronics
(214) 380-6464
Time Electronics
(214) 241-7441
Dallas
Hamilton/Avnet
(214) 404-9906
Pioneer Standard
(214) 386-7300
Houston
Arrow Electronics
(713) 530-4700
Pioneer Standard
(713) 988-5555
Richardson
Anthem Electronics
(214) 238-7100
Zeus Components
(214) 783-7010
Stafford
Hamilton/Avnet
(713) 240-7733

UTAH
Midvale
Bell Industries
(801) 255-9611
Salt Lake City
Anthem Electronics
(801) 973-8555
Arrow Electronics
(801) 973-6913
Hamilton/Avnet
(801) 972-4300
West Valley
Time Electronics
(801) 973-8181

WASHINGTON
Bellevue
Almac-Stroum Electronics
(206) 643-9992
Bothell
Anthem Electronics
(206) 483-1700
Kent
Arrow Electronics
(206) 575-4420
Redmond
Hamilton/Avnet
(206) 881-6697

WISCONSIN
Brookfield
Arrow Electronics
(414) 792-0150
Mequon
Taylor Electric
(414) 241-4321
Waukesha
Bell Industries
(414) 547-8879
Hamilton/Avnet
(414) 784-4516

CANADA
WESTERN PROVINCES
Burnaby
Hamilton/Avnet
(604) 437-6667
Semad Electronics
(604) 420-9889
Calgary
Hamilton/Avnet
(403) 250-9380
Semad Electronics
(403) 252-5664
Zentronics
(403) 272-1021
Edmonton
Zentronics
(403) 468-9306
Richmond
Zentronics
(604) 273-5575
Saskatoon
Zentronics
(306) 955-2207
Winnipeg
Zentronics
(204) 694-1957

EASTERN PROVINCES
Brampton
Zentronics
(416) 451-9600
Mississauga
Hamilton/Avnet
(416) 677-7432
Nepean
Hamilton/Avnet
(613) 226-1700
Zentronics
(613) 226-8840
Ottawa
Semad Electronics
(613) 727-8325
Pointe Claire
Semad Electronics
(514) 694-0860
St. Laurent
Hamilton/Avnet
(514) 335-1000
Zentronics
(514) 737-9700
Willowdale
ElectroSonic Inc.
(416) 494-1666

SALES OFFICES

ALABAMA
Huntsville
(205) 721-9367

ARIZONA
Tempe
(602) 966-4563

CALIFORNIA
Inglewood
(213) 645-4226
Roseville
(916) 786-5577
San Diego
(619) 587-0666
Santa Clara
(408) 562-5900
Tustin
(714) 259-8880
Woodland Hills
(818) 888-2602

COLORADO
Boulder
(303) 440-3400
Colorado Springs
(303) 578-3319
Englewood
(303) 790-8090

CONNECTICUT
Hamden
(203) 288-1560

FLORIDA
Boca Raton
(407) 997-8133
Orlando
(305) 629-1720
St. Petersburg
(813) 577-1380

GEORGIA
Norcross
(404) 441-2740

ILLINOIS
Schaumburg
(312) 397-8777

INDIANA
Carmel
(317) 843-7160
Fort Wayne
(219) 484-0722

IOWA
Cedar Rapids
(319) 395-0090

KANSAS
Overland Park
(913) 451-4402

MARYLAND
Hanover
(301) 796-8900

MASSACHUSETTS
Burlington
(617) 273-3170

MICHIGAN
Grand Rapids
(616) 940-0588
W. Bloomfield
(313) 855-0166

MINNESOTA
Bloomington
(612) 854-8200

NEW JERSEY
Paramus
(201) 599-0955

NEW MEXICO
Albuquerque
(505) 884-5601

NEW YORK
Fairport
(716) 223-7700
Liverpool
(315) 451-9091
Melville
(516) 351-1000
Wappinger Falls
(914) 298-0680

NORTH CAROLINA
Cary
(919) 481-4311

OHIO
Dayton
(513) 435-6886
Dublin
(614) 766-3679
Independence
(216) 524-5577

ONTARIO
Mississauga
(416) 678-2920
Nopean
(613) 596-0411

OREGON
Portland
(503) 639-5442

PENNSYLVANIA
Horsham
(215) 672-6767

PUERTO RICO
Rio Piodras
(809) 758-9211

QUEBEC
Lachine
(514) 636-8525

TEXAS
Austin
(512) 346-3990
Houston
(713) 771-3547
Richardson
(214) 234-3811

UTAH
Salt Lako City
(801) 322-4747

WASHINGTON
Bellevue
(206) 453-9944

WISCONSIN
Brookfield
(414) 782-1818

National Semiconductor

BELL INDUSTRIES
Electronic Distribution Group
1161 N. Fair Oaks Avenue
Sunnyvale, California 94089
408 734-8570
FAX NO. 408 734-8875

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-5000
TWX: (910) 339-9240

SALES OFFICES (Continued)

INTERNATIONAL OFFICES

Electronica NSC de Mexico SA
Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: 52-5-524-9402

National Semicondutores
Do Brasil Ltda.
Av. Brig. Faria Lima, 1383
6.0 Andor-Conj. 62
01451 Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Fax: (55/11) 211-1181 NSBR BR

National Semiconductor GmbH
Industriestrasse 10
D-8080 Furstenfeldbruck
West Germany
Tel: (0-81-41) 103-0
Telex: 527-649

National Semiconductor (UK) Ltd.
The Maples, Kembrey Park
Swindon, Wiltshire SN2 6UT
United Kingdom
Tel: (07-93) 614141
Telex: 444674
Fax: (07-93) 697522

National Semiconductor Benelux
Vorstlaan 100
B-1170 Brussels
Belgium
Tel: (02) 6-61-06-80
Telex: 61007

National Semiconductor (UK) Ltd.
Ringager 4A, 3
DK-2605 Brondby
Denmark
Tel: (02) 43-32-11
Telex: 15179
Fax: (02) 43-31-11

National Semiconductor S.A.
Centre d'Affaires-La Boursidiere
Bâtiment Champagne, BP 90
Route Nationale 186
F-92357 Le Plessis Robinson
France
Tel: (1) 40-94-88-88
Telex: 631065
Fax: (1) 40-94-88-11

National Semiconductor (UK) Ltd.
Unit 2A
Clonskeagh Square
Clonskeagh Road
Dublin 14
Tel: (01) 69-55-89
Telex: 91047
Fax: (01) 69-55-89

National Semiconductor S.p.A.
Strada 7, Palazzo R/3
20089 Rozzano
Milanofiori
Italy
Tel: (02) 8242046/7/8/9

National Semiconductor S.p.A.
Via del Caravaggio, 107
00147 Rome
Italy
Tel: (06) 5-13-48-80
Fax: (06) 5-13-79-47

National Semiconductor (UK) Ltd.
P.O. Box 29
N-1321 Stabekk
Norway
Tel: (2) 12-53-70
Fax: (2) 12-53-75

National Semiconductor AB
Box 2016
Stensätravägen 13
S-12702 Skärholmen
Sweden
Tel: (08) 970190
Telex: 10731

National Semiconductor
Calle Agustin de Foxa, 27
28036 Madrid
Spain
Tel: (01) 733-2958
Telex: 46133

National Semiconductor
Switzerland
Alte Winterthurerstrasse 53
Postfach 567
CH-8304 Wallisellen-Zürich
Switzerland
Tel: (01) 830-2727
Telex: 828-444

National Semiconductor
Kauppakartanonkatu 7 A22
SF-00930 Helsinki
Finland
Tel: (90) 33-80-33
Telex: 126116

National Semiconductor
Postbus 90
1380 AB Weesp
The Netherlands
Tel: (0-2940) 3-04-48
Telex: 10-956
Fax: (0-2940) 3-04-30

National Semiconductor Japan
Ltd.
Sanseido Bldg. 5F
4-15 Nishi Shinjuku
Shinjuku-ku,
Tokyo 160 Japan
Tel: 3-299-7001
Fax: 3-299-7000

National Semiconductor
Hong Kong Ltd.
Suite 513, 5th Floor
Chinachem Golden Plaza,
77 Mody Road, Tsimshatsui East,
Kowloon, Hong Kong
Tel: 3-7231290
Telex: 52996 NSSEA HX
Fax: 3-3112536

National Semiconductor
(Australia) PTY, Ltd.
1st Floor, 441 St. Kilda Rd.
Melbourne, 3004
Victory, Australia
Tel: (03) 267-5000
Fax: 61-3-2677458

National Semiconductor (PTE),
Ltd.
200 Cantonment Road 13-01
Southpoint
Singapore 0208
Tel: 2252226
Telex: RS 33877

National Semiconductor (Far East)
Ltd.
Taiwan Branch
P.O. Box 68-332 Taipei
7th Floor, Nan Shan Life Bldg.
302 Min Chuan East Road,
Taipei, Taiwan R.O.C.
Tel: (86) 02-501-7227
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (Far East)
Ltd.
Korea Branch
13th Floor, Dai Han Life Insurance
63 Building,
60, Yoido-dong, Youngdeungpo-ku,
Seoul, Korea 150-763
Tel: (02) 784-8051/3 - 785-0696-8
Telex: K24942 NSRKLO
Fax: (02) 784-8054

© 1988 National Semiconductor Corp.

National Semiconductor

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-5000
TWX: (910) 339-9240

BELL INDUSTRIES
Electronic Distribution Group
1161 N. Fairoaks Avenue
Sunnyvale, California 94089
(408) 734-8570
FAX NO. (408) 734-8875

SALES OFFICES (Continued)

INTERNATIONAL OFFICES

Electronica NSC de Mexico SA
Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: 52-5-524-9402

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 1383
6.0 Andor-Conj. 62
01451 Sao Paulo, SP, Brasil
Tel: (55/11) 212-5066
Fax: (55/11) 211-1181 NSBR BR

National Semiconductor GmbH
Industriestrasse 10
D-8080 Furstenfeldbruck
West Germany
Tel: (0-81-41) 103-0
Telex: 527-649

National Semiconductor (UK) Ltd.
The Maple, Kembrey Park
Swindon, Wiltshire SN2 6UT
United Kingdom
Tel: (07-93) 61-41-41
Telex: 444-674
Fax: (07-93) 69-75-22

National Semiconductor Benelux
Vorstlaan 100
B-1170 Brussels
Belgium
Tel: (02) 6-61-06-80
Telex: 61007

National Semiconductor (UK) Ltd.
Ringager 4A, 3
DK-2605 Brondby
Denmark
Tel: (02) 43-32-11
Telex: 15-179
Fax: (02) 43-31-11

National Semiconductor S.A.
Centre d'Affaires-La Boursidiere
Bâtiment Champagne, B.P. 90
Route Nationale 186
F-92357 Le Plessis Robinson
France
Tel: (1) 40-94-88-88
Telex: 631065
Fax: (1) 40-94-88-11

National Semiconductor (UK) Ltd.
Unit 2A
Clonskeagh Square
Clonskeagh Road
Dublin 14
Tel: (01) 69-55-89
Telex: 91047
Fax: (01) 69-55-89

National Semiconductor S.p.A.
Strada 7, Palazzo R/3
20089 Rozzano
Milanofiori
Italy
Tel: (02) 8242046/7/8/9

National Semiconductor S.p.A.
Via del Cararaggio, 107
00147 Rome
Italy
Tel: (06) 5-13-48-80
Fax: (06) 5-13-79-47

National Semiconductor (UK) Ltd.
P.O. Box 29
N-1321 Stabekk
Norway
Tel: (2) 12-53-70
Fax: (2) 12-53-75

National Semiconductor AB
Box 2016
Stensatravagen 13
S-12702 Skarholmen
Sweden
Tel: (08) 970190
Telex: 10731

National Semiconductor
Calle Agustin de Foxa, 27
28036 Madrid
Spain
Tel: (01) 733-2958
Telex: 46133

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
Ch-8304 Wallisellen-Zurich
Switzerland
Tel: (01) 830-2727
Telex: 828-444

National Semiconductor
Kauppakartanonkatu 7 A22
SF-00930 Helsinki
Finland
Tel: (90) 33-80-33
Telex: 126116

National Semiconductor
Postbus 90
1380 AB Weesp
The Netherlands
Tel: (0-29-40) 3-04-48
Telex: 10-956
Fax: (0-29-40) 3-04-30

National Semiconductor Japan Ltd.
Sanseido Bldg. 5F
4-15 Nishi Shinjuku
Shinjuku-ku
Tokyo 160 Japan
Tel: 3-299-7001
Fax: 3-299-7000

National Semiconductor Hong Kong Ltd.
Suite 513, 5th Floor,
Chinachem Golden Plaza,
77 Mody Road, Tsimshatsui East,
Kowloon, Hong Kong
Tel: 3-7231290
Telex: 52996 NSSEA HX
Fax: 3-3112536

National Semiconductor (Australia) PTY, Ltd.
1st Floor, 441 St. Kilda Rd.
Melbourne, 3004
Victory, Australia
Tel: (03) 267-5000
Fax: 61-3-2677458

National Semiconductor (PTE), Ltd.
200 Cantonment Road 13-01
Southpoint
Singapore 0208
Tel: 2252226
Telex: RS 33877

National Semiconductor (Far East) Ltd.
Taiwan Branch
P.O. Box 68-332 Taipei
7th Floor, Nan Shan Life Bldg.
302 Min Chuan East Road,
Taipei, Taiwan R.O.C.
Tel: (86) 02-501-7227
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (Far East) Ltd.
Korea Branch
13th Floor, Dai Han Life Insurance
63 Building,
60, Yoido-dong, Youngdeungpo-ku,
Seoul, Korea 150-763
Tel: (02) 784-8051/3, 785-0696/8
Telex: 24942 NSPKLO
Fax: (02) 784-8054

© 1989 National Semiconductor Corp. TL2630 RRD-RRD90M079/Printed in U.S.A.