400094

National Semiconductor Corporation

Series 32000® Databook

1986

Series 32000 Databook

National Semiconductor

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.

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 produisant 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.

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

Series 32000

DATABOOK

Introduction 1

CPU—Central Processing Units 2

Slave Processors 3

Peripherals 4

Data Communications and LANs 5

Disk Control and Interface 6

DRAM Interface 7

Development Tools 8

Software Support 9

Application Notes 10

Physical Dimensions/Appendices 11

TRADEMARKS

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

Abuseable™	ELSTAR™	Naked-8™	STAR™
Anadig™	E-Z-LINK™	National®	Starlink™
ANS-R-TRAN™	GENIX™	NAX 800™	STARPLEX™
APPS™	GNX™	Nitride Plus™	STARPLEX II™
Auto-Chem Deflasher™	HEX 3000™	Nitride Plus Oxide™	SuperChip™
BI-FET™	INFOCHEX™	NML™	SYS32™
BI-FET II™	Integral ISE™	NOBUS™	TAPE-PAK™
BI-LINE™	Intelisplay™	NSC800™	TDS™
BIPLAN™	ISE™	NSX-16™	TeleGate™
BLC™	ISE/06™	NS-XC-16™	The National Anthem®
BLX™	ISE/08™	NURAM™	Time✓Chek™
Brite-Lite™	ISE/16™	OXISS™	TLC™
BTL™	ISE32™	Perfect Watch™	Trapezoidal™
CIM™	LMCMOS™	Pharma✓Chek™	TRI-CODE™
CIMBUS™	Macrobus™	PLAN™	TRI-POLY™
Clock✓Chek™	Macrocomponent™	Polycraft™	TRI-SAFE™
COMBO™	Meat✓Chek™	POSitalker™	TRI-STATE®
COPS™ microcontrollers	Microbus™ data bus (adjective)	QUAD3000™	TURBOTRANSCEIVER™
DATACHECKER®	MICRO-DAC™	RAT™	VIP™
DENSPAK™	μtalker™	RTX16™	VR32™
DIB™	Microtalker™	Script✓Chek™	XMOS™
Digitalker®	MICROWIRE™	SCX™	XPU™
DISCERN™	MICROWIRE/PLUS™	Shelf-Chek™	Z STAR™
DISTILL™	MOLE™	SERIES/800™	883B/RETS™
DNR™	MST™	Series 32000®	883S/RETS™
DPVM™		SPIRE™	

Ada® is a registered trademark of the U.S. Government, Ada Joint Program Office
VERDIX and VADS are trademarks of the VERDIX Corporation
UNIX® is a registered trademark of AT&T.
IBM® is a registered trademark of International Business Machines Corporation
VisiCalc is a trademark of Visi Corporation
VAX™, VMS™, DEC™, PDP-11™, RSX-11™ are trademarks of Digital Equipment Corporation
CP/M™ is a trademark of Digital Research Corporation
Z80® is a registered trademark of Zilog Corporation
MULTIBUS® is a registered trademark of Intel Corporation
Model 19™ is a trademark of DATA I/O Corporation
VRTX®, IOX®, FMX® are registered trademarks of Hunter & Ready Corporation
TRACER™ is a trademark of Hunter & Ready Corporation
PAL® and PALASM™ are trademarks of and are used under license from Monolithic Memories, Inc.

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

Table of Contents

Section 1

Introduction 1-3
Key Features of Series 32000 1-4
Series 32000 Component Descriptions 1-5
Series 32000 Part Numbering Scheme 1-6
Hardware Chart 1-8
Systems and Software Chart 1-9
Support Devices 1-10
Datasheet Description 1-11
Military/Aerospace Program 1-12
Series 32000 Programs and Services 1-15
Introduction to Series 32000 Architecture 1-16
Benefits of Demand Paged Virtual Memory 1-52

Section 2 CPU—Central Processing Units

NS32332-10, -12, -15 32-Bit Advanced Microprocessor with Virtual Memory 2-3
NS32132-6, -8, -10 High-Performance Microprocessors 2-76
NS32032-6, -8, -10 High-Performance Microprocessors 2-146
NS32C016-6, -10, -15 High-Performance Microprocessors 2-211
NS32016-6, -8, -10 High-Performance Microprocessors 2-275
NS32008-6, -8, -10 High-Performance Microprocessors 2-339

Section 3 Slave Processors

NS32081-6, -8, -10 Floating Point Unit (FPU) 3-3
NS32082-6, -8, -10 Memory Management Unit (MMU) 3-20
NS32382-10, -15 Memory Management Unit 3-59

Section 4 Peripherals

NS32C201-6, -10, -15 Timing Control Unit (TCU) 4-3
NS32201-6, -8, -10 Timing Control Unit (TCU) 4-25
NS32301-10, -15 Advanced Timing Control Unit (TCU) 4-49
NS32202-6, -8, -10 Interrupt Control Unit (ICU) 4-64
NS16450/INS8250A Asynchronous Communications Element 4-89
NS16550 Asynchronous Communications Element with FIFOs 4-104

Section 5 Data Communications and LANs

HPC16040-High Performance Microcontroller 5-3
NS32405/NS405—Series Display/Terminal Management Processor 5-26
NS32490/DP8390 Network Interface Controller 5-65
NS32491/DP8391 Serial Network Interface 5-113
NS32492/DP8392 Coaxial Transceiver Interface 5-122
NS32440/DP8340 Serial Biphase Serial Transmitter/Encoder 5-130
NS32441/DP8341 Serial Biphase Serial Receiver/Decoder 5-139
NS32442/DP8342 High Speed Serial Transmitter/Encoder 5-150
NS32443/DP8343 High Speed Receiver/Decoder 5-160

Section 6 Disk Control and Interface

NS32951/DP8451 Data Synchronizer 6-3
NS32955/DP8455 Data Synchronizer 6-3
NS32961/DP8461 Data Separator 6-3
NS32965/DP8465 Data Separator 6-3
NS32962/DP8462 2, 7 Code Data Synchronizer 6-29
NS32963/DP8463B 2, 7 ENDEC 6-49
NS32964/DP8464B Disk Pulse Detector 6-58
NS32966/DP8466 Disk Data Controller 6-81
NS32970/DP8470 Floppy Data Separator and Write Precompensation 6-133
NS32972/DP8472 & NS32974/DP8474 Floppy Disk Controller Plus 6-143

Table of Contents (Continued)

Section 7 DRAM Interface

NS32800-2/DP8400-2 E2C2 Expandable Error Checker/Corrector 7-3
NS32802A/DP8402A 32-bit Parallel Error Detection and Correction Circuits (EDAC's) 7-37
NS32809A/DP8409A Multi-Mode Dynamic RAM Controller/Driver 7-53
NS32812/DP84412 Dynamic RAM Controller Interface Series Circuit for the Series 32000 CPU 7-75
NS32817/18/19/DP8417/18/19 64k, 256k Dynamic RAM Controller/Drivers 7-88
NS32813/DP84512 Dynamic RAM Controller Interface Circuit for the NS32332 7-113
NS32828/DP8428 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114
NS32829/DP8429 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114

Section 8 Development Tools

SYS32/20 Development System 8-3
VR32—Target/Development System 8-4
In-System Emulators
ISE32—NS32032 In-System Emulator 8-8
ISE16—NS32016 In-System Emulator 8-17
Development Boards
DB32000 Development Board 8-28
DB32016 Development Board 8-33
OEM Products
ICM-3332 Integrated Computer Module (NS32332 Based) 8-39
ICM-3216 Integrated Computer Module (NS32016 Based) 8-46

Section 9 Software Support

System V/Series 32000 Operating System 9-3
Series 32000 GENIX 4.2 Operating System 9-5
Series 32000 GENIX Native and Cross-Support (GNX) Language Tools 9-7
Series 32000 EXEC ROMable Real-Time Multitasking EXECUTIVE 9-11
Series 32000 Real Time Software Components VRTX/IOX/FMX/TRACER 9-15
Series 32000 VERDIX Ada Development System (VADS) for System V/Series 32000 Native Host 9-19
Series 32000 VERDIX Ada Development System (VADS) for VAX/VMS 9-21

Section 10 Application Notes

AN-383 Interfacing the NS32081 as a Floating-Point Peripheral 10-3
AB-26 Instruction Execution Times of FPU NS32081 Considered for Stand-Alone Configurations 10-11
AN-396 Debugging a 32032 Based System with an ISE16 10-12
AN-404 10 MHz No Wait States NS32016 System 10-20
AN-405 Using Dynamic RAM with Series 32000 CPUs 10-31
AN-406 Interfacing the Series 32000 CPUs to the MULTIBUS 10-38
AN-449 A Systems-Oriented Microprocessor Bus 10-43

Section 11 Appendices

Glossary of Terms 11-3
Programming Reference Guide 11-10
Series 32000 Quick Reference Timing 11-42
Physical Dimensions 11-63
Data Bookshelf
Sales and Distribution Offices

Alpha-Numeric Index

AB-26 Instruction Execution Times of FPU NS32081 Considered for Stand-Alone Configurations 10-11
AN-383 Interfacing the NS32081 as a Floating-Point Peripheral 10-3
AN-396 Debugging a 32032 Based System With an ISE16 10-11
AN-404 10 MHz, No Wait States NS32016 System 10-20
AN-405 Using Dynamic RAM With Series 32000 CPU's 10-31
AN-406 Interfacing the Series 32000 CPUs to the MULTIBUS 10-38
AN-449 A Systems-Oriented Microprocessor Bus 10-43
Benefits of Demand Paged Virtual Memory 1-52
DB32000 Development Board 8-28
DB32016 Development Board 8-33
DP8340 Serial Bi-Phase Transmitter/Encoder 5-130
DP8341 Serial Bi-Phase Receiver/Decoder 5-139
DP8342 High-Speed Serial Transmitter/Encoder 5-150
DP8343 High-Speed Serial Receiver/Decoder 5-160
DP8390 Network Interface Controller 5-65
DP8391 Serial Network Interface 5-113
DP8392 Coaxial Transceiver Interface 5-122
DP8400-2 E^2C^2 Expandable Error Checker/Corrector 7-3
DP8402A 32-Bit Parallel Error Detection and Correction Circuits (EDAC's) 7-37
DP8409A Multi-Mode Dynamic RAM Controller/Driver 7-53
DP8417 64K, 256K Dynamic RAM Controller/Drivers 7-88
DP8418 64K, 256K Dynamic RAM Controller/Drivers 7-88
DP8419 64K, 256K Dynamic RAM Controller/Drivers 7-88
DP8428 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114
DP8429 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114
DP8451 Data Synchronizer 6-3
DP8455 Data Synchronizer 6-3
DP8461 Data Separator 6-3
DP8462 2, 7 Code Data Synchronizer 6-29
DP8463B 2, 7 ENDEC 6-49
DP8464B Disk Pulse Detector 6-58
DP8465 Data Separator 6-3
DP8466 Disk Data Controller 6-81
DP8470 Floppy Disk Support Chip Data Separator and Write Precompensation 6-133
DP8472 Floppy Disk Controller Plus 6-143
DP8474 Floppy Disk Controller Plus 6-143
DP84412 Dynamic RAM Controller Interface Series Circuit for the Series 32000 CPU 7-75
DP84512 Dynamic RAM Controller Interface Circuit for the NS32332 7-113
HPC 16040 High-Performance Microcontroller 5-3
ICM-3332 Integrated Computer Module 8-39
ICM-3216 Integrated Computer Module 8-46
INS 8250A Asynchronous Communications Element 4-89
Introduction to Series 32000 Architecture 1-16
ISE16 In-System Emulator 8-17
ISE32 In-System Emulator 8-8
Military/Aerospace Program 1-12
NS405 Series Display/Terminal Management Processor 5-26
NS16450 Asynchronous Communications Element 4-89
NS16550 Asynchronous Communications Element With FIFO's 4-104
NS32008-6, -8, -10 High Performance Microprocessors 2-339
NS32016-6, -8, -10 High Performance Microprocessors 2-275
NS32C016-6, -10, -15 High Performance Microprocessors 2-211

Alpha-Numeric Index (Continued)

NS32032-6, -8, -10 High Performance Microprocessors 2-146
NS32081-6, -8, -10 Floating Point Unit (FPU) 3-3
NS32082-6, -8, -10 Memory Management Unit (MMU) 3-20
NS32132-6, -8, -10 High-Performance Microprocessors 2-76
NS32201-6, -8, -10 Timing Control Unit (TCU) 4-25
NS32C201-6, -10, -15 Timing Control Unit (TCU) 4-3
NS32202-6, -8, -10 Interrupt Control Unit (ICU) 4-64
NS32301-10, -15 Advanced Timing Control Unit (TCU) 4-49
NS32332-10, -12, -15 32-Bit Advanced Microprocessor with Virtual Memory 2-3
NS32382 Memory Management Unit 3-59
NS32405 Series Display/Terminal Management Processor 5-26
NS32440 Serial Bi-Phase Transmitter/Encoder 5-130
NS32441 Serial Bi-Phase Receiver/Decoder 5-139
NS32442 High-Speed Serial Transmitter/Encoder 5-150
NS32443 High-Speed Serial Receiver/Decoder 5-160
NS32490 Network Interface Controller 5-65
NS32491 Serial Network Interface 5-113
NS32492 Coaxial Tranceiver Interface 5-122
NS32800 E^2C^2 Expandable Error Checker/Corrector 7-3
NS32802A 32-Bit Parallel Error Detection and Correction Circuits (EDAC's) 7-37
NS32809A Multi-Mode Dynamic RAM Controller/Driver 7-53
NS32812 Dynamic RAM Controller Interface Series Circuit
for the Series 32000 CPU 7-75
NS32813 Dynamic RAM Controller Interface Circuit for the NS32332 7-113
NS32817 64K, 256K Dynamic RAM Controller/Drivers 7-88
NS32818 64K, 256K Dynamic RAM Controller/Drivers 7-88
NS32819 64K, 256K Dynamic RAM Controller/Drivers 7-88
NS32819X 64K, 256K Dynamic RAM Controller/Drivers 7-88
NS32828 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114
NS32829 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114
NS32951 Data Synchronizer 6-3
NS32955 Data Synchronizer 6-3
NS32961 Data Separator 6-3
NS32962 2, 7 Code Data Synchronizer 6-29
NS32963 2, 7 ENDEC 6-49
NS32964 Disk Pulse Detector 6-58
NS32965 Data Separator 6-3
NS32966 Data Controller 6-81
NS32970 Floppy Disk Support Chip Data Separator and
Write Precompensation 6-133
NS32972 Floppy Disk Controller Plus 6-143
Programming Reference Guide 11-10
Series 32000 EXEC ROMable Real Time Multitasking EXECUTIVE 9-11
Series 32000 Genix 4.2 Operating System 9-5
Series 32000 Genix Native and Cross-Support (GNX) Language Tools 9-7
Series 32000 Quick Reference Timing 11-42
Series 32000 Real-Time Software Component VRTX/IOX/FMX/TRACER 9-15
Series 32000 VERDIX Ada Development System (VADs)
for System V/Series 32000 Native Test 9-19
Series 32000 VERDIX Ada Development System (VADs) for VAX/VMS 9-21
SYS32/20 Development System 8-3
System V/Series 32000 Operating System 9-3
VR32 Target/Development System 8-4

Section 1

Section 1 Contents

Introduction 1-3
Key Features of Series 32000 1-4
Series 32000 Component Descriptions 1-5
Series 32000 Part Numbering Scheme 1-6
Hardware Chart 1-8
Systems and Software Chart 1-9
Support Devices 1-10
Datasheet Description 1-11
Military/Aerospace Program 1-12
Series 32000 Programs and Services 1-15
  Technical Support Engineering Center 1-15
  Special Programs 1-15
  Microcomputer Systems Division 1-15
Introduction to Series 32000 Architecture 1-16
Benefits of Demand Paged Virtual Memory 1-52

Introduction

Series 32000 offers the most complete solution to your 32-bit microprocessor needs via CPUs, slave processors, system peripherals, evaluation/development tools and software.

We at National Semiconductor firmly believe that it takes a total family of microprocessors to effectively meet the needs of a system designer.

This Series 32000 Databook presents technical descriptions of Series 32000 8-, 16- and 32-bit microprocessors, slave processors, peripherals, software and development tools. It is designed to be updated frequently so that our customers can have the latest technical information on the Series 32000.

Series 32000 leads the way in state-of-the-art microprocessor designs because of its advanced architecture, which includes:

- 32-Bit Architecture
- Demand Paged Virtual Memory
- Fast Floating-Point Capability
- High-Level Language Support
- Symmetrical Architecture

When we at National Semiconductor began the design of the Series 32000 microprocessor family, we decided to take a radical departure from popular trends in architectural design that dated back more than a decade. We chose to take the time to design it properly.

Working from the top down, we analyzed the issues and anticipated the computing needs of the 80's and 90's. The result is an advanced and efficient family of microprocessor hardware and software products.

Clearly, software productivity has become a major issue in computer-related product development. In microprocessor-based systems this issue centers around the capability of the microprocessor to maximize the utility of software relative to shorter development cycles, improved software reliability and extended software life cycles.

In short, the degree to which the microprocessor can maximize software utility directly affects the cost of a product, its reliability, and time to market. It also affects future software modification for product enhancement or rapid advances in hardware technology.

Our approach has been to define an architecture addressing these software issues most effectively. Series 32000 combines 32-bit performance with efficient management of large address space. It facilitates high-level language program development and efficient instruction execution. Floating-point is integrated into the architecture.

This combination gives the user large system computing power at two orders of magnitude less cost.

But we didn't stop there. Advanced architecture isn't enough. Our top-down approach includes the hardware, software, and development support products necessary for your design. The evaluation board, in-system emulator, software development tools, including a VAX-11 cross-software package, and third party software are also available now for your evaluation and development.

The Series 32000 is a solid foundation from which National Semiconductor can build solutions for your future designs while satisfying your needs today.

For further information please contact your local sales office.

Key Features of Series 32000®

Some of the features that set the Series 32000 family apart as the best choice for 32-bit designs are as follows:

FAMILY OF MICROPROCESSOR CHIP SETS

Series 32000 is more than just a single chip set, it is a family of chip sets. By mixing and matching Series 32000 CPUs with compatible slave processors and support chips, a system designer has an unprecedented degree of flexibility in matching price/performance to the end product.

CLEANEST 32-BIT SUPER MINI COMPUTER ARCHITECTURE

Series 32000 was designed around a 32-bit architecture from the beginning. It has a fully symmetrical instruction set so that all addressing modes and all data types can be operated on by all instructions. This makes it easy to learn the architecture, easy to program in assembly language, and easy to write code-efficient, high-level language compilers.

DEMAND-PAGED VIRTUAL MEMORY MANAGEMENT

Series 32000 provides hardware support for Demand-Paged Virtual Memory Management. This allows use of low-cost disk storage to increase the apparent size of main memory, and is an efficient method of managing very large address spaces. It is also the same popular memory management method used by DEC and IBM in their minicomputers and mainframes.

APPLICATION-SPECIFIC SLAVE PROCESSORS

Series 32000 architecture allows users to design their own application-specific slave processors to interface with the existing chip set. These processors can be used to increase your overall system performance by accelerating customized CPU instructions that you would otherwise implement in software. At the same time, software compatibility is maintained, i.e., it is always possible to subsitute lower-cost software modules in place of the slave processor.

FLOATING-POINT UNIT

NS32081 Floating-Point Unit provides high-speed arithmetic computation with high precision and accuracy at low cost. The NS32081 supports the entire Series 32000 family of CPUs and complies with the proposed IEEE standard for floating-point arithmetic, Task P754.

OPERATING SYSTEM SUPPORT

Series 32000 features such as hardware support for Demand-Paged Virtual memory management, user software protection and modular programming make it much easier to implement powerful, reliable and efficient operating systems. These features along with its symmetrical architecture and powerful instruction set make the Series 32000 the most efficient and highest performance UNIX engine.

HIGH-LEVEL LANGUAGE SUPPORT

Series 32000 has special features that support high-level languages, thus improving software productivity and reducing development costs. For example, there are special instructions that help the compiler deal with structured data types such as Arrays, Strings, Records, and Stacks. Also, modular programming is supported by special hardware registers, software instructions, an external addressing mode, and architecturally supported link tables.

Series 32000 Component Descriptions

Device	Description	Bus Width			Process	Package Type
		Internal	External			
			Address	Data		
CENTRAL PROCESSING UNITS (CPU's)						
NS32532	Advanced CMOS Central Processing Unit	32	32	32	M²CMOS	TBD
NS32332	Advanced Central Processing Unit	32	32	32	XMOS™ (NMOS)	84-pin PGA
NS32132	Central Processing Unit	32	24	32	XMOS (NMOS)	68-pin LCC Leadless Chip Carrier
NS32C032	CMOS Central Processing Unit	32	24	32	CMOS	68-pin LCC Leadless Chip Carrier
NS32032	Central Processing Unit	32	24	32	XMOS (NMOS)	68-pin LCC Leadless Chip Carrier
NS32C016	CMOS Central Processing Unit	32	24	16	CMOS	48-pin DIP Dual-In-Line Package
NS32016	Central Processing Unit	32	24	16	XMOS (NMOS)	48-pin DIP Dual-In-Line Package
NS32008	Central Processing Unit	32	24	8	XMOS (NMOS)	48-pin DIP Dual-In-Line Package
SLAVE PROCESSORS						
NS32382	Advanced Memory Management Unit	32	32	32	XMOS (NMOS)	PGA
NS32082	Memory Management Unit	32	24	16	XMOS (NMOS)	48-pin DIP Dual-In-Line Package
NS32310	Intelligent Floating Point Controller	64	—	32	M²CMOS	PGA
NS32081	Floating Point Unit	64	—	16	XMOS (NMOS)	24-pin DIP Dual-In-Line Package
PERIPHERALS						
NS32301	Advanced Timing Control Unit	—	—	—	Bipolar	28-pin DIP
NS32C201	CMOS Timing Control Unit	—	—	—	CMOS	24-pin DIP Dual-In-Line Package
NS32201	Timing Control Unit	—	—	—	Bipolar	24-pin DIP Dual-In-Line Package
NS32202	Interrupt Control Unit	32	—	16	XMOS (NMOS)	40-pin DIP Dual-In-Line Package
NS32203	Direct Memory Access Controller	—	—	16	XMOS (NMOS)	48-pin DIP Dual-In-Line Package

Series 32000 Part Numbering Scheme

Over the past few years, National's 32-bit Microprocessor Family has come a long way. The product has met with unprecedented acceptance in the marketplace—and is well on its way to being the 32-bit industry standard.

To highlight the completeness of Series 32000, all related products have a 4-character 'Series' prefix which will cause them to sort together in the following sequence in published material such as the Price Schedules.

Prefix	Product Type
NSP-	Technical Publications
NSR-	Service
NSS-	Development Systems
NSV-	Evaluation Tools
NSW-	Software
NS32	Components

This scheme applies to order/part numbers only. It should be noted that certain products may, in addition to their unique order/part number, also have a *marketing name*. For example, we expect you will find it more comfortable to refer to the Development System as "VR32" rather than VR32-1001!

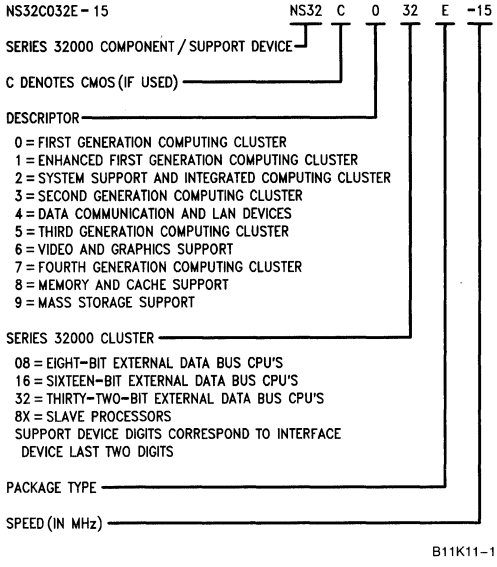
Following the 4-character prefix, the remaining 11 characters specify the product in as intelligible a fashion as possible.

We have included in the Series 32000 family of microprocessors a number of products that a designer most frequently requires to create a state-of-the-art system.

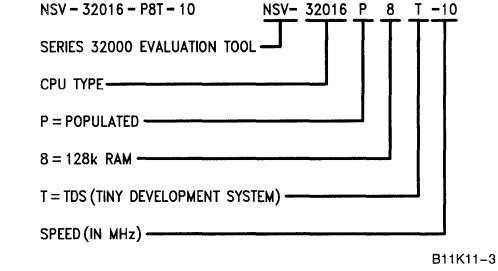
Among these support devices are Data Communications and Local Area Network IC's as well as Disk Control and Interface and DRAM Interface devices.

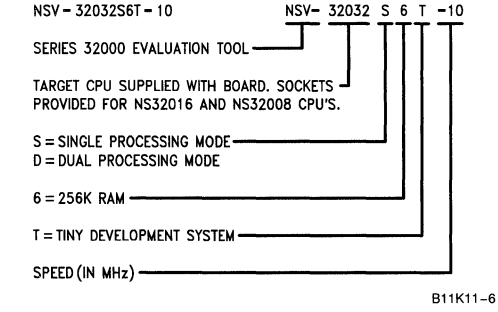
These support components are numbered in Series 32000 fashion and are explained on the following page. Take as an example the NS32965. The NS32 describes a Series 32000 component. The 9 signifies a Data Communication/LAN device and the 65 are the last two digits in the equivalent National Semiconductor Interface device.

Components

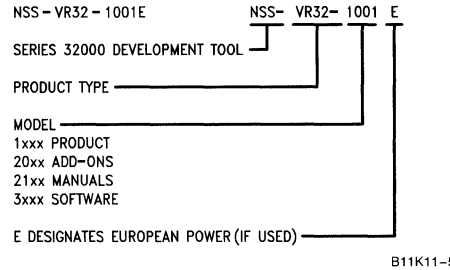


Evaluation Tools

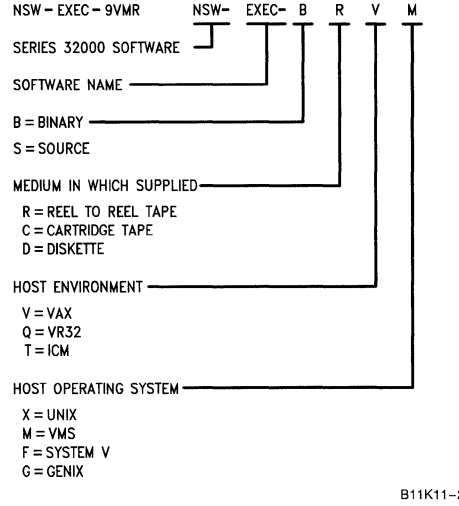




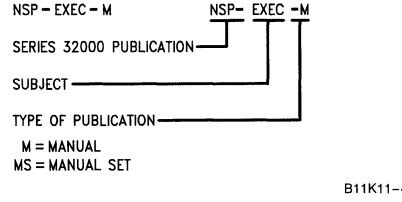
Development Tools



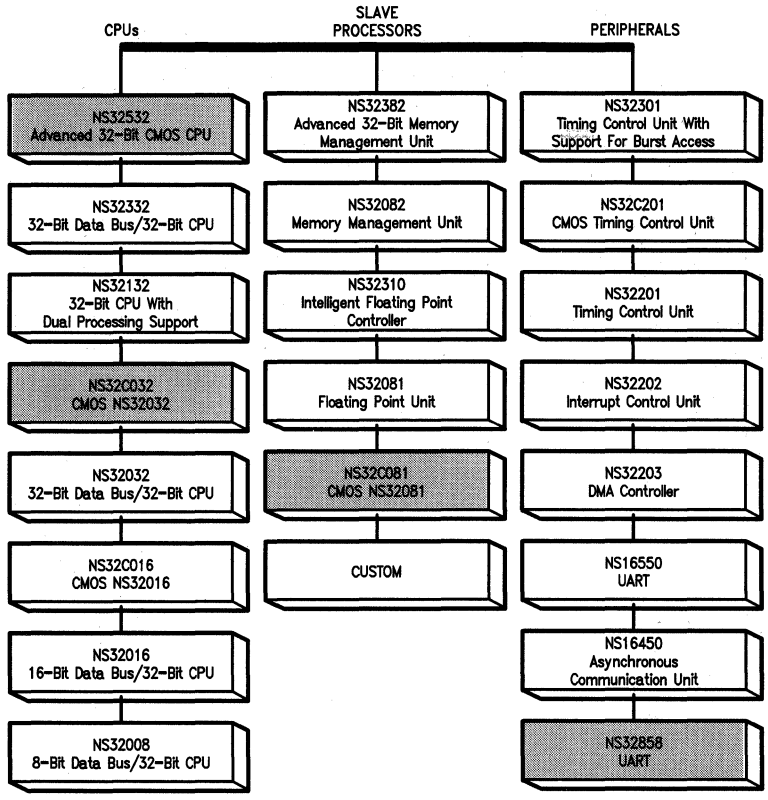
Software



Publications



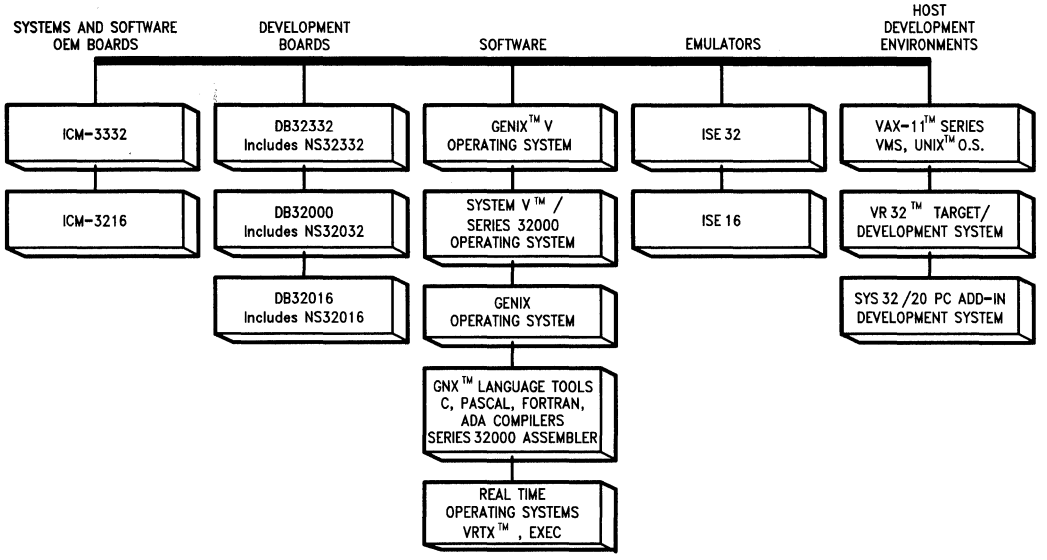
Hardware Chart



1K12-1

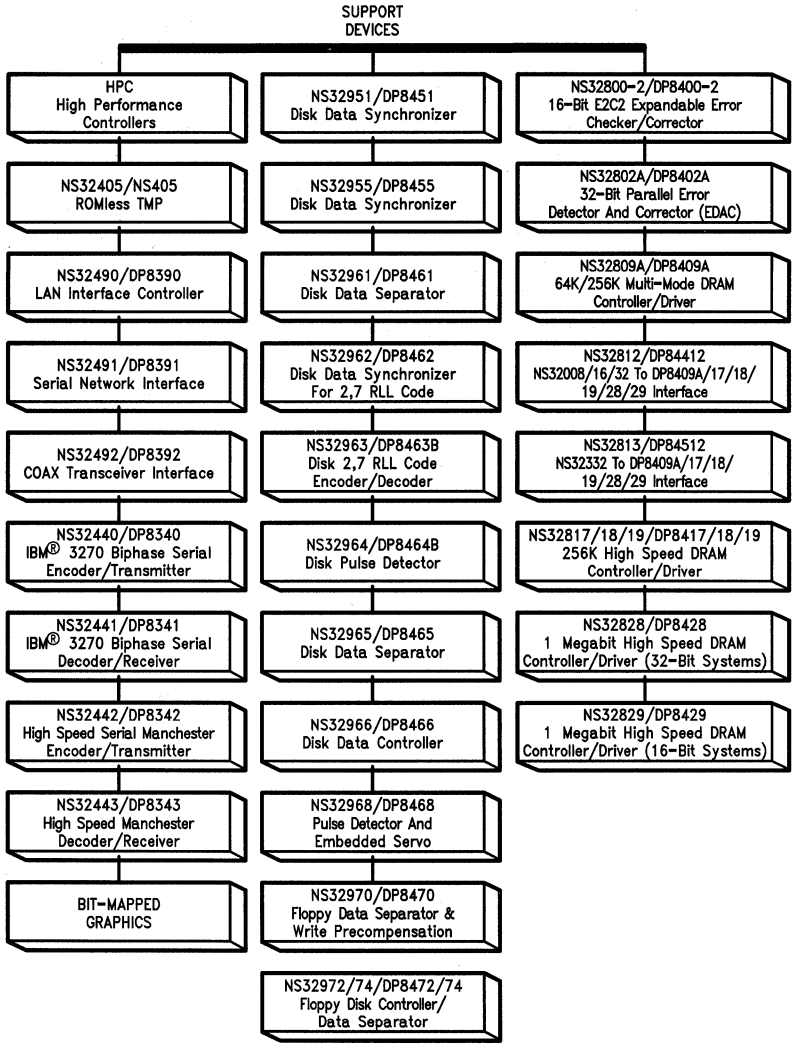
Note: Products in the shaded boxes are additional hardware components planned to support the Series 32000 CPUs. Please contact your local National Sales Office for further information on their availability.

Systems and Software Chart



1K13-1

Support Devices Chart



1K12-2

Data Sheets/Description

Series 32000® information is grouped into one of three categories depending on the type of information presented. These categories are:

Advanced Information — This is the first official information released about a future Series 32000 device. It contains very basic information about a product and usually precedes sample devices by approximately six months. This type of data sheet is distinguished by the words "Advanced Information" appearing in the header of the first page.

Preliminary — This document contains an extensive discussion of device operation and provides complete parametric information such as Maximum Ratings, Thermal Characteristics, Electrical Characteristics, Bus Timing, and I/O Port Timing as applicable. Timing diagrams are included to support the tabular material. All of the parametric information given is the result of early testing of initial product from the manufacturing process. Values given are subject to change without notice. This type of data sheet is distinguished by the words "Preliminary" appearing in the header of the first page.

Final Data Sheet — This data sheet evolves from the Preliminary data sheet. It is a result of test information collected from a fully-implemented manufacturing process. The parametric information has been analyzed and approved. National Semiconductor considers this a fully characterized device. This type of data sheet is distinguished by the absence of any designation appearing in the header of the first page.

Military/Aerospace Programs from National Semiconductor

This section is intended to provide a brief overview of military products available from National Semiconductor. For further information, refer to our 1986 Reliability Handbook which is expected to be available by mid 1986.

MIL-M-38510

The MIL-M-38510 Program, which is sometimes called the JAN IC Program, is administered by the Defense Electronics Supply Center (DESC). The purpose of this program is to provide the military community with standardized products that have been manufactured and screened to government-controlled specifications in government certified facilities. All 38510 manufacturers must be formally qualified and their products listed on DESC's Qualified Products List (QPL) before devices can be marked and shipped as JAN products.

There are two processing levels specified within MIL-M-38510: Classes S and B. Class S is typically specified for space flight applications, while Class B is used for aircraft and ground systems. National is a major supplier of both classes of devices. Screening requirements are outlined in Table III.

Tables I and II explain the JAN device marking system.

Copies of MIL-M-38510, the QPL, and other related documents may be obtained from:

Naval Publications and Forms Center
5801 Tabor Avenue
Philadelphia, PA 19120
(212) 697-2179

DESC Specifications

DESC specifications are issued to provide standardized versions of devices which are not yet available as JAN product. MIL-STD-883 Class B screening is coupled with tightly controlled electrical specifications which have been written to allow a manufacturer to use his standard electrical tests. A current listing of National's DESC specification offerings can be obtained from our franchised distributors, sales representatives, or DESC. DESC is located in Dayton, Ohio.

MIL-STD-883

Although originally intended to establish uniform test methods and procedures, MIL-STD-883 has also become the general specification for non-JAN military product. Revision C of this document defines minimum requirements for a device to be marked and advertised as 883-compliant. Included are design and construction criteria, documentation controls, electrical and mechanical screening requirements, and quality control procedures. Details can be found in paragraph 1.2.1 of MIL-STD-883.

National offers both 883 Class B and 883 Class S product. The screening requirements for both classes of product are outlined in Table III.

As with DESC specifications, a manufacturer is allowed to use his standard electrical tests provided that all critical parameters are tested. Also, the electrical test parameters, test conditions, test limits, and test temperatures must be clearly documented. At National Semiconductor, this information is available via our RETS (Reliability Electrical Test Specification) program. The RETS document is a complete description of the electrical tests performed and is controlled by our QA department. Individual copies are available upon request.

Some of National's older products are not completely compliant with MIL-STD-883, but are still required for use in military systems. These devices are screened to the same stringent requirements as 883 product but are marked "-Mil".

Military Screening Program (MSP)

National's Military Screening Program was developed to make screened versions of advanced products such as gate arrays and microprocessors available more quickly than is possible for JAN and 883 devices. Through this program, screened product is made available for prototypes and brassboards prior to or during the JAN or 883 qualification activities. MSP products receive the 100% screening of Table III, but are not subjected to group C and D quality conformance testing. Other criteria such as electrical testing and temperature range will vary depending upon individual device status and capability.

TABLE I. The MIL-M-38510 Part Marking

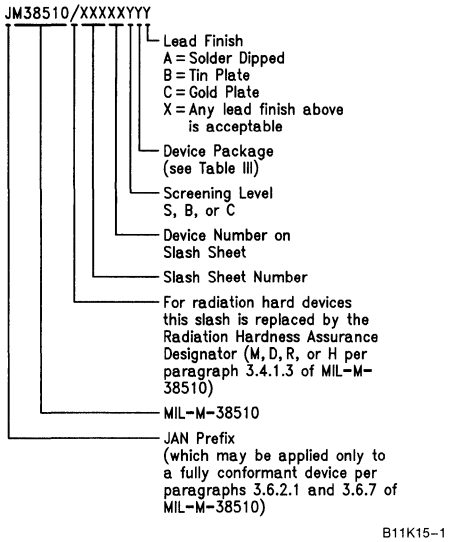


TABLE II. JAN Package Codes

38510 Package Designation	Microcircuit Industry Description
A	14-pin 1/4″ x 1/4″ (metal) flat pack
B	14-pin 3/16″ x 1/4″ flat pack
C	14-pin 1/4″ x 3/4″ dual-in-line
D	14-pin 1/4″ x 3/8″ (ceramic) flat pack
E	16-pin 1/4″ x 7/8″ dual-in-line
F	16-pin 1/4″ x 3/8″ (metal or ceramic) flat pack
G	8-pin TO-99 can or header
H	10-pin 1/4″ x 1/4″ (metal) flat pack
I	10-pin TO-100 can or header
J	24-pin 1/2″ x 1-1/4″ dual-in-line
K	24-pin 3/8″ x 5/8″ flat pack
L	24-pin 1/4″ x 1-1/4″ dual-in-line
M	12-pin TO-101 can or header
N	(Note 1)
P	8-pin 1/4″ x 3/8″ dual-in-line
Q	40-pin 3/16″ x 2-1/16″ dual-in-line
R	20-pin 1/4″ x 1-1/16″ dual-in-line
S	20-pin 1/4″ x 1/2″ flat pack
T	(Note 1)
U	(Note 1)
V	18-pin 3/8″ x 15/16″ dual-in-line
W	22-pin 3/8″ x 1-1/8″ dual-in-line
X	(Note 1)
Y	(Note 1)
Z	(Note 1)
2	20-terminal 0.350″ x 0.350″ chip carrier
3	28-terminal 0.450″ x 0.450″ chip carrier

Note 1: These letters are assigned to packages by individual detail specifications and may be assigned to different packages in different specifications.

TABLE III. 100% Screening Requirements

Screen	Class S		Class B	
	Method	Reqmt	Method	Reqmt
1. Wafer Lot Acceptance	5007	All Lots		—
2. Nondestructive Bond Pull	2023	100%		—
3. Internal Visual (Note 1)	2010, Condition A	100%	2010, Condition B	100%
4. Stabilization Bake	1008, Condition C, Min, 24 Hrs. Min	100%	1008, Condition C, Min, 24 Hrs. Min	100%
5. Temp. Cycling (Note 2)	1010, Condition C	100%	1010, Condition C	100%
6. Constant Acceleration	2001, Condition E (Min) Y_1 Orientation Only	100%	2001, Condition E (Min) Y_1 Orientation Only	100%
7. Visual Inspection (Note 3)		100%		100%
8. Particle Impact Noise Detection (PIND)	2020, Condition A (Note 4)	100%		—
9. Serialization	(Note 5)	100%		—
10. Interim (Pre-Burn-In) Electrical Parameters	Per Applicable Device Specification (Note 13)	100%	Per Applicable Device Specification (Note 6)	—
11. Burn-In Test	1015 240 Hrs. at 125°C Min (Cond. F Not Allowed)	100%	1015, 160 Hrs. at 125°C Min	100%

TABLE III. 100% Screening Requirements (Continued)

Screen	Class S		Class B	
	Method	Reqmt	Method	Reqmt
12. Interim (Post-Burn-In) Electrical Parameters	Per Applicable Device Specification (Note 13)	100%		—
13. Reverse Bias Burn-In (Note 7)	1015; Test Condition A, C, 72 Hrs. at 150°C Min (Cond. F Not Allowed)	100%		—
14. Interim (Post-Burn-In) Electrical Parameters	Per Applicable Device Specification (Note 13)	100%	Per Applicable Device Specification	100%
15. PDA Calculation	5% Parametric (Note 14) 3% Functional — 25°C	All Lots	5% Parametric (Note 14)	All Lots
16. Final Electrical Test a) Static Tests 1) 25°C (Subgroup 1, Table I, 5005) 2) Max & Min Rated Operating Temp (Subgroups 2, 3, Table I, 5005) b) Dynamic Tests & Switching Tests, 25°C (Subgroups 4, 9, Table I, 5005) c) Functional Test, 25°C (Subgroup 7, Table I, 5005)	Per Applicable Device Specification	 100% 100% 100% 100%	Per Applicable Device Specification	 100% 100% 100% 100%
17. Seal Fine, Gross	1014	100% (Note 8)	1014	100% (Note 9)
18. Radiographic (Note 10)	2012 Two Views	100%		—
19. Qualification or Quality Conformance Inspection Test Sample Selection	(Note 11)	Samp.	(Note 11)	Samp.
20. External Visual (Note 12)	2009	100%		100%

Note 1: Unless otherwise specified, at the manufacturer's option, test samples for Group B, bond strength (Method 5005) may be randomly selected prior to or following internal visual (Method 5004), prior to sealing provided all other specification requirements are satisfied (e.g. bond strength requirements shall apply to each inspection lot, bond failures shall be counted even if the bond would have failed internal visual).

Note 2: For Class B devices, this test may be replaced with thermal shock method 1011, test condition A, minimum.

Note 3: At the manufacturer's option, visual inspection for castastrophic failures may be conducted after each of the thermal/mechanical screens, after the sequence or after seal test. Catastrophic failures are defined as missing leads, broken packages or lids off.

Note 4: The PIND test may be performed in any sequence after step 9 and prior to step 16. See MIL-M-38510, paragraph 4.6.3.

Note 5: Class S devices shall be serialized prior to interim electrical parameter measurements.

Note 6: When specified, all devices shall be tested for those parameters requiring delta calculations.

Note 7: Reverse bias burn-in is a requirement only when specified in the applicable device specification. The order of performing burn-in and reverse bias burn-in may be inverted.

Note 8: For Class S devices, the seal test may be performed in any sequence between step 16 and step 19, but it shall be performed after all shearing and forming operations on the terminals.

Note 9: For Class B devices, the fine and gross seal tests shall be performed separate or together in any sequence and order between step 6 and step 20 except that they shall be performed after all shearing and forming operations on the terminals. When 100% seal screen cannot be performed after shearing and forming (e.g. flatpacks and chip carriers) the seal screen shall be done 100% prior to those operations and a sample test (LTPD = 5) shall be performed on each inspection lot following these operations. If the sample fails, 100% rescreening shall be required.

Note 10: The radiographic screen may be performed in any sequence after step 9.

Note 11: Samples shall be selected for testing in accordance with the specific device class and lot requirements of Method 5005.

Note 12: External visual shall be performed on the lot any time after step 19 and prior to shipment.

Note 13: Read and Record when post burn-in data measurements are specified.

Note 14: PDA shall apply to all static, dynamic, functional and switching measurements at either 25°C or maximum rated operating temperature.

Series 32000 Programs and Services

Technical Support Engineering Center (TSEC)

SERVICE CENTER

NSC offers a full 90 day warranty period on each Development Systems product that it sells. This warranty can be enhanced, by purchasing at the time of sale, an added one year warranty which equates out to a fifteen month period coverage. Contact MCS Logistics at the toll-free numbers listed below for more information.

The Technical Support Engineering Center has highly trained technical specialists available to assist customers over the telephone with any Development System problems. The technical specialists utilize SPIRE, a computerized technical data base designed for rapid search, to solve customer and technical problems. This data-base can be made available for customer use and communications to the technical staff. Contact the SPIRE administrator at the numbers below for more information.

Depot repair services are available for board and system products. Our customers can use the toll-free numbers to contact the service center for immediate solutions.

(800) 538-1866,
(800) 672-1811 for California
(800) 223-3248 for Canada.

When indicated other features of MCS service are used. These include a service problem report (SPR) that modifies a customer profile database, a request for engineering action (REA) report that aids in product improvement, and an escalation procedure that is used when necessary to involve applications and design engineering to help bring any problem to a rapid resolve.

National's field engineers are located in Santa Clara, Canada and Europe and are available for dispatch to customer sites to repair our Development Systems products. Extensive spare parts inventories are maintained for such use.

Special Programs

Series 32000 Consultant Program

The Series 32000 Consultant Program was developed to create a network of consulting firms throughout the United States which act as independent agents for National Semiconductor's Series 32000 Family. These agents are available to help companies design in Series 32000 products. NSC provides a referral listing of all certified agents and their area of expertise.

Series 32000 User Society

The charter of the Series 32000 User Society is to advance the effective utilization of National's microprocessors. The Society promotes the exchange of information and ideas between Series 32000 software and hardware users. The Society newsletter, which discusses design innovations and new applications for the Series 32000 family, facilitates the exchange of microprocessing information among Series 32000 users.

The University Program

Begun as merely a concept several years ago, National Semiconductor's University Program has now emerged as one of the company's most successful programs. The University Program was originally created to establish a relationship between National and the academic community that would foster the exchange of information and keep students abreast of modern advancements in technology.

Today, the University Program provides a wide variety of services to universities such as university product kits, equipment loans, student research aid and on-campus product demonstrations. Although probably best known for its Series 32000 product kit, the University Program now offers equipment from *all* departments within National, at substantial savings.

The University Program catalog provides a complete, up-to-date list of all student/university services as well as program application forms and course materials to guide instructors in introducing students to advanced microprocessors.

Because tomorrow's technology is dependent upon today's nurturing of up-and-coming scientists and engineers, National is committed to supporting universities, particularly in the area of microprocessor technology. National hopes that more universities will share in this commitment by becoming a part of the University Program.

For more information on any of these programs, contact Linda Price, Program Manager, National Semiconductor Corporation, 2900 Semiconductor Drive, M/S 7C-261, Santa Clara, California 95052-8090, 408-733-2600 ext. 463.

Microcomputer Systems Division

The Microcomputer Systems Division's goal is to become a leading force in the microcomputer systems marketplace.

To achieve this goal, a total systems approach has been taken on the Series 32000 program to provide the customer with the necessary hardware and software support, evaluation and development tools, training, service and technical literature.

The focus is on upward migration paths, system integration at all levels and the preservation of the user's software investment.

Four groups (Microprocessor, OEM Board Level Products, Software Products and Development Systems) offer a broad capability to solve customer needs at various levels of performance and integration.

Introduction to Series 32000 Architecture 3rd Edition

Chapter 1 Computer Architecture

1.1 Introduction

The architecture of a computer describes what that computer looks like to people who write software for it. More precisely, the architecture is the complete and detailed specification of the interface between the computer and software. The architecture specifies those elementary instructions that are decoded and executed directly by the machine. But it is important to keep in mind that architecture describes only *what* the computer does, not *how* it does it. Two machines are said to have the same architecture if all the software written for one can execute on the other, even if the actual hardware construction of the two machines is entirely different. For example, the members of the IBM System 360–370 family all have basically the same architecture, but the technology used to implement that architecture ranges from discrete transistors to Very Large Scale Integration (VLSI).

Occasionally, the term architecture is used in a more general sense as the boundary between different levels of the whole system. (For example, terms such as "operating system architecture" are occasionally employed.) In this document we will use "architecture" exclusively for the boundary between the actual machine hardware and the software.

1.1.1 The Role of the Computer Architect

A computer architect is someone who designs computer architectures. The terms architecture and architect obviously have been adapted from their ordinary use in the building construction industry. The words are apt because in many ways the job of a computer architect is similar to that of an ordinary architect. Both are more concerned with the overall design of a structure and its appearance to users than to the exact details of the construction, which is the province of the structural engineer or general contractor in the buiiding industry and the hardware designer in the computer industry.

The relationship between computer architect and computer implementor is analogous to the relationship between an architect and a general contractor. The architect designs the overall appearance of the building, balancing a number of conflicting goals (e.g., the desirable view provided by many large windows and the equally desirable goal of energy efficiency), always keeping in mind what is possible with current construction technology (the availability and cost of materials). The general contractor is responsible for translating the architect's vision into a building. If the contractor discovers that some detail of the building's architecture will be too difficult or too expensive to build, or that it will lead to an unsafe structure, the architect may have to make changes.

Similarly, the computer architect designs the external appearance (to software) of the computer, balancing a number of conflicting goals (e.g., complete protection vs. simplicity of use), and always keeping in mind the current state of semiconductor technology. The computer implementor translates this design into silicon. If the implementor finds that some feature of the computer architecture is too difficult or too expensive to implement, or if another feature causes the computer to run significantly slower, the computer architect may have to make changes.

The role of the architect in both industries is to make an intelligent compromise among a number of desirable goals and to balance this against the limitations of current technology to get a cost-effective design. Architectural mistakes usually result when one goal is single-mindedly pursued to the exclusion of other goals, or when a desired goal is simply not technologically feasible.

A certain amount of controversy currently surrounds a number of issues associated with computer architecture. As defined, computer architecture is just the boundary between the hardware and software. The controversy is fundamentally over where that boundary should be drawn, and what trade-offs should be made between various features for reasons of performance. Discussion has centered around three main topics:

- What is the best way to support high level languages?
- How should memory be organized?
- What protection features should be provided by the hardware?

In the remaining sections of this chapter we will examine these three topics, introduce some of the points at issue, and present the Series 32000 approach to each topic.

1.2 High Level Language Support

All evidence suggests that programming in a high level language (e.g., Pascal) is more productive than programming in assembly language. Some researchers have found that high level language programmers can produce the same number of debugged lines of code per day as can assembly language programmers. Since a line of code in a high level language usually performs a more complex operation than a line of code in assembly language, the high level programmer is more productive.

Studies have shown that both the time to debug a program and the difficulty in understanding and maintaining it are proportional to the number of instructions, with little dependency on the complexity of each instruction. Since several instructions might be required for each high level language statement, the savings in programming time and cost over an equivalent assembly language program are obvious.

Before the advent of Series 32000, however, these advantages had been partially offset by the inherent inefficiency of high level languages as opposed to assembly language programs. Depending on the compiler, the computer, and the application, a compiled program might be anywhere from 0% to 300% longer and slower than the best assembly language program. The basic reason for the inherent inefficiency of high level languages (we will call the HLLs, occasionally) when they are targeted to contemporary architectures is that these architectures were not designed to support compilers.

1.2.1 Deficiencies of Current Architectures

The shortcomings of current computer architectures are largely attributable to what Glenford Myers has called the *semantic gap*,[1] a measure of the difference between the concepts in high level languages and the concepts in the computer architecture. The objects and operations reflected in these architectures are seldom closely related to the objects and operations provided in the programming languages. This semantic gap contributes to software unreliability, performance problems, excessive program size, compiler complexity, and distortions of the language.

Here are some of the heavily used concepts in high level languages, along with a few comments on the architectural support for these concepts provided by most computer architectures:

Arrays. The array is one of the most frequently used data structures in most HLLs. An array is a set of entries, each with the same data type (thus we speak of arrays of integers, arrays of characters, etc.). Most languages provide for multidimensional arrays, performing operations on entire arrays and checking to see that array subscripts do not exceed the boundary of the array. Most computer architectures, however, provide very limited architectural features to support any of these constructions.

Records. A record consists of a number of components (usually called fields) that can be of different data types. Thus a record might consist of characters, integers, and real numbers (for instance, a criminal record). There is nothing in the architecture of most microprocessors that supports records.

Strings. Most languages contain the concepts of fixed and variable sized strings, and of string processing operations such as concatenation and searching for a specified substring within a string. Many microprocessor architectures provide no string processing instructions at all.

Procedures. The basic program unit in modern HLLs is the procedure. A procedure call entails saving the state of the calling procedure, dynamically allocating and initializing local storage for the called procedure, passing arguments, and executing the called procedure. Most microprocessor architectures provide no support for any of these operations.

Modules. Modern HLLs (Pascal, Ada) implement the concept of a software module containing several procedures and associated data. Each module may be developed independently of all other modules and combined for final execution. This modularization reduces software development cost and time, increases design flexibility, and simplifies system design. Up to now most processors have not supported the modular software concept.

One source of current problems is that contemporary architectures are *asymmetric*, and therefore do not permit the concepts in HLLs to be efficiently modeled in machine language. Symmetry is the degree to which all addressing modes exist for all operands and all required operators exist for every data type. Chapter 2 discusses symmetry in detail and also defines the key terms, such as *addressing mode* and *data type*.

1.2.2 The Series 32000 Approach

These deficiencies in contemporary microprocessor architectures have been addressed by the designers of Series 32000. They have made a major effort to bridge the semantic gap with this new architecture. Series 32000 architecture, in fact, is designed specifically to support high level language compilers; it enables even relatively unsophisticated compilers to produce efficient code. Special addressing modes are provided to access such HLL constructions as arrays and records, and new operators are provided that are specifically tailored for high level languages.

Addressing Modes. Series 32000 architecture supports four standard *addressing modes* (i.e., mechanisms for accessing operands) common to most processors: register, immediate, absolute, and register relative. In addition, Series 32000 introduces four HLL-oriented addressing modes: top-of-stack mode is very useful for evaluating arithmetic expressions in high level languages. Scaled indexing mode can be used to access elements in byte, word, double-word, or quad-word arrays. Memory relative mode can be used for manipulating fields in a record. External mode can be used to access data in separately compiled modules. (See Chapter 2 for a discussion of addressing modes.)

New Operators. In addition to the conventional CPU instructions, such as data movement, arithmetic logic, and shifts, the architecture includes advanced instructions which are very useful in an HLL environment. The CHECK instruction determines whether an array index is within bounds. The INDEX instruction implements the recursive indexing step for multi-dimensional arrays. The STRING instruction manipulates data strings. ENTER and EXIT instructions minimize the overhead in procedure calls by managing the resources (registers, stack frame) allocated at the beginning of a procedure and reclaimed at the end. (See Chapter 2 for more on these instructions.)

[1] Glenford J. Myers, *Advances in Computer Architecture*, Wiley 1978

1.2.3 Controversial Topics

The addressing modes and new operators provided by Series 32000 clearly represent an advance over contemporary architectures. Yet two of the issues faced by Series 32000 designers remain controversial.

- Should three operand instructions be provided?
- Should instructions be primarily register-oriented, memory-to-memory, or top-of-stack?

Three Operand Instructions

It is occasionally claimed that an architecture must provide general three operand instructions if it is to truly support a HLL. (A three operand instruction is, as the name implies, an instruction which contains two source operands as well as a destination. For example, an instruction to directly implement the FORTRAN statement,

A = B + C

would be a three operand instruction with operands A, B and C and the operator +). The reasoning behind this claim is basically that if three operand statements are common in high level languages, then presence of three operand instructions in the architecture will result in greater code density. The VAX-11, for example, permits three operand instructions for most arithmetic operations.

However, a study by D.E. Knuth of Stanford University in 1971[2] showed that in 250,000 lines of FORTRAN code, 80% of all assignments were of the form

A *op* B or A = B

It follows that three operand HLL statements are extremely rare, and the need for such a construction in the architecture is unproven. Moreover, since provision for three operand instructions imposes a certain burden of its own (whether in code density or execution speed), the utility of this instruction category must certainly be questioned. The designers of Series 32000 felt that the need for three operand instructions was not great enough to justify that overhead. In fact, Series 32000 provides greater code density than the VAX-11.

Registers

It is also occasionally claimed (for example by Glenford Myers in his book *Advances in Computer Architecture*[3]) that registers are alien to the concepts in HLLs and should be done away with in the interests of bridging the semantic gap. The designers of Series 32000 disagree. The high level language concept that relates most strongly to registers is the idea of the set of variables that are local to a procedure. The modular programming methodology described above encourages the use of a number of small procedures instead of large monolithic programs. Each of these procedures usually makes use of only a few variables of its own, but these variables are used over and over again in that procedure. For instance, a procedure that manipulates an array must constantly refer to the array index.

The chief advantage of registers is that they allow a working set of variables to be kept close at hand where they can be accessed quickly. This working set of variables is stored in the register set. Studies by William Wulf, *et al.*[4] have indicated that five registers are sufficient for almost all applications. Series 32000 CPUs use 8 (i.e., 2^3) general-purpose registers and several specialized registers for particular pointers. Series 32000 architecture *allows* memory-to-memory operations, but it does not *require* them.

Registers allow the compiler writer to optimize the execution of HLL statements, whereas a purely memory-to-memory machine must constantly carry the overhead of referencing all variables in main memory. A purely stack-oriented machine (i.e., an architecture where all variables are assumed to be on the top two locations of the stack) is essentially equivalent to a machine with two registers. Many studies have shown that pure stack machines do not give any significant advantage over a general register machine.[5]

1.3 Memory Organization

There are three aspects to memory organization: (1) the overall memory architecture, which is basically how the logical memory looks to the computer program; (2) logical-to-physical address translation (mapping) which maps the logical structure of memory onto hardware; and (3) virtual memory mechanisms. Series 32000 has a *linear* memory architecture; it supports *page-based* mapping; and it provides a number of mechanisms which support a virtual memory system.

1.3.1 Linear versus Segmented Memory Architecture

The main memory of a computer is organized as a set of consecutively numbered storage cells. In most computers these memory cells contain eight bits (a *byte*). The location number associated with one of these physical storage cells is called a *physical address*, and the set of all physical addresses is called *physical address space*. The physical address space is thus determined by the actual hardware in the computer's memory system.

On the other hand, a program running on a computer can generate a set of addresses that is limited only by the number of bits in an address. This set of addresses is not necessarily related to the actual amount of physical memory in the system. For example, consider a computer with a 16-bit address field in instructions and 4,096 (4K) bytes of memory. A program on this computer can address 65,536 (64K) locations, for the simple reason that 2^{16} (65,536) 16-bit numbers exist. The set of these numbers is called *logical address space*; it is the set of *logically possible addresses* (even if they are not realized physically); it is the set of all addresses that can be generated by a program. The organization of the logical address space defines the *memory architecture*. The two main types of memory architecture are *linear* and *segmented*.

---

[2] D. E. Knuth, "An Empirical Study of FORTRAN Programs," *Software Practice and Experience*, 1, 2 (April–June, 1971) 105-133.

[3] Meyers, *op. cit.*, p. 23

VAX-11 is a trademark of Digital Equipment Corporation.

[4] W. A. Wulf, *et al.*, *The Design of an Optimizing Compiler*, North Holland, 1975

[5] Meyers, *op. cit.*, p. 49

In a linear address space, addresses start at location zero and proceed in linear fashion (i.e., with no holes or breaks) to the upper limit imposed by the total number of bits in a logical address. With Series 32000, there can be up to 24 bits in a logical address, resulting in 16 million (2^{24}) bytes. In fact, Series 32000 architecture makes provision for 32-bit logical addresses, allowing 4 billion (2^{32}) bytes of logical memory to be addressed.

The alternative to a linear memory architecture is a so-called "segmented" memory architecture. A segmented address space is basically a collection of small linear address spaces. A rigid distinction is made between the *segment* (the particular address space in which a datum is located) and the *displacement* of the datum within the segment (the distance in bytes from the start of the segment to the location in question). A segmented address is consequently a two-component value. The first component (the segment selector) picks out one of the segments while the second component specifies the displacement within the segment. (See Figure 1-1 for a comparison of linear and segmented memory.)

The advantages of segmented memory center around protection issues. The claim is made that a segmented memory better accords with the organization of modern, modular programs and structured data than does a linear memory. Consequently, mechanisms for preventing access to segments, or preventing segments from being read or written into can be used to protect meaningful program units. In other words, since the structure of the logical address space of a segmented architecture reflects the logical structure of the program, protection mechanisms provided for segments naturally accrue to meaningful program units.

This is in fact true. However, except for a few processors (e.g., the MULTICS processor) few segmented machines have consistently carried out this program. For example, most current segmented architectures impose a limit of 64K bytes on the length of a segment. But in order for segmentation to realize its protection advantages, segments should be allowed to have arbitrary size. A 2-megabyte segment, after all, will be needed to hold a 2-megabyte array, if the program organization is to reflect the program structure. And in modern bit mapped graphics systems (a typical application for 16-bit microcomputers), 2-megabyte arrays are common. Moreover, since programs can consist of hundreds or even thousands of modules, it is important for the architecture to support large numbers of segments if segmentation is to be used properly. Large data bases are a typical application that will require either segments of arbitrary size or a great many segments.

It is unfortunately the case that most segmented architectures allow only small segments (i.e., less than 64K bytes) and usually support only a limited number of them (typically, fewer than 128). The size limitation is an artifact of earlier days when the entire (linear) address space was only 64K bytes long. The designers of the segmented machines expanded the address space of their earlier processors, while attempting to preserve some measure of software compatibility, by making the old 64K-byte linear address space one of the new 64K bytes segments. The 8086 and its relationship to the 8080 is the most painful illustration of this phenomenon.

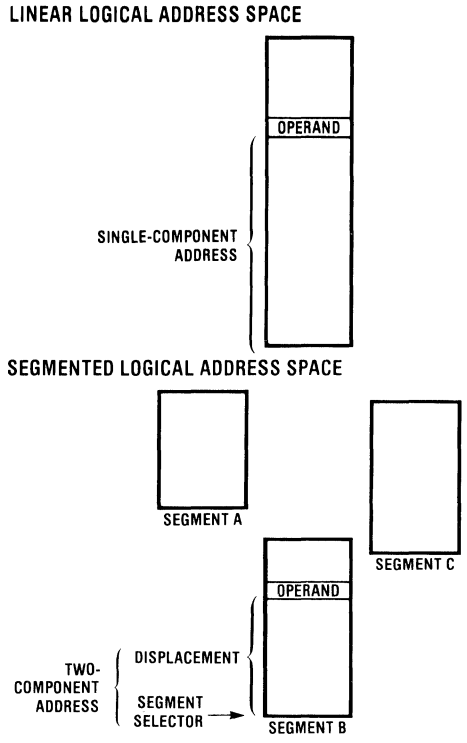


Figure 1-1. Linear vs. Segmented Address Space

In such segmented architectures, all data structures larger than the maximum segment size must be broken down to fit into several segments, since an address pointer cannot be incremented from the top of one segment to the bottom of another segment. By contrast, a linear address space can accommodate data structures of any size up to the maximum size of memory.

Series 32000 provides the protection advantages of segmentation without the segment-size disadvantages, by permitting segments to be constructed out of an arbitrary number of fixed-size memory units. These memory units are called *pages*, and they form the basis for Series 32000 mapping, virtual memory, and memory protection mechanisms (see Sections 1.3.2, 1.3.3, and 1.4).

Series 32000 *permits* a form of segmentation—that is, it lets the operating system keep track of collections of pages with the same protection attributes—but it does not *require* segmentation by building it into the architecture. Moreover, the segmentation permitted by Series 32000 is more general than that built into standard segmented architectures (for example, segments can have arbitrary size).

1.3.2 Page-Based Mapping and Alternatives

Mapping is based on the distinction outlined in Section 1.3.1 between logical address space and physical address space. Mapping is basically the process of translating a logical address into an arbitrary physical address. Without mapping, logical addresses are simply equated with physical addresses; by exploiting mapping, a logical address can be assigned to an arbitrary physical address. Mapping thus provides a kind of generalized relocation mechanism.

Unmapped memory is adequate for simple, single user, single-task systems, which is why most microcomputer applications until now have been unmapped. However, the large memory and increased power of 16/32-bit microcomputers have led to their being employed in multi-user, multi-task applications. And in these cases mapping is highly desirable, for without mapping, the different programs in a multiprogramming system or the different tasks in a multitasking system must operate within the same logical address space. Consequently, each program or task must be careful not to access any address outside its assigned partition, and in general everyone must be familiar with the detailed organization of memory in order to make full use of it.

By contrast, mapping allows each program or task to be assigned its own logical address space, with the mapping mechanism responsible for translating these independent logical address spaces into the same physical address space. Since the programs and tasks have separate logical address spaces, there is no chance of interference.

Since it is too cumbersome to control the translation of each logical address individually, mapping is ordinarily done in blocks of addresses. The simplest and historically the earliest mapping systems mapped the entire logical address space of a program as one unit. (See Figure 1-2 for a diagram of such a system.)

More recent systems are based on mapping smaller chunks of memory, rather than the entire logical address space of a program. There are basically two kinds of address translation schemes, differing only in the structure of the mapping blocks. One form is based on variable-sized segments, the other is based on fixed-size units called "pages." Series 32000 employs a page-based mapping system.

With Series 32000, the logical address space is broken up into 32,768 pages, each with a fixed size of 512 bytes. The physical address space is broken up into the same number of pieces, each piece the same size as a page. These pieces of physical memory into which the pages are mapped are called *page frames*. Figure 1-3 shows a part of Series 32000 mapping scheme.

A page-based mapping system is usually more efficient than a segment-based mapping system because of the memory fragmentation problem associated with segment-based systems. This problem occurs often in segmented multi-program systems when the available memory space becomes fragmented into many small pieces and not enough contiguous physical memory is available to contain one large segment. By contrast, since all pages are the same size, if any physical page frame is available it can hold any page.

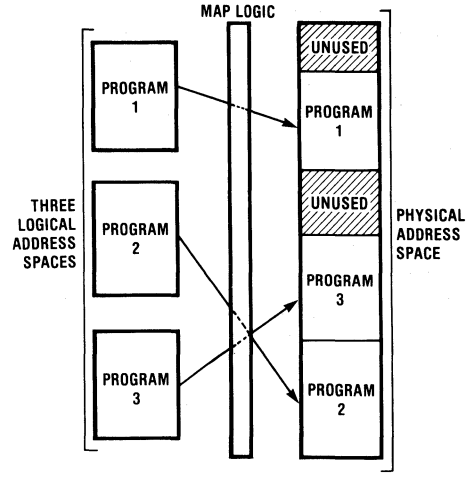


Figure 1-2. Mapping the Entire Address Space

The mapping operation is performed by the NS32082 MMU (Memory Management Unit) and is explained thoroughly in Chapter 3. This translation process is performed automatically, making use of a table in memory that contains the physical address of each page frame.

Each program or task can have its own such table, and changing the current table is simply a matter of changing an MMU register that points to the starting address of the current table. Therefore, each program or task can have its own map from logical memory to physical memory, and therefore each program or task can have its own logical address space.

Entries in the table contain protection bits along with physical addresses. These protection bits are used to provide each page with a set of protection attributes (e.g., read only). The operating system can treat a collection of pages with the same attributes as a segment in the sense of Section 1.3.1. Page based mapping thus provides a mechanism for implementing segmentation.

1.3.3 Virtual Memory

In many computer systems, the logical address space is far larger than the actual memory hardware. *Virtual memory* is a mechanism for circumventing the limits on physical memory size. Under a virtual memory system, it appears to users as if the entire logical address space were available for storage. But, in fact, at any given moment only a few pages of the logical address space are mapped onto physical space. The other pages are not present in main memory at all; instead, the information in these pages is stored on a secondary storage device, such as a disk, whose cost-per-bit is more economical.

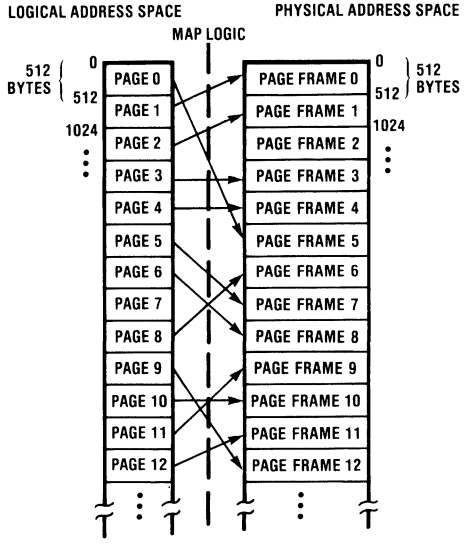


Figure 1-3. Page-Based Mapping

In a virtual memory system, whenever the computer generates a memory address, the hardware checks whether that address lies in a page that is actually in memory. If it does, the address is translated to the appropriate physical address, and the memory reference takes place normally. If the indicated page is not in memory, an operation called a page swap is performed, and the operating system software loads the missing page from disk. If this operation is performed swiftly, the user will have the illusion of a gigantic physical memory. For efficiency, when the referenced location has to be brought from the peripheral to the main memory, other locations likely to be referenced next are also brought in. Information not currently in use is removed from the main memory and returned to peripheral storage, thus making room for the new material.

Of course the beauty of virtual memory is that the user or programmer does not have to be aware of the process. He uses one consistent set of addresses called virtual addresses. The memory management hardware keeps track of where the information resides at any given time and translates the virtual address into a real location in physical memory. When the CPU finds the requested virtual address to be unavailable in main memory, it notifies the operating system which initiates a swap.

When the data to be replaced has not been modified during the time it was resident in main memory, there is no need to write it back to the peripheral device since an up-to-date copy already exists there. Under such a circumstance the old data is simply overwritten with the new data.

Virtual memory was first implemented on the Atlas computer at Manchester University, using special hardware. All computers with virtual memory since the Atlas have also required special hardware functions to implement virtual memory. Many current microprocessors do not have adequate mechanisms to support virtual memory systems. For example, in both the Z8000 and the 68000 no provision was made for restarting an instruction that causes a page fault. In Series 32000 virtual memory systems, this special hardware is provided by the NS32082 MMU (with support from the CPU chip).

1.4 Protection

The last major area of debate about computer architecture concerns the whole topic of protection: memory protection, program protection, user protection. The basic issue is what should be the granularity of the protection mechanisms that are provided. The basic difficulty is that the finer the granularity, the more the overhead associated with protection.

Some systems implement a hierarchy of protection levels from most privileged to least privileged. These levels are often called *rings*. Each ring has its own access control information for a page. Generally, a more privileged ring has access to all the information in a less privileged ring. However, because the number of rings is severely limited, usually to four, and because tasks often do not have a strictly hierarchical relationship, ring systems are seldom flexible enough for modern operating systems.

Instead, a capability-based protection system is often proposed as an alternative which allows nonhierarchical relationships between an arbitrary number of tasks. In a capability-based operating system each task has a table of operations it is allowed to perform that may affect other tasks in the system. This table is protected from direct modification by the task. Thus, the only way a task can perform an operation which could affect another task is if it has the appropriate capability in its capability table. A task may give a specific capability to another task. By restricting the distribution and type of capabilities it gives out, a task may tightly control access to the services it provides.

One problem with most capability-based systems is that the concept is carried to such lengths that it interferes with efficient accessing and processing of information within a task. Since the cost of protection is always high in these capability systems, performance suffers.

The designers of Series 32000 felt that a capability based protection scheme could be implemented at some level in the system, but that the appropriate level to do this was in the kernel of the operating system, not in the architecture itself. The basic reason for leaving capabilities out of the architecture is twofold: (1) the extra burden should not be imposed on all programmers who use this architecture or on every memory reference; (2) the implementation of a capability-based system is such a new and complex task that locking such a system into silicon before it is thoroughly proven can be very risky. The designers of Series 32000 preferred to work out the bugs in their operating system before they froze it permanently in silicon.

The protection features actually implemented in Series 32000 architecture can be divided into three groups:

1. Supervisor/user mode. A distinction is made between two operating modes of the CPU: *supervisor mode* in which all the power of the instruction set is available, and *user mode* in which only a restricted subset of the instructions are available. Supervisor mode is intended for operating systems and other trusted programs. User mode is intended for those programs that are not trusted.
2. Separate address spaces for each task. Each task running on Series 32000 has its own collection of pages constituting its address space. Access to another task's address space is impossible.
3. Protection bits in the page and pointer table entries. Associated with each page are bits that define whether that page can be read but not written into, read and written into, or neither read nor written.

All these protection features are discussed in Chapter 3, Section 3.4.

Chapter 2
High Level Language Support on Series 32000

2.1 Introduction

In Chapter 1, it was shown that with conventional architectures, the gain in programming efficiency produced by writing in high level languages instead of assembly language is usually undermined by the larger amount of memory required to store the code. This phenomenon is a result of the large number of instructions that must be generated by the compiler to map HLL concepts onto the more restricted repertoire of machine instructions. Performance is also diminished because of the large number of memory transactions generated by the instructions. In addition, when the differences between the abstractions called for by a problem and the capabilities directly implemented in the computer's hardware is very great, the code generation portion of a compiler must be extremely complex.

A primary design objective for Series 32000 was for the structure and behavior of the processor's architecture to correspond in a reasonable way with the objects and operations of high level languages. The goal was to develop a symmetrical architecture particularly suited to being the target for compilers. The architecture of Series 32000 meets that goal; it enables symmetric use of general purpose registers, memory locations, addressing modes, data types and instructions.

Compilers can easily generate high-performance (very dense and efficient) code for Series 32000. Series 32000 is particularly well suited to the Pascal high level language. Because of Series 32000's general-purpose registers, the program also executes faster. In addition, the architecture avoids special-case instructions and addressing modes that compilers have difficulty making use of.

In this chapter we will examine in detail the means by which the concepts of HLLs are supported by Series 32000 architecture; namely, by a symmetrical architecture, a sophisticated instruction set, and expanded addressing capabilities.

2.2 Data Types Supported

The objects and concepts of a high level language include constants, variables, expressions, and functions, each of which is of a particular *data type*, the type determining the range of values which the constant, variable, expression, or function can assume in the program.

A data type is said to be *supported* by a computer if the computer's instruction set contains operators that directly manipulate the data type or else has operators and addressing modes that facilitate its manipulation. Data types directly manipulated by the hardware are called *primitive data types*. Those data types supported by the hardware, but not manipulated directly, consist of ordered collections of primitive types and are called *structured data types*.

Series 32000 supports the following data types:

- primitive data types (see Figure 2-1)
  - —integers (signed and unsigned)
  - —floating point
  - —booleans
  - —Binary Coded Decimal (BCD) digits
  - —bit fields
- structured data types
  - —arrays
  - —records
  - —strings
  - —stacks

2.2.1 Integer Data Types

The integer data type is used to represent *integers*, i.e., whole numbers without fractional parts. Integers may be signed (negative as well as positive) or unsigned (positive only). Integer data types on Series 32000 are available in three sizes: 8-bit (byte), 16-bit (word) and 32-bit (double word). Signed integers are represented as binary two's complement numbers and have values in the range -2^7 to 2^7-1, -2^{15} to $2^{15}-1$, or -2^{31} to $2^{31}-1$; unsigned integers have values in the range 0 to 2^8-1,0 to $2^{16}-1$, or 0 to $2^{32}-1$. When integers are stored in memory, the least-significant byte is stored at the lowest address; the most significant byte at the highest address.

2.2.2 Floating Point Data Types

The *floating point* data type is used to represent real numbers, i.e., numbers with fractional parts. Floating point numbers are represented by an encoded version of the familiar scientific notation:

$$n = s \times f \times 10^e$$

where s is the **sign** of the number, f is called the **fraction**, or **mantissa**, and e is a positive or negative integer called the **exponent**. (Figure 2-1 shows how these values are represented by fields within the number.) Floating point numbers are available in two sizes: 32-bit (single precision) and 64-bit (double precision). Double precision offers both a larger range (larger exponent) and more precision (larger mantissa). Series 32000 floating point data type is compatible with the proposed IEEE floating point standard.

Manipulation of the floating point data type is actually handled by the NS32081 Floating Point Processor (FPU) (see Section 4.4, Slave Processors). If an FPU exists in the system, the user can treat floating point numbers (both single and double precision) as any other Series 32000 data types and may use any of the Series 32000 addressing modes to reference them. Also, conversion is provided from every integer and floating format to every other integer and floating format. If an FPU is not present, these functions must be simulated in software.

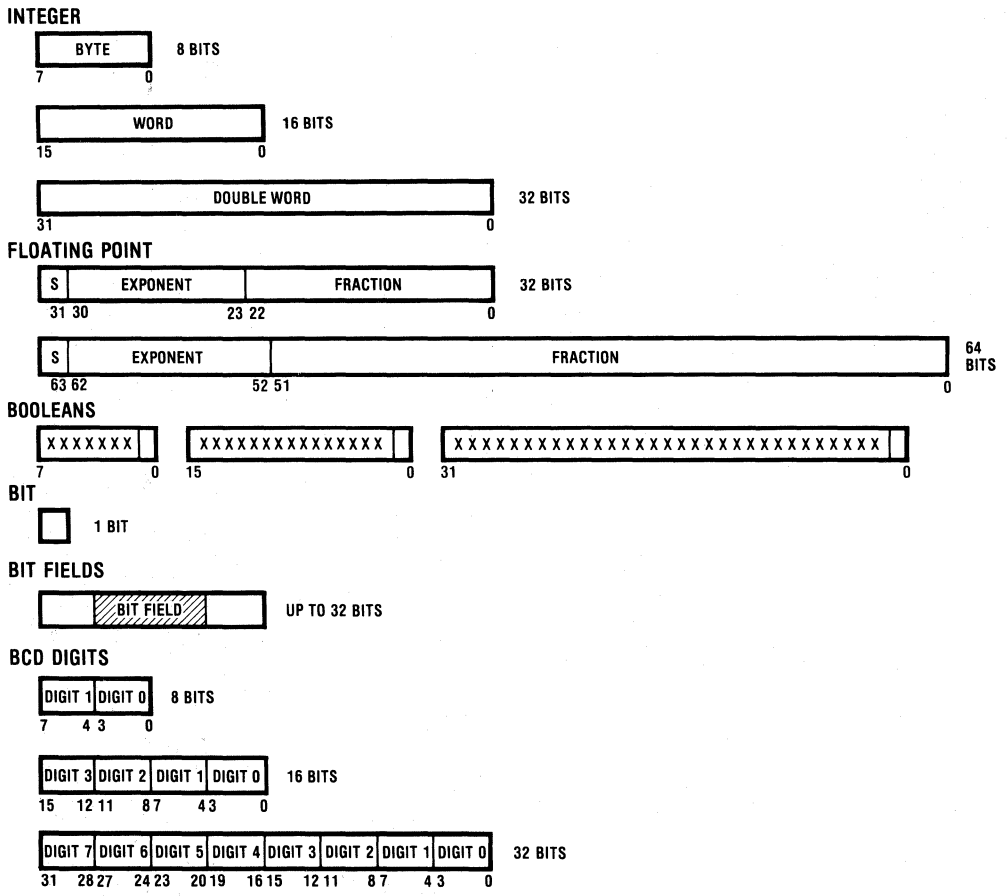


Figure 2-1. Primitive Data Types

2.2.3 Other Primitive Data Types (Booleans, Bits, BCD Digits)

The *boolean* (or logical) data type is a single bit whose value, 1 or 0, represents the two logic values **true** and **false**. A boolean data type has many uses in a program, for example, to save the results of comparisons, to mark special cases, and in general to distinguish between two possible outcomes or conditions. Booleans are represented on Series 32000 by integers (byte, word, or double word). True is integer 1; false is integer 0.

The *bit field* data type is different from other primitive data types in that the basic addressable unit is measured in bits instead of bytes. With Series 32000, bit fields may be 1 to 32 bits long, and located arbitrarily with respect to the beginning of a byte. They are useful when a data structure includes elements of nonstandard lengths, since they allow programs to manipulate fields smaller than a byte.

With the *binary-coded decimal (BCD)* data type, unsigned decimal integers can be stored in the computer, using 4 bits for each decimal digit. The BCD data type is represented on Series 32000 by three formats, consisting of 2, 4, or 8 digits. Two BCD digits may be packed into a byte, four to a word, or eight to a double word; thus one byte may represent the values from 0 to 99, as opposed to 0 to 225 for a normal unsigned 8-bit number. Similarly, a word can represent values in the range 0 to 9999, or a double word can represent values in the range 0 to 99999999.

Though BCD requires more bits to represent a large decimal number, it does have certain advantages over binary. For many business applications, the amount of actual computing to be done between source input and output is small, so that converting data from binary to decimal formats can represent a significant fraction of the total processing overhead. BCD arithmetic eliminates this conversion overhead since the computations are actually performed in decimal. Also of importance for business applications is the loss of accuracy which can result from conversions from decimal to binary and back again, a loss which is avoided by using decimal arithmetic.

2.2.4 Arrays

An *array* is a structured data type consisting of a number of components, all of the same data type, such that each data element can be individually identified by an integer index. Arrays represent a basic storage mode for all high level languages.

In Pascal programs, for example, each element of an array is referenced by the array name and an index value giving the component's position in the array. Arrays range from simple one-dimensional vector arrays to more complex multi-dimensional arrays. The elements of an array may be integers, floating point numbers, booleans, characters, or more complex objects built up from these types.

Series 32000 provides special operators that facilitate calculation of the array index and determination if the index is outside the limits of the array (see Section 2.3.4, Block, String and Array Instructions). In addition, certain Series 32000 *addressing modes* facilitate quick access to array elements (see Section 2.5.2).

2.2.5 Records

A *record*, like an array, is a structured data type with several components. However, unlike arrays, the components of a record may each be of a different data type. In high level languages, such as Pascal, a component of a record is selected by using both the name of the record variable and the name of the component. Usually, records are grouped into large arrays, called *files* in COBOL, *structures* in PL/I, and *record structures* in Pascal.

Series 32000 addressing modes facilitate quick access to record elements (see Section 2.5.2, High Level Language Addressing Modes).

2.2.6 Strings

A *string* is an array of integers, all of the same length. The integers may be bytes, words, or double words. Strings are common data structures in high level languages. For example, strings of ASCII characters (i.e., bytes) are commonly used to contain alphanumeric text.

With Series 32000, a string is represented by a sequence of integers stored in contiguous memory. Special operators exist that facilitate comparison of strings, movement of strings, and searching strings for particular integer values (see Section 2.3.4, Block, String and Array Instructions).

2.2.7 Stacks

A *stack* is a one-dimensional data structure in which values are entered and removed one item at a time at one end, called the *top of stack*. It consists of a block of memory and a variable called the *stack pointer*.

Stacks are important data structures in both systems and applications programming. They are used to store return address and status information during subroutine calls and interrupt servicing. Also, algorithms for expression evaluation in compilers and interpreters depend on stacks to store intermediate results. Block-structured HLLs such as Pascal keep local data and other information on a stack. Parameters of a procedure in a block structured HLL are usually passed on a stack, and assembly language programs sometimes use this convention as well.

Series 32000 supports both a User Stack and an Interrupt Stack. Depending on the mode of operation, one of the two stack pointers (SP_0 or SP_1) contains the memory address of the top item on the stack. Instructions exist which allow for explicit manipulation of the stack pointer, and the current stack can be used in almost all Series 32000 instructions to hold an operand (see Section 2.5.2, High Level Language Addressing Modes).

For example, an item may be pushed onto the stack by subtracting the length of the item from the stack pointer (since stacks, by convention, grow downward in memory), then moving the item to the address now pointed to by the stack pointer. An item may be popped off the stack by moving the item pointed to by the stack pointer to the destination, then adding the length of the item to the stack pointer. Both of these operations are performed by selecting the Top of Stack Addressing Mode.

Instructions also exist which push or pop the contents of one or more registers. For example, the Jump to Subroutine instruction causes the Program Counter's contents to be pushed on the stack, and the Enter instruction causes the contents of the Frame Pointer and specified General Registers to be pushed on the stack. (See Section 2.3.6, Register Manipulation Instructions, for more details.)

2.3 Instruction Set

One of the most important considerations in evaluating a computer architecture is the relationship between the machine's primitive data types and the instructions that manipulate those data types.

For example, if a processor has byte, word, and double word integers, it should have an Add operator that operates on each of these in a uniform and consistent manner. Series 32000 architecture provides a complete and comprehensive set of instructions for every hardware-recognized primitive data type. In addition, special instructions are available that facilitate manipulation of structured data types.

The instruction set includes over 100 basic instruction types, chosen on the basis of a study of the use and frequency of specific instructions in various applications; special-case instructions, which compilers cannot use, have been avoided. The instruction set is further expanded through the use of special Slave processors, acting as extensions to the CPU.

This instruction set is symmetrical, that is, instructions can be used with any general addressing mode (see Section 2.3.7), any operand length (byte, word, and double-word), and can make use of any general purpose register.

Series 32000 instructions are genuine two operand instructions, though many instructions use more (up to five) operands. This, combined with the consistent and symmetric architecture, reduces the code size considerably.

2.3.1 Integer Instructions

A large set of arithmetic instructions are provided for integer manipulation: addition and subtraction, multiplication and division (with various remainder, rounding, modulus and result-length options), two's complement, and absolute value. Other instructions include:

- Move operators that allow either zero or sign extension (a useful feature when the size of the destination exceeds the size of the source).
- Shift operators allowing logical and arithmetic shifts, as well as rotation left or right, by any amount.
- Boolean instructions (And, Or, Exclusive Or, Complement, and Bit Clear) allowing each bit in a data word to be manipulated independently.
- Two BCD arithmetic operators, Add and Subtract, handling up to eight digits at a time.
- Extended Multiply and Divide operators which return a result which is twice the size of the operands which they read.

2.3.2 Floating Point (FPU) Instructions

The NS32081supports 32-bit and 64-bit precision floating point calculations, as well as 8-, 16-, and 32-bit fixed point calculations. In addition to the floating Add, Subtract, Multiply, Divide, and Compare instructions, there is a Move instruction that doubles as a conversion instruction for converting from integer to floating point format. Instructions are also provided to ROUND off a floating point number to the nearest integer, to TRUNCate a floating point number toward zero, and to convert a floating point number to the largest integer less than or equal to itself (the FLOOR of that number). For positive floating point numbers, these last two operations have the same effect; they differ, however, for negative numbers. For example, -3.17 truncates to -3, but its floor is -4.

These instructions are implemented by the FPU and display the same symmetry, addressing modes and flexibility as the rest of the instruction set. The architecture of Series 32000 makes available to the FPU all Series 32000 addressing modes, and any instructions can be register-to-register, memory-to-register or memory-to-memory.

2.3.3 Boolean, Bit, and Bit Field Instructions

Boolean instructions treat a data word as an array of bits, and allow each bit to be handled independently. Boolean operators include And, Or, Exclusive Or, Complement, and Bit Clear.

Series 32000 provides a special *Boolean* Not instruction for implementing high level languages which require that TRUE = 1 and FALSE = 0. To simplify the handling of Boolean expressions in compilers, a Set on condition instruction stores a "1" into its only operand if a condition code check is satisfied; if not, it stores a "0".

Bit instructions allow convenient handling of individual bits or arbitrarily large bit arrays. In addition to the ability to set, clear, complement, or test any bit in memory or in a register, Series 32000 provides semaphore primitives (test and set, test and clear) for multiprocessing and multitasking coordination. Also provided is a Convert to Bit-Field Pointer Instruction which converts a byte address and a bit offset into a bit address. This allows a field address to be converted to an integer and thus passed to a procedure or function, a facility which is very useful in HLLs. A Find First Set instruction searches a sequence of bits, either in memory or in a register and returns the bit number of the first "1" bit it sees.

Two *Bit Field* instructions can access bit fields up to 32 bits in length anywhere in memory, independent of byte alignments. The Extract instruction reads a bit field, expands the result to the length specified in the opcode, and then stores the expanded result into another operand. An Insert instruction reads an operand of the length specified in the opcode and stores the low-order part into a bit field.

2.3.4 Block, String, and Array Instructions

For the many iterative operations which are required in high level languages, the Block Move and Block Compare instructions facilitate efficient generation of compiler code. They are written the same way as the standard memory-to-memory move and comparison instructions, except for the addition of a third displacement operand, which specifies how many elements (bytes, words or double words) are to be moved or compared.

Strings of bytes, words, or double words are easily manipulated with the Move String, Compare String, and Skip instructions. To avoid destructive overwriting, move and compare operations can proceed from low addresses to high addresses, or vice versa. These operations can proceed unconditionally, or be terminated when a comparison condition is met (when either a specific value is encountered or when a value is no longer encountered). Also, a string of instructions may be interrupted or aborted, and then restarted where it left off. These string instructions are comparable in their power to those available on large minicomputer and mainframe computers.

For array handling, two instructions are provided, Check and Index. The Check instruction determines whether an array index is within bounds. It allows the user to specify both an upper and a lower bound. It also subtracts the lower bound from the value being checked and stores the difference in a register, where it can be used in an Index instruction or in an index addressing mode.

The array Index instruction implements one step of a multidimensional array-address calculation. The opcode specifies the length of the second and third operands; the first operand is a general purpose register. The Index instruction performs a multiplication and an addition, leaving the result in a register. The result is then used in another Index instruction for the next dimension, or it is used in an index addressing mode.

2.3.5 Jumps, Branches, and Calls

A number of different Jumps and Branches are implemented: simple Jump, Jump to Subroutine, simple Branch, Conditional Branch, and Multiway Branch (a branch is a PC-relative Jump). Since the displacement in these instructions can be as large as the memory, there is no limit to their range. In addition, several different returns

are supported: return from subroutine, return from trap and return from interrupt. The latter two are discussed in more detail in the section covering interrupts and traps (Section 4.3). Instructions for calls to and returns from external procedures are discussed in Chapter 4, Section 4.2.3 (Programming with Modules).

2.3.6 Register Manipulation Instructions

Any general purpose register (see Section 2.4) can be accessed via the general addressing modes (see Section 2.5). Thus any Series 32000 instruction that uses a general addressing mode to access one of its operands can manipulate these registers. In addition, several instructions are provided explicitly for register manipulation.

The Save and Restore instructions manipulate the general purpose registers. The instruction format for these operations includes an immediate field of 8 bits, each bit specifying which of the eight general purpose registers are to be stored or fetched from the stack.

Instructions manipulating the special purpose registers allow these registers to be loaded and stored; bits in the program status register may be set and cleared, and the stack pointer may be adjusted. Other instructions for these registers are discussed in Section 2.4.2, CPU Special Purpose Registers.

2.3.7 Instruction Format

Series 32000 has a variable-length instruction format in which instructions are represented as a series of bytes. Figure 2-2 shows the general format of a Series 32000 instruction.

The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. (Addressing modes are discussed in Section 2.5.) Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

The Opcode specifies the operation to be performed, for example, ADD, MOV, etc., and the number of operands to be used in the instruction. The specification of an operand length (B, W, D, F, or L) is written appended to the opcode. For example, ADDW, specifies the addition of two word-long operands, while MOVF specifies a move to a single precision floating point operand. The length specification in integer instructions is encoded in the basic instruction as B = 00, W = 01, or D = 11; the length specification in floating point instructions is encoded in the basic opcode as F = 1 or L = 0.

The General Addressing Mode fields specify the addressing mode to be used to access the instruction's operands.

Index Bytes appear in the instruction format when either or both Gen fields specify Scaled Index mode. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, with the remaining bits interpreted as a signed (two's complement) value. (See Figure 2-3.) The size of an immediate value is determined from the Opcode field.

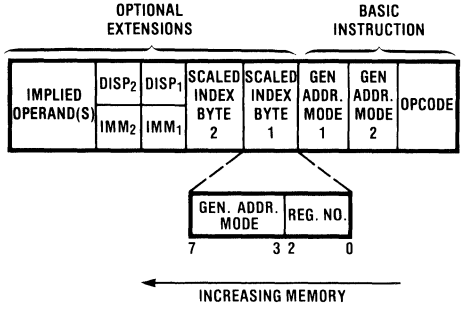


Figure 2-2. General Instruction Format

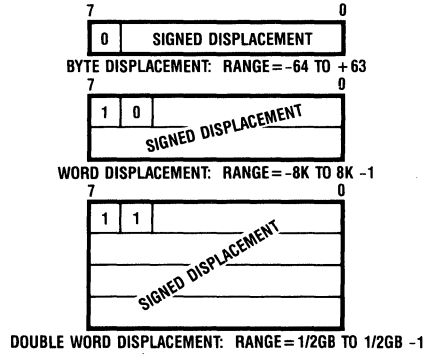


Figure 2-3. Displacement Encodings

2.3.8 Special Encodings

Two other special encodings, *reg* and *quick*, allow the very compact encoding of frequently used instructions. For example, there are quick forms of add, move and compare instructions which encode a small integer operand (range from −8 to +7) in place of a second general addressing mode.

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition.

2.4 Register Set

Series 32000 architecture supports thirty-seven registers, grouped into two register sets: sixteen general purpose registers and twenty-one special purpose registers (see Figure 2-4). Eight of the general purpose registers are located on the CPU; the other eight are located on the FPU. The twenty one special purpose registers include nine on the CPU, one on the FPU, and ten on the MMU. Besides storing operands, and the results from arithmetic operations, these registers may also be used for the temporary storage of program instructions and control information concerning which instruction is to be executed next.

2.4.1 CPU General Purpose Registers

Internal to the CPU are eight 32-bit general purpose registers R0 through R7, which provide local, high speed storage for the processor. They can be used to store bytes, words, double words, and quadruple words.

All general purpose registers are available to all instructions. Thus, the computer has freedom in its use of the registers and needn't do much housekeeping. The architecture also enables general purpose registers to be used as accumulators, data registers and address pointers. This represents a great improvement over machines that permit only a few registers to serve as address pointers, creating a bottleneck in address calculations, a very important function in high level language programming.

2.4.2 CPU Special Purpose Registers

The eight special purpose registers on the CPU chip are used for storing address and status information. The MOD register and the Processor Status Register are both 16 bits; the other registers are effectively 24 bits in length, though an additional eight bits (which in the current implementation are always set to zero) have been provided to allow for future expansion.

PC: The *Program Counter* register is a pointer to the first btye of the currently executing instruction. After the instruction is completed, the program counter is incremented to point to the next instruction. Since this register is 24 bits wide, all 16M bytes of memory can be directly addressed without the need for segmented addresses.

SP_0, SP_1: The SP_0 register points to the lowest address of the last item stored on the Interrupt Stack. This stack is normally used only by the operating system, primarily for temporary data storage and for holding return information for operating system subroutines, and interrupt and trap service routines. The SP_1 register points to the lowest address of the last item stored on the User Stack. This stack can be used by normal user programs formation.

FP: The *Frame Pointer* register is used by a procedure to access parameters and local variables on the stack. It is set up when a procedure is entered, and points to the stack frame of the currently executing procedure, which contains the parameters for the currently executing subroutine and also the volatile (as opposed to static) local variables. The procedure parameters are addressed with positive offsets from the frame pointer; the local variables of the procedure are addressed with negative offsets from the frame pointer.

SB: The *Static Base* register points to the global variables of a software module (see Section 4.2, Modular Software). All references to a module's data are relative to this register.

INTBASE: The *Interrupt Base* register holds the address of the dispatch table for interrupts and traps (see Sections 4.3.2 and 4.3.3).

MOD: The *Module* register holds the address of the Module Descriptor of the currently executing software module (see Chapter 4, Section 4.2.2).

PSR: The *Processor Status Register* holds the CPU status and control flags for Series 32000. The PSR is sixteen bits long, and is divided into two eight-bit halves. The low-order eight bits are accessible to all programs, but the high-order bits are accessible only to programs executing in Supervisor Mode. Among the bits in the PSR are the Carry bit, the Trace bit, (which causes a trap to be executed after every instruction), the Mode bit (which is set when the processor is in user mode), the Interrupt Enable bit (which if set will cause interrupts to be accepted), and several other bits which can be used by comparison instructions.

CFG: The I bit indicates the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the INT pin are "vectored"; if it is clear, these interrupts are "non-vectored". The F, M and C bits indicate the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

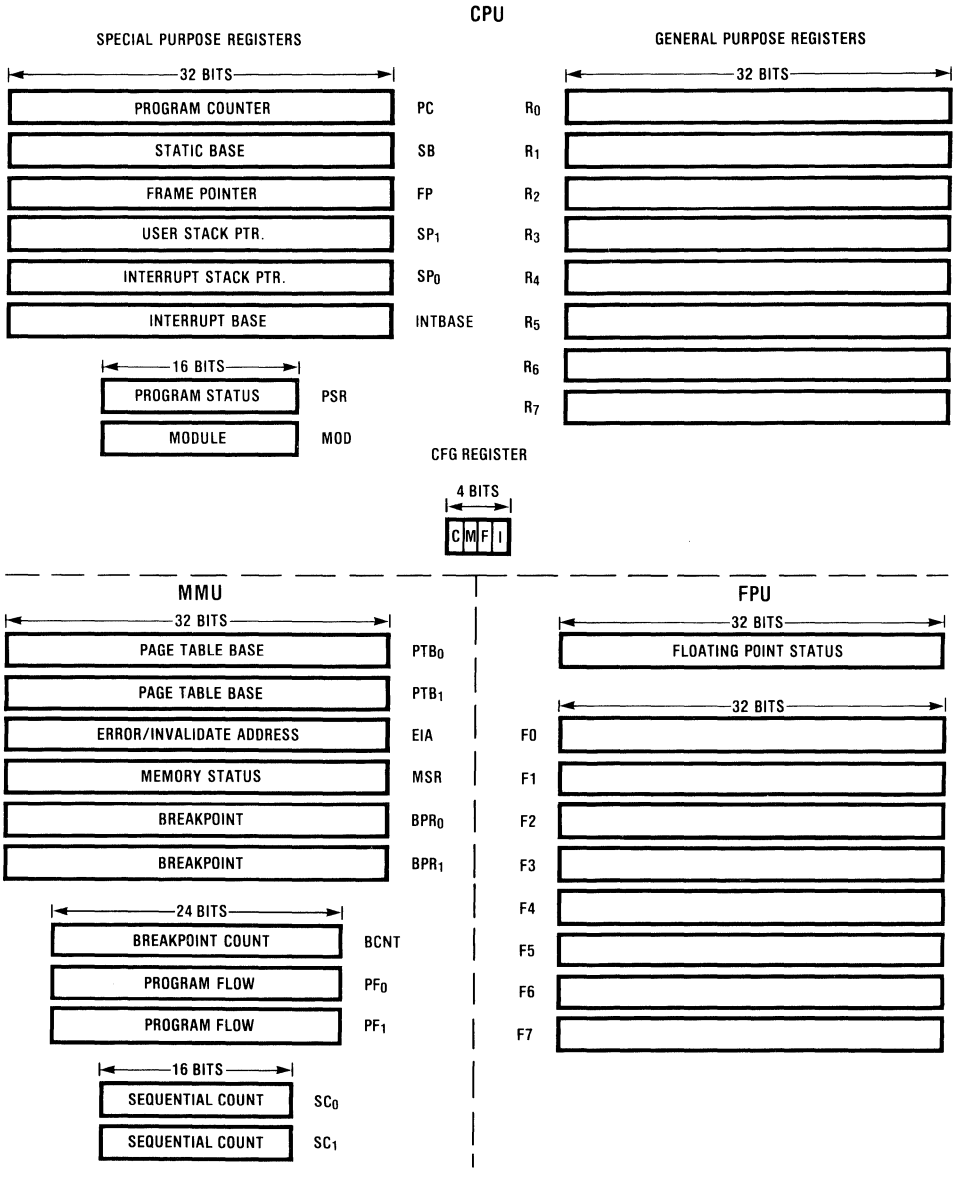


Figure 2-4. Register Set

2.4.3 FPU Registers

The *Floating Point Unit* registers are located on the NS32081 FPU slave processor and consist of eight 32-bit registers and a dedicated Floating Point Status Register. The eight floating point registers can each store a single precision operand or half of a double precision operand. When 64-bit double precision operands are to be operated upon, the specified register (n) and the next register (n + 1) are concatenated for the operation. Register n + 1 contains the high-order bits.

The *Floating Point Status* register (FSR) holds mode control information, error bits and trap enables. Like the other registers, the FSR is 32-bits wide. (See Chapter 4, Section 4.4.2, for more about the FPU slave processor.)

2.4.4 MMU Registers

The optional memory management architecture uses the following 32-bit dedicated registers to control address translation:

PTB_0, PTB_1: The *Page Table Base* registers are controlled by the operating system and point to the starting location of the translation tables in physical memory. All supervisor mode addresses are translated with the PTB_0 register. User mode addresses are translated using this register if the DS bit in the MSR is one; if this bit is zero, the PTB_1 register is used.

EIA: The *Error/Invalidate Address* register is used to invalidate addresses in the translation buffer. The translation buffer is a transparent cache of the most recently used page table entries. When an entry in a page table is modified in memory, the copy of it in the translation buffer is deleted by writing the address of the affected virtual page into the EIA register. When a PTB register is modified, all cache entries made using that register are deleted. The EIA is also used to store the address which caused a memory management exception to occur.

MSR: The *Memory Status Register* holds fields which control and examine the memory management status, and is only accessible in the supervisor mode. The functions of this register are discussed in Chapter 4, Section 4.4.2 and 4.4.3 (MMU and FPU).

Other registers in the MMU provide high level software debug facilities during program execution. These are discussed in Section 4.5.3.

2.5 Addressing Modes

Information encoded in an instruction includes a specification of the operation to be performed, the type of operands to be manipulated, and the location of these operands. An operand can be located in a register, in the instruction itself (as an immediate operand), or in memory. Instructions can specify the location of their operands by nine *addressing modes*. Two addressing modes are used to access operands in registers and in instructions—Register mode and Immediate mode. The other modes are used to access operands in memory. The address of the operand is calculated in accordance with the desired addressing mode. The calculation is done by taking the sum of up to three components:

- a displacement element in an instruction
- a pointer (i.e., an address) in a register or in memory
- an index value in a register

The nine addressing modes may also be divided into those which are standard for microprocessor architectures, and those which are particularly suited to the operations and data structures of high level languages.

2.5.1 Standard Modes

The following standard addressing modes are supported by Series 32000 architecture (see Figure 2-5 for a diagram of each one):

- Register
- Immediate
- Absolute
- Register relative

REGISTER: In the Register addressing mode, the operand is in one of the elght general purpose reglsters. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

IMMEDIATE: The immediate mode operand is in the instruction. The length of the immediate mode operand is specified by the operand length or by the basic instruction length.

ABSOLUTE: With absolute mode, the operand address is the value of a displacement in the instruction.

REGISTER RELATIVE: The register relative mode computes an effective address (the operand address) by adding a displacement given in the instruction to a pointer in a general purpose register.

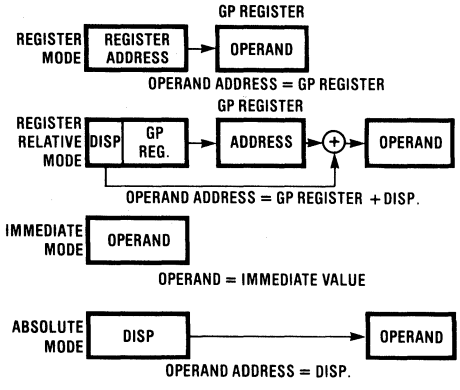


Figure 2-5. Standard Addressing Modes

2.5.2 High Level Language Modes

In addition to these standard addressing mode types, Series 32000 employs several addressing mode types which, in combination with the already powerful instruction set, make Series 32000 a superb vehicle for high level languages. They are listed below and diagrammed in Figure 2-6:

- Memory Space
- Memory Relative
- External
- Top of Stack
- Scaled Index

MEMORY SPACE: This addressing mode is identical to Register Relative, discussed above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high level languages.

MEMORY RELATIVE: The Memory Relative mode allows pointers located in memory to be used directly, without having to be loaded into registers (as is required in other microprocessors). Memory relative mode is useful for handling address pointers and manipulating fields in a record. When this addressing mode is used, the instruction specifies two displacements. The first displacement is added to a specified special purpose register, and a double word is fetched from this address. The operand address is the sum of this value and the second displacement. In accessing records, the second displacement specifies the location of a field in the record pointed to by the double word. The exact size of the contents of this field is programmable.

EXTERNAL: The External Addressing mode is unique to Series 32000, and supports the software module concept, which allows the modules to be relocated without linkage editing. This mode is used to access operands that are external to the currently executing module. Associated with each module is a Link Table, containing the absolute addresses of external variables. The external addressing mode specifies two displacements: the ordinal number of the external variable (i.e., the linkage table entry to be used) and an offset to a sub-field of the referenced variable (e.g., a sub-field of a Pascal record). External addressing is discussed further in Section 4.2 (Modular Software).

TOP OF STACK: In this addressing mode, also unique to Series 32000, the currently selected Stack Pointer (SP_0 or SP_1) specifies the location of the operand. Depending on the instruction, the SP will be incremented or decremented, allowing normal push and pop facilities. This addressing mode allows manipulation or accessing of an operand on the stack by all instructions. For instance, the TOS value can be added to the contents of a memory location, a register, or to itself, and the result saved on the stack. On most other microprocessors, in which top of stack addressing is limited to a very small number of instructions, these manipulations would require several instructions to achieve the same results. The great advantage of this addressing mode is that it allows quick reference using a minimum number of bits to intermediate values in arithmetic computations.

SCALED INDEX: This addressing mode computes the operand address from one of the general purpose registers and a second addressing mode. The register value is multiplied by one, two, four or eight (index byte, index word, index double, or index quad). The effective address of the second addressing mode is then added to the multiplied register value to form the final operand address. The Scaled Index mode is used for addressing into arrays, when the elements of the array are bytes, words, double words, floating point numbers or long floating point numbers.

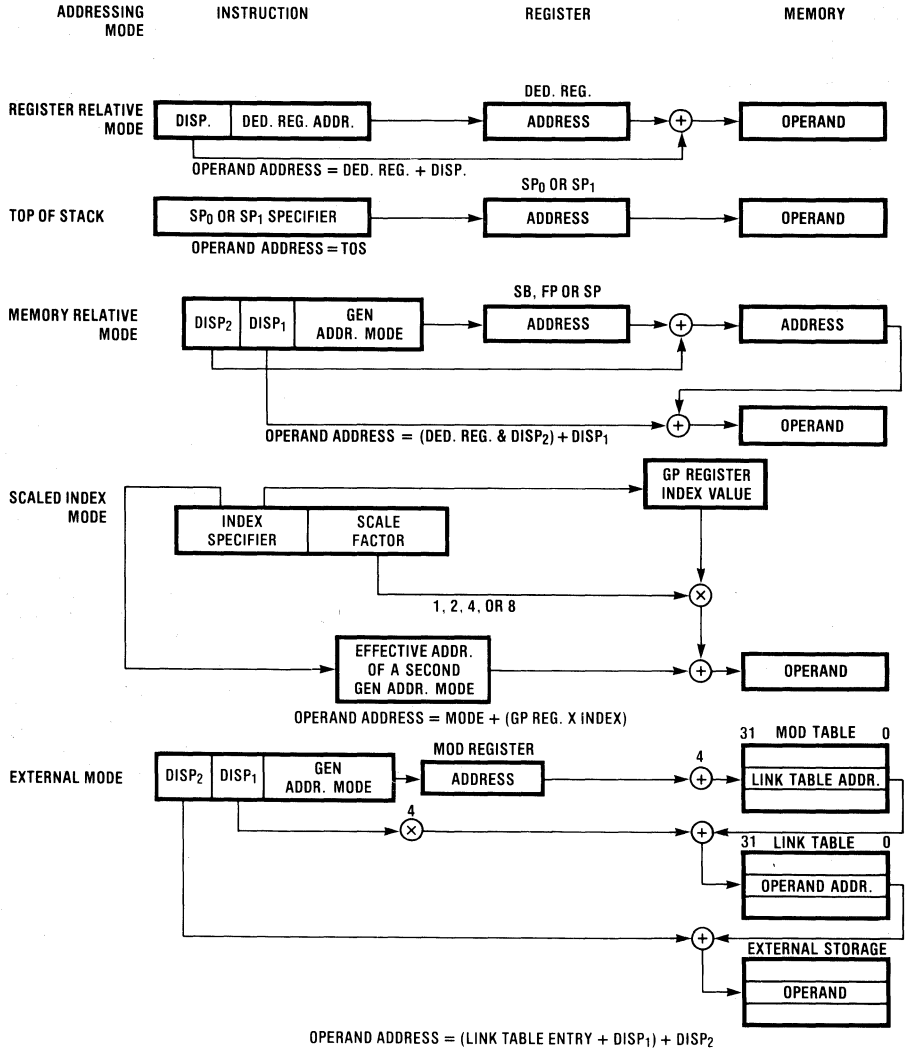


Figure 2-6. High Level Language Addressing Modes

Chapter 3
Memory Organization

3.1 Introduction

Microprocessors were first developed when the design of complex, special-purpose chips became so expensive that it was more cost-effective to use a general-purpose programmable device instead of a special-purpose chip. The programs for these early microprocessors were very small, typically requiring 2K to 8K bytes of memory and rarely exceeding 16K bytes. (This was just as well, since memory was very expensive.)

Now, almost exactly a decade since the microprocessor was invented, the memory requirements for typical applications approach those of minicomputers or even mainframes. Consequently, the memory organization issues discussed in Chapter 1 have arisen.

In this chapter, we will cover the memory organization and memory management mechanisms of Series 32000. The key topics to be discussed are page based mapping, virtual memory, memory protection, and virtual machines. The address translation, virtual memory, and memory protection mechanisms of the Series 32000 architecture are contained in the NS32082 Memory Management Unit (MMU). The MMU also contains the logic for debugging (see Chapter 4, Section 4.5) as well as on-chip cache. Special instructions are provided in the Series 32000 instruction set to control the MMU.

3.2 Mapping Mechanisms with the Series 32000

Series 32000 has a logical address space of 16 million bytes divided into 32,768 pages, each with a fixed size of 512 bytes. The physical address space is the same size and is also divided into similarly sized page frames. As described in Section 1.3.2, address translation (mapping) is the process of translating a logical address to a physical address. In Series 32000 architecture, address translation is done in units of a page. Thus two addresses next to each other in the same logical page will be next to each other in the same physical page frame, although two pages which are contiguous in logical memory may not be contiguous in physical memory.

For purposes of implementing the address translation, the 24 bits of a logical address may be thought of as consisting of two fields: the page selector field, which is the upper fifteen bits, and the offset field, which is the lower nine bits. Only the page selector bits are actually translated in the mapping process. The nine bits of the offset specify a location within a page and are passed through the mapping process unaltered. The mapping process is performed automatically by the MMU.

Basically the mapping operation consists of treating the page selector field as an index into a table of physical addresses. Entries in this table hold the upper fifteen bits of the physical address of a page frame. When a logical address is sent to the MMU, its lower 9 bits are appended to the 15-bit physical address in the table and the resulting 24-bit physical address is actually used to fetch data. (See Figure 3-1 for a diagram of this operation. This figure shows an abstract view of the Series 32000 mapping operation; in reality, a two.level mapping is employed—see Section 3.2.).

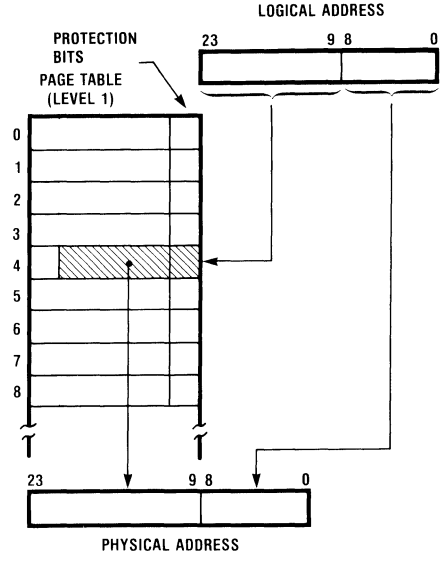


Figure 3-1. Mapping

3.2.1 Page Tables, Pointer Tables, and Entries

The address translation mechanism is carried out by tables in memory. The MMU contains a special register (PTB_1) that points to the beginning of the *page table*. This table has 256 entries, each of which is 4 bytes wide. Thus its total size is 1,024 bytes. Each entry in the page table points to a *pointer table*. Pointer tables contain 128 entries of 4 bytes; thus, the pointer tables are each contained in a page. Each entry in a pointer table points to a physical page. (See Figure 3-2 for a diagram of this pointer tree.)

Each program or task can have its own page table, and changing the page table is simply a matter of changing an MMU register that points to the starting address of the current page table. Therefore, each program or task can have its own map from logical memory to physical memory, and therefore each program or task can have its own logical address space.

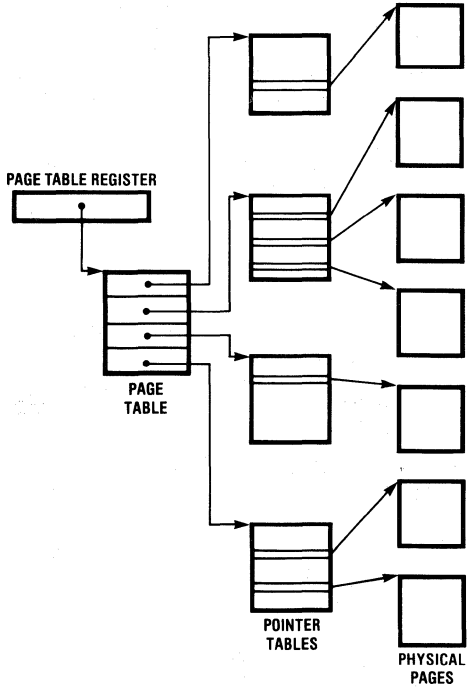


Figure 3-2. Hierarchy of Tables

Each entry in the page table or in one of the pointer tables has the same basic format (see Figure 3-3).

The high order 23 bits contain the starting physical address of the specified page frame.

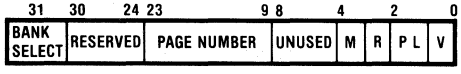


Figure 3-3. Page or Pointer Table Entry

Bits 0 through 4 contain status bits:

V The Valid Bit indicates whether the entry specifies a page that is present in memory. (See Section 5.3.1, Page Faults and the Valid Bit).

R The Referenced Bit indicates whether the page has been accessed. This bit is automatically set when the corresponding page has been accessed for reading or writing. (See Section 3.3.3, Support for Page Swapping Algorithms).

M The Modified Bit indicates whether the page has been modified during its residence in main memory. This bit is automatically set when the corresponding page is written to. (See Section 3.3.3, Support for Page Swapping Algorithms).

PL The Protection Level field indicates the level of protection provided for the page. (See Section 3.4, Protection).

3.2.2 The Complete Mapping Process

The mapping operation shown schematically in Figure 3-1 is actually accomplished by the following process:

The page selector component of the logical address (the high-order 15 bits), shown in Figure 3-1, actually consists of two subfields: the high-order 8 bits, which select an entry in the page table, and the lower-order 7 bits, which select an entry in the appropriate pointer table. (The offset component of a logical address specifies the displacement from the base of a page to the specified item). Figure 3-4 shows a more complete version of the mapping process outlined in Figure 3-1.

To speed up the mapping process the MMU provides an associative cache on the chip itself. The cache contains the 32 most recently accessed logical addresses along with their translated physical addresses. Each entry consists of the high-order 15 bits of a logical address and the high-order 15 bits of the translated physical address (see Figure 3-5).

When a logical address is passed from the CPU to the MMU, the MMU first attempts to match that logical address with an entry in the cache. If the entry is present, the physical address portion of the entry is used immediately. If the entry is not present, the MMU must fetch the page table and pointer table entries from memory before address translation can be performed.

If the entry is present, address translation requires only one clock cycle. If the entry is not present, address translation may take up to approximately 20 clock cycles. This associative table is transparent to the user and calculations indicate that it dramatically speeds up address translation since the hit ratio (the percentage of time the cache contains the entry) is about 97%.

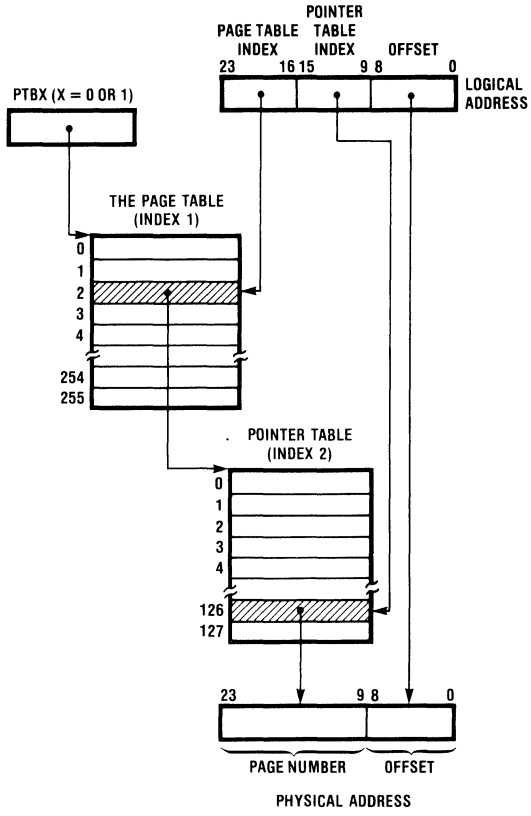


Figure 3-4. Table-Driven Mapping

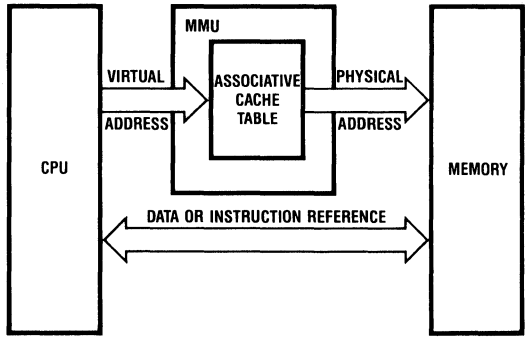


Figure 3-5. Associative Cache

3.3 Virtual Memory Mechanisms with Series 32000

Programs share many traits in common with human beings. For example, they obey Parkinson's Law. Just as work expands to fill the time available, so programs tend to expand over their lifetime to fill the physical memory available to them. Once the memory limits have been reached, further expansion of the program is difficult and error prone, usually requiring hard to manage overlays. The ideal solution to this problem is to give the program a virtually infinite (limitless) memory. A program in an infinite memory can be enlarged without bumping into any barriers. Unfortunately, memory costs usually preclude enormous physical memories. *Virtual memory*, however, gives the programmer the illusion of a gigantic memory at minimal cost.

With virtual memory, the user regards the combination of main and peripheral storage as a single large storage. The user can write large programs without worrying about the physical memory limitations of the system. To accomplish this, the operating system places some of the user programs and data in peripheral storage and brings them into main memory only as they are needed.

Series 32000 makes virtual memory operating systems easy to implement by means of its page based mapping mechanism. Programs and data are swapped between main memory and secondary storage units of a page, as was described in Chapter 1. In addition, the architecture provides several other mechanisms which support virtual memory.

Three bits in the page entry are important for virtual memory systems. These bits were discussed briefly in Section 3.2.1. In the following three subsections we will cover in much greater detail the use of these three bits in virtual memory systems. Also covered will be the instruction abort and re-execution facility, the other Series 32000 feature specifically designed to support virtual memory.

3.3.1 Page Faults and the Valid Bit

The **Valid Bit** in a page or pointer table entry indicates whether the corresponding page is present in main memory or not. Whenever an address is generated by the CPU and passed to the MMU for translation into a physical address, the MMU checks the valid bit of the table entry specified by the incoming logical address. If the valid bit is 1, the page is assumed to be present in main memory and address translation proceeds in the normal fashion.

If, however, the valid bit is 0, then the page is assumed not to be in main memory and a *page fault* occurs. A page fault is a hardware generated trap that is used to tell the operating system to bring the missing page in from secondary storage. The page fault occurs in the MMU, which generates an ABORT signal to the CPU. The ABORT signal causes the CPU to immediately halt execution of the current instruction.

3.3.2 Instruction Abort and Re-execution

When a page fault occurs, for whatever reason, the MMU sends the ABORT signal to the CPU. At this point the CPU will stop executing the instruction and return any register that was altered by the instruction to its condition before the instruction started. The operating system will then be called to initiate a page swap. Once the appropriate page is in memory, the CPU and MMU also must insure that the aborted instruction can be re-executed.

One of the problems in implementing virtual memory systems is that an instruction may generate a page fault at any time during the course of its execution. If the instruction itself occupies several bytes, it may overlap a page boundary, and the act of fetching an instruction may itself cause a page fault. Or the process of fetching the source or destination operand may cause a page fault.

In order to permit the instruction to be restarted, the ABORT signal usually causes the CPU to be returned to its state before the aborted instruction happened. The program counter is automatically saved as are the processor status register, the stack pointer and several other registers. When the operating system has completed the page swap, it executes a RETURN FROM TRAP instruction and execution resumes with the aborted instruction, all registers being restored to their old values.

String handling instructions require special treatment during an abort. Obviously it is not desirable to have a long string instruction repeated from the beginning if an abort occurs somewhere in the string. Series 32000 provides for the aborted instruction to be re-executed from the point where the problem occurred.

3.3.3 Support for Page Swapping Algorithms

To facilitate virtual memory implementation, two other bits in the page and pointer table entries are used: the Referenced Bit (R) and the Modified Bit (M).

It has been tacitly assumed that there is a vacant page frame in which to put the newly loaded page. In general such will not be the case, and it will be necessary to remove some page (i.e., copy it back into the secondary memory) in order to make room for the new page. Thus an algorithm that decides which page to remove is needed.

Choosing a page to remove at random is certainly not a good idea. If the page containing the instruction is the one chosen, another page fault will occur as soon as an attempt is made to fetch the next instruction. Most operating systems try to predict which of the pages in memory is the least useful, in the sense that its absence would have the smallest adverse effect on the running program. One way of doing so is to make a prediction when the next reference to each page will occur and remove the page whose next reference lies farthest in the future. In other words, to try to select the page that will not be needed for a long time.

One popular algorithm evicts the page least recently used because that page has a high *a priori* probability of not being in the wording set. This algorithm is called the Least Recently Used algorithm. The Referenced bit can be used to implement a version of this algorithm.

The Referenced bit is set by the hardware when the page is referenced (read or written) by an instruction. By periodically checking and clearing this bit in all page and pointer table entries, the operating system can gain insight into the frequency with which pages are being used.[1] This information can be used to select pages to be swapped out; for example, on a least recently used basis.

If a page about to be evicted has not been modified since it was read in (a likely occurrence if the program contains program rather than data) then it is not necessary to write it back into secondary memory, because an accurate copy already exists there. If, however, it has been modified since it was read in, the copy in secondary storage is no longer accurate and the page must be rewritten. The Modified bit is set by the hardware whenever a page is written to during the time it is resident in main memory.

When the time comes to swap this page the operating system can check this bit to see if there is a need for updating the copy on disc. If the bit is 1 (i.e., the page has been modified) then the page must be swapped out to secondary storage. If, however, this bit is 0, then the page has not been modified since it was last read in and it can simply be discarded.

3.4 Memory Protection Mechanisms with Series 32000

The page mechanism can also provide the basis for memory protection within a logical address space. Each page can have attributes associated with it that indicate how the page can be accessed. These attributes can allow reads only, reads and writes, or they can prevent any access at all. Entries in the page and pointer tables contain protection bits (the PL field) along with physical addresses (see Section 3.2.1). These protection bits define the attribute of that page (e.g., read only).

The interpretation of the protection bits depends on the operating mode of the CPU. A given setting of the PL field will be interpreted differently when the CPU is in Supervisor mode than when the CPU is in User mode. The bits have the following interpretation.

PL	Supervisor Mode	User Mode
00	read only	no access
01	read and write	no access
10	read and write	read only
11	read and write	read and write

The operating system can treat a collection of pages with the same attributes as a segment in the sense of Section 1.3.1. For example, a constants segment might be a set of pages containing data with the read-only attribute set, so users could not modify the data. Thus, page-based mapping provides a mechanism for implementing segmentation.

Intertask protection is accomplished by giving each task its own set of page tables. Thus each task has its own address space, providing maximum flexibility and virtual memory for each task. By changing the single register that points to the page table, one can switch to the new task's address space.

3.5 Virtual Machines

If the virtual memory hardware allows application software to execute in a different address space from the operating system, then it is possible to implement virtual machines. Software running on a virtual machine believes that it is running on a processor whose hardware provides the functions that are in fact provided by the operating system. In fact, the virtual memory hardware and I/O devices are simulated by the operating system with the aid of the real memory management hardware and I/O devices. Thus software which normally must be run alone (e.g., an operating system) can be run under the control of another operating system. This can be very useful for debugging a new operating system or running several incompatible operating systems on the same machine.

Figure 3-6 shows a simplified diagram of such a virtual machine.

Operating system A and operating system B run in different address spaces. System A manipulates the actual Series 32000 hardware, whereas system B manipulates an illusory machine consisting of Series 32000 hardware and virtual peripherals simulated by system A. The actual mechanisms employed to create such a virtual machine are somewhat technical and are covered in detail in the Series 32000 Instruction Set Reference Manual. Basically, system A constructs a simulated table onto the real page table. Virtual I/O devices are simulated similarly.

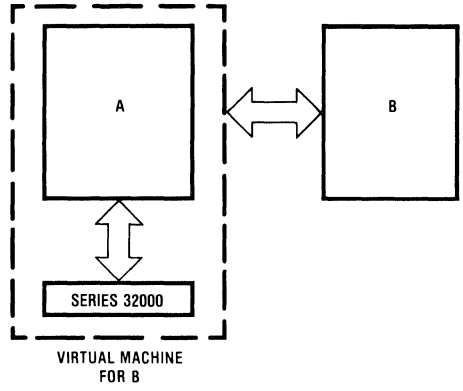


Figure 3-6. Virtual Machines

[1] Peter J. Denning, "Working Sets Past and Present," *IEEE Transactions on Software Engineering*, (SE-6, No. 1, 1980)

Chapter 4
Other Features of Series 32000 Architecture

4.1 Introduction

This chapter will discuss additional architectural features of Series 32000 that reduce the traditional gap between the semantics of high level programming languages and microprocessor architectures. Specifically, these are features which support good software design and programming practices. The topics covered include support for modular software design, input/output implementation, extension of the instruction set by the means of slave processors, and software debugging support.

4.2 Modular Software

Modular programming is one of the principle techniques for the systematic design of well-structured software. Large programs are among the most complex creations of human intellect. This complexity has been a major factor contributing to software unreliability. The concept of modularity in software design provides a means of overcoming natural human limitations for dealing with programming complexity by specifying the subdivision of large and complex programming tasks into smaller and simpler subtasks, or *modules*, each of which performs some well-defined portion of the complete processing task. Such modules may then be independently designed, written, tested, and compiled, perhaps by different programmers working in parallel.

Programs which are written as a set of modules are more likely to be correct. They are more easily understandable and therefore more easily modified, maintained and documented. Also, because communication between modules is permitted only through well-defined interfaces, the inner workings of a module need not be known to other modules. This protects a module's code and allows design changes to be done locally to a module without side effects on other modules or on the use of the system.

Nearly all HLLs incorporate features to support modular programming. For example, programs in Ada, the new Department of Defense high-order language, are composed of one or more program units—subprograms, packages or tasks—which can be compiled separately. In Pascal, separately compiled program modules may refer to variables, functions or procedures declared in another module by using certain extensions to the language, e.g., Import and Export directives.

The ultimate extension of the concept of modularity, and the ultimate simplicity in software design and implementation, is achieved when the modules are written to be used in ROM form. Such software modules are simple hardware-like components and require a minimal amount of program design overhead.

Up to now, microprocessor architectures have provided inadequate and cumbersome architecture support for a modular programming methodology. The following section will discuss the problems associated with the implementation of modularity by a microprocessor; the two subsequent sections will explain Series 32000's architectural solutions to these difficulties.

4.2.1 Overview

The major difficulty limiting the widespread use of libraries of ROM modules has been the necessity of modifying a module's addresses when it is linked with other separately compiled modules and loaded into memory for execution. Since addresses in ROM cannot be modified, it has been difficult to devise a uniform method of employing ROM modules in programs. Even when the module's code can be modified, (e.g., modules on disk), this is a tedious and often lengthy enterprise.

The problems result from the fact that when several modules are combined into a single memory image, a module's final position can vary widely. Consequently, all addresses in jumps and calls or in data accesses that are dependent on knowing the module's absolute address at run time must be different according to where the module is loaded. Similarly, when a module calls another module, the address of the called module will be dependent on the relative position of the two modules. Thus, a module's code will not be identical for each position it occupies in memory, and a linkage editor must be used to modify the addresses in each module according to its assigned position in memory.

4.2.2 Support Mechanisms

Software modules which have been compiled and assembled are known as *Object Modules* and are typically stored in relocatable object code. The function of a linkage editor is to merge the object modules into a single linear address space which may then be loaded into memory for execution. This requires binding (converting to an absolute value) all unresolved addresses. *Relocation* refers to the binding of the non-sequential addresses within the module (calls, returns, branches, and non-sequential data references); *linking* is the process of binding the addresses of subroutines or variables in other modules.

On Series 32000, no editing is required on non-sequential addresses (jumps) within a module, since Series 32000 assembly language code is position independent (PIC). This is achieved by the use of addressing modes which

form an effective memory address relative to a base register—PC, FP, SP or SB. Since the relative distance between two non-sequential addresses remains constant, the same offset relative to the base register can be used in all positions in memory. This means a program can be loaded anywhere in memory and run correctly. In addition, facilities provided by the MMU allow a program to be moved in memory after it has been linked and loaded. This is especially important in time-sharing systems where programs must be swapped in and out of main memory to allow sharing of the processor. Also, because the base register relative addressing mode allows 30-bit signed displacements, which is 6 bits more than any logical address, no code editing is ever necessary for branching, regardless of the amount of code in a module.

Position-independent code combined with the Series 32000 virtual memory mechanism allows a program to be relocated in the virtual address space as well as the physical address space. Machines that use paging or a relocation register, but lack base register relative addressing, allow programs to be moved in physical memory but do not allow them to be moved to a different virtual address after linking.

For references to variables and subroutines in other modules, Series 32000 provides a sophisticated linkage facility such that no editing of a module's external addresses is required.

To begin with, all programs for Series 32000 are organized as modules. Each module consists of three components:

1. The *Program Code* component contains the code to be executed by the processor and the module's constant data (or "literals").
2. The *Static Data* component contains the module's global variables and data, i.e., data which may be accessed by all procedures within the module. In a Pascal program, for example, this component would contain the data structures declared in the outermost block.
3. The *Link Table* contains two types of entries: External Variable Descriptors and External Procedure Descriptors. The External Variable Descriptor is the absolute address of a variable located in the static data component or program code area of another module. This value is used in the External Addressing mode, in conjunction with the current Mod Table address (see below), to compute the effective address of the external variable. The External Procedure Descriptor is used in the Call External Procedure (CXP) instruction and will be discussed in Section 4.2.3 of this chapter. There is one entry in the Link Table for each external variable and procedure referenced by the module.

In a typical system, the linker program (in conjunction with the loader) specifies the locations of the three components of a module. The static data and Link Table typically reside in RAM; the code component can be either RAM or ROM. The three components can be mapped into noncontiguous locations in memory, and each can be independently relocated. Since the Link Table contains the absolute addresses of external variables, the linker need not assign absolute memory addresses for these in the module itself; they may be assigned at load time.

To allow the transfer of control from one module to another, Series 32000 provides three structures: a Module Table in memory and two dedicated registers on the CPU.

1. The *Module Table* is set up in random-access memory starting at virtual address 0 and contains a *Module Descriptor* for each module in the address space of the program. A Module Descriptor has four 32-bit entries corresponding to each component of a module: The *Static Base* entry contains the base address of the beginning of the module's static local data area. The *Link Base* points to the beginning of the module's Link Table. The *Program Base* is the address of the beginning of the code and constant data for the module; since a module may have multiple entry points, this pointer is used with an offset from the Link Table to find them. One entry is currently unused but has been allocated to allow for future expansion.
2. The *Mod Register* on the CPU contains the address of the Module Descriptor for the current module.
3. The *Static Base Register* contains a copy of the Static Base component of the Module Descriptor of the currently executing module, i.e., it points to the beginning of the current module's static data area.

See Figure 4-1 for a description of a module's environment.

With Series 32000, modules need not be linked together prior to loading. As modules are loaded, a linking loader simply updates the Module Table and fills the linkage table entries with the appropriate values. No modification of a module's code is required. Thus, modules may be stored in read-only memory and may be added to a system independently of each other, without regard to their individual addressing. Also, since the pointers in the Module Table reach any point within the address space, modules can be located anywhere in memory.

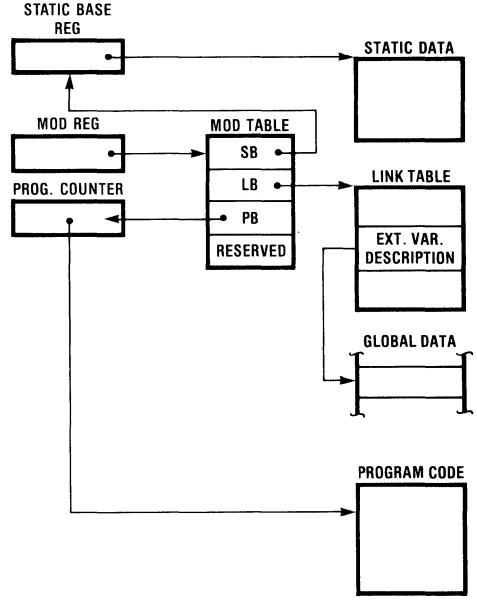


Figure 4-1. Module Run-Time Environment

4.2.3 Programming with Modules

The Call External Procedure (CXP) instruction is used to execute a procedure residing in another module. Recall that the Link Table contains two types of entries for each module in the program's address space: External Variable Descriptors and External Procedure Descriptors. The latter entries each consist of two 16-bit fields. The MODULE field contains the address of the referenced procedure's Module Table entry. The OFFSET field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer (in called module's Mod Table). This allows a called procedure to be found automatically, without requiring the calling routine to supply any addressing information.

Figure 4-2 depicts the execution of the CXP instruction where Module #2 calls Module #3.

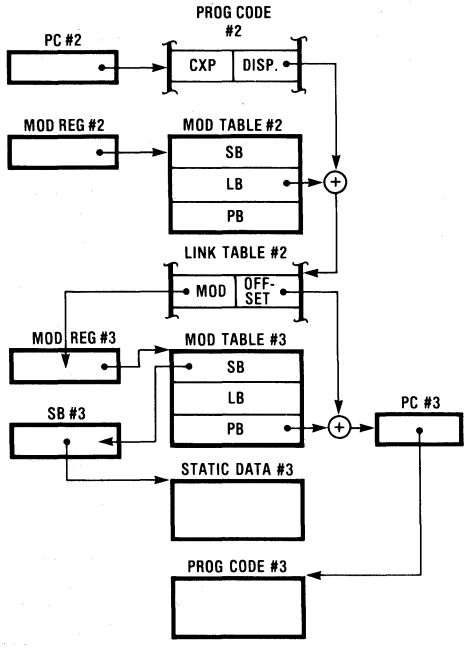


Figure 4-2. CXP Instruction

This instruction automatically performs the following sequence of operations:

1. The External Procedure Descriptor for Module #3 is found by adding a displacement specified in the instruction to the Link Table address of Module #2. (In the assembly language program this displacement is represented by a label name; the actual numerical value of the displacement is assigned by the assembler).
2. The current status of Module #2 is saved by pushing the contents of its PC and Mod registers onto the stack.
3. The Module field of the Link Table's External Procedure Descriptor for Module #3 is moved into the MOD register so that this register now points to the Module Table for Module #3.
4. The Static Base value in the Module Table is placed in the Static Base Register (this is done to speed up accesses to the module's static variables, which would otherwise be referenced by indexing into the Module Table).
5. The Offset field in the External Procedure Descriptor is added to the contents of the Mod Table's Program Base and this value is placed in the PC. The CPU is now in the environment of Module #3.

The Call External Procedure With Descriptor (CXPD) instruction allows an External Procedure Descriptor to be passed as a parameter to a called module. The External Procedure Descriptor from the calling module's Link Table is pushed onto the stack, and the called module may then use this value to call the procedure.

The Enter and Exit instructions minimize the overhead in procedure calls by automatically managing the resources that must be allocated at the beginning of a procedure and reclaimed at the end.

The Enter instruction saves the Frame Pointer (FP) of the calling module on the stack and loads the Stack Pointer value into the Frame Pointer register so that they now point to the same location, i.e., the saved Frame Pointer value on the stack. Space on the stack is allocated for the procedure's local variables, and a specified number of registers required for use by the procedure are pushed on the stack. See Figure 4-3 for an example of one procedure calling another.

Series 32000's use of the Frame Pointer allows the procedure to allocate local variables on the stack and address them as fixed offsets from the FP. Also, once the local storage is allocated, the stack can still be used for temporary storage without affecting the addressing of the locals. The programmer need not keep track of the changing offset between the SP and local storage, which is especially advantageous for nested procedure calls and recursive functions.

The Exit instruction automatically restores the registers saved by the Enter instruction, loads the value of the Frame Pointer into the Stack Pointer thus deallocating the procedure variables, and restores the previous Frame Pointer.

The Return from External Procedure (RXP) instruction restores the Static Base, the Mod Register and the PC of the calling procedure. In addition, this instruction can be used to remove the parameters which were passed to the called procedure.

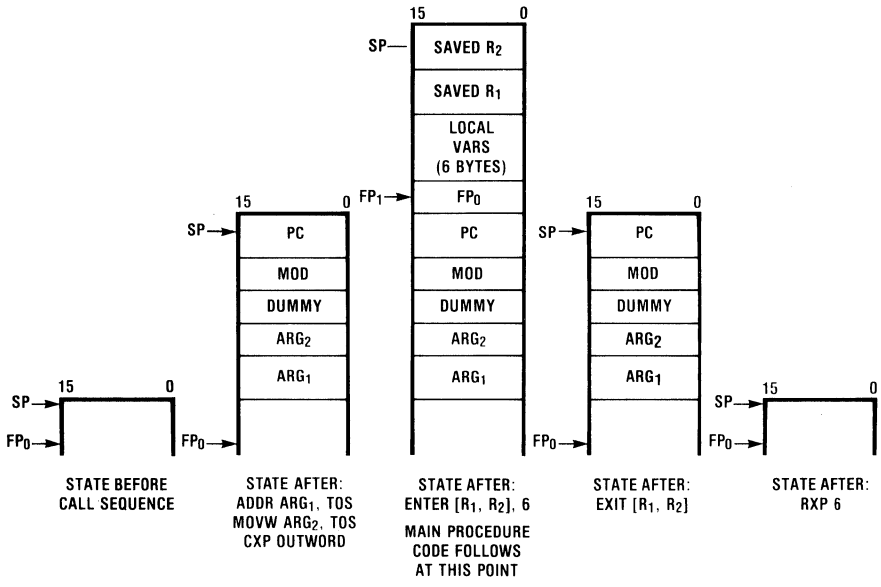


Figure 4-3. Stack Flow for Procedure Calls

Data accesses by modules are provided in the following manner:

1. Parameters and local variables on the stack may be stored and accessed with the Memory Space addressing mode or the Memory Relative addressing mode using the Frame Pointer register. Parameters are addressed with positive offsets from the Frame Pointer; local variables are addressed with negative offsets from the Frame Pointer.
2. A module's static data is accessed by using the Memory Space addressing mode with the Static Base register. Since displacement fields relative to SB register can be 1, 2 or 4 bytes, no limit is imposed on the amount of static data a module may have. Note that on other microprocessors, which handle static data in the same way as any other external references, no protection is provided for accesses by other modules. Series 32000 provides this protection at the hardware level. The Mod Table allows each module to have its own static data area so that a procedure being executed by a module will not modify that module's data. In applications requiring two or more tasks to be executing the same code concurrently, this protection is essential to insure re-entrancy.
3. For operands that are external to the currently executing module, the External addressing mode is used. This addressing mode specifies two displacements. The first is added to the Link Base entry in the Mod Table to obtain the External Variable Descriptor entry in the Link Table. The second displacement is added to the External Variable Descriptor to compute the effective address of the operand. Since both displacements may be as large as the logical address space, there is no limit to the size of the Link Table or to the size of the external variable (which might be a structure rather than a single data element).

Indexing by the contents of any one of the CPU's eight general purpose registers is an option on all addressing modes which generate an effective address to memory, so that a static or external variable can also be an array. For example, to access an array that has been passed by reference, the starting address of the array may be found by using the Memory Space Mode relative to the FP; this value can then be loaded into a general purpose register and used with the Scaled Index mode.

4.3 Input/Output

The input/output structure defined by a computer's architecture provides the interface between the central processor and the outside world, as well as between the processor and its secondary storage devices, external support circuits and slave processors.

The first two sections will discuss one aspect of Series 32000's architectural support for I/O operations, specifically, its sophisticated and efficient exception handling mechanism.

4.3.1 Overview

Program *exceptions* are conditions which alter the normal sequence of instruction execution, causing the processor to suspend the current process and call the operating system for service. An exception resulting from the activity of a source external to the processor is known as an *interrupt*; an exception which is initiated by some action or condition in the program itself is called a *trap*. Thus, an interrupt need have no relationship to the executing program, while a trap is caused by the executing program and will recur each time the program is executed. Series 32000 recognizes twelve exceptions: nine traps and three interrupts.

The exception handling technique employed by an interrupt-driven processor determines how fast the processor can perform input/output transfers, the speed with which transfers between tasks and processes can be achieved, and the software overhead required for both. Therefore, it determines to a large extent the efficiency of a processor's multiprogramming and multitasking (including real-time) capabilities.

Exception handling on Series 32000 makes use of the hardware structures provided for external procedure calls (see Section 4.2.2) and, in addition, establishes a Dispatch Table in memory whose base address is contained in the CPU Interrupt Base register. This table contains an External Procedure Descriptor for each interrupt service procedure required. See Figure 4-4.

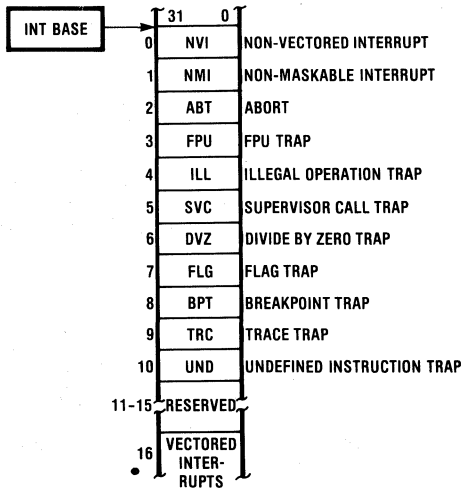


Figure 4-4. Dispatch Table

For purposes of addressing the Dispatch Table, each of the twelve exceptions has been assigned a number. This exception number (or Interrupt vector) is used to compute the starting address of the service procedure for the particular exception required, i.e., the exception number is multiplied by 4, added to the contents of the Interrupt Base register, and this value is used as an index into the Dispatch Table to obtain the External Procedure Descriptor of the service routine to call.

When an exception occurs, the CPU automatically preserves the complete machine state of the program immediately prior to the occurrence of the exception. Depending on the kind of exception, it will restore and/or adjust the contents of the Program Counter, the Processor Status register, and the current Stack Pointer. A copy of the PSR is made and pushed onto the Interrupt Stack. The PSR is set to reflect Supervisor Mode and the selection of the service routine's Interrupt Stack. The Interrupt exception number is then used to obtain the address of the External Procedure Descriptor from the Dispatch Table, and an External Procedure Call is made. As with any such call, the Mod register and the Program Counter are pushed onto the Interrupt Stack. See Figure 4-5.

To return control to the interrupted program, one of two instructions is used. The Return From Trap instruction (RETT) is used for all traps and nonmaskable interrupts. It restores the PSR, the Mod register, and the PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it discards a specified number of parameters from the User's stack. See Figure 4-6.

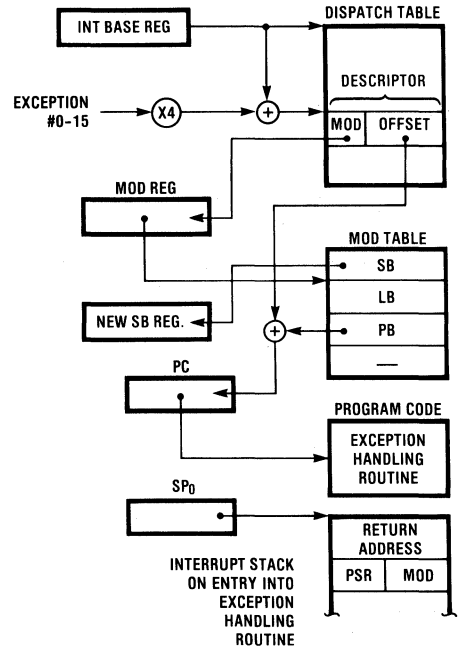


Figure 4-5. Non-Vectored Interrupts and Traps

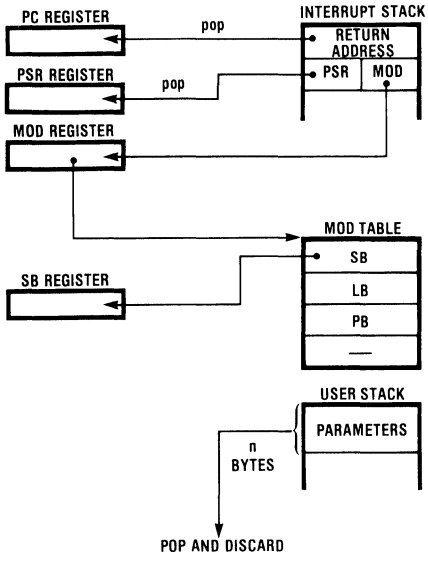


Figure 4-6. Return from Trap Instruction

For maskable interrupts, the Return from Interrupt (RETI) instruction is used. It is basically the same as the Return From Trap instruction except that any Interrupt Control Units (see Section 4.3.3 below) are informed that interrupt service has completed. Also, since interrupts are generally asynchronous external events, this instruction does not pop any parameters.

Series 32000 implements a five level priority system for scheduling exceptions which occur in the same instruction. They are ordered as follows:

1. Traps other than trace (highest priority)
2. Abort trap
3. Non-maskable interrupt
4. Maskable interrupts
5. Trace trap (lowest priority)

Maskable interrupts may individually be assigned separate relative priorities (see below). Exceptions with the same priority are serviced in the order received.

This, then, is the basic plan for exception handling on Series 32000. The specifics of interrupt and traps are discussed in the following two sections of this chapter.

4.3.2 Interrupts

Series 32000 provides three types of interrupts: Nonmaskable, Vectored, and Non-vectored.

Non-maskable interrupts cannot be disabled and occur when catastrophic events (such as imminent power failure) require immediate handling in order to preserve system integrity. A non-maskable interrupt also occurs when a breakpoint condition is met. (See Chapter 4, Section 4.5.2).

The Non-vectored interrupt mode may be used by smaller systems in which an interrupt priority system is not required. In this case, no index into the Dispatch Table is needed, and the CPU simply uses a default vector of zero.

For Vectored interrupts, prioritization of interrupt requests is provided by the NS32202 *Interrupt Control Unit*. The basic idea in a priority interrupt mechanism is that each device along with its interrupt handler is assigned a rank indicating its priority. An interrupt handler can then be interrupted only by devices with a higher priority.

Each Interrupt Control Unit can prioritize up to sixteen interrupt requests, eight of which can be from external peripheral devices. The ICU provides a vector used as an index into the Dispatch Table to obtain the address of the service routine required. In a system with only one ICU, the vectors provided must be in the range of 0 through 127.

To further expand the interrupt handling capability of a system, a single NS32202, acting as the Master ICU, can be cascaded with up to sixteen additional NS32202s, providing up to 256 levels of hardware or software interrupt. To support the cascaded configuration, a *Cascade Table* is established in memory, in a negative direction from the Dispatch Table. The entries in the table are the 32-bit addresses pointing to the Vector Registers in each ICU. To address the Cascade Table, the ICU provides a negative vector number. The fact that it is a negative number indicates to the CPU that the interrupt vector is from a cascaded ICU. See Figure 4-7 for a detailed explanation of cascaded interrupts.

The Interrupt Control Unit can function in either a fixed priority or an auto-rotate mode. In auto-rotate mode, the interrupt source, after being serviced, is rotated automatically to the lowest priority position.

All interrupts except the non-maskable interrupt may be disabled by the program with the Bit Clear in PSR instruction; each of the ICU's 16 interrupt sources can be individually masked by setting a bit in that device's Mask Register.

Interrupt handling on Series 32000 provides a number of features which contribute to efficiency and programming flexibility. For example, rather than saving all registers when an interrupt occurs, Series 32000 automatically saves only the Program Counter, the Program Status Register and the Mod Register; the other registers are under program control. They may be saved and restored by specifying the required ones in a single instruction, allowing for extreme flexibility in adjusting interrupt response speed. Fast context switching for interrupts is facilitated by the treatment of all memory locations as though they are internal general purpose registers by virtue of memory to memory operations. This allows a temporary variable to be left in memory during a context switch. Also, the use of an Interrupt Stack allows context switching in a multiprogramming or multitasking environment to be done without having to disable interrupts.

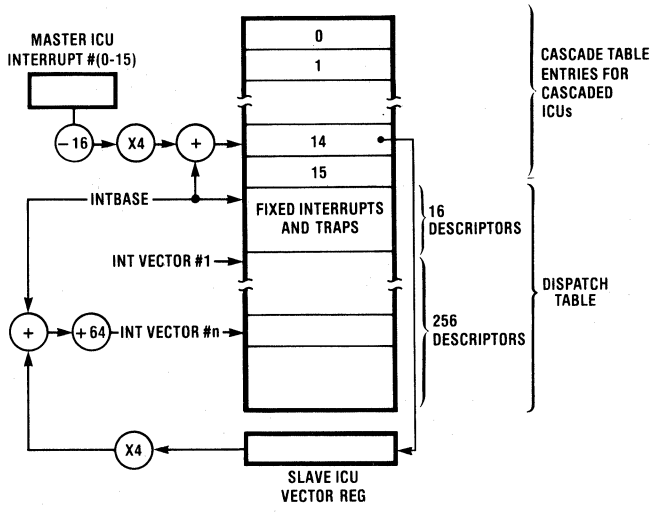


Figure 4-7. Cascaded Vectored Interrupts

4.3.3 Traps

Series 32000 recognizes nine traps. Three of the traps are implemented by explicit instructions: the Flag Trap (FLAG) allows overflow checking in any arithmetic operation and is enabled by setting the F bit in the PSR; the Supervisor Call Trap (SVC) is used to transfer to system mode software in a controlled way, typically to access facilities provided by the operating system. The Breakpoint Trap (BPT) instruction is used for program debugging and is discussed in this chapter, Section 4.5.2.

The Abort Trap (ABT) occurs when an attempt is made to access a protected page in memory or when page swapping is required in the MMU. If the page fault occurs in a string instruction, the processor state is set to reflect the progress made by the instruction up to the time of the trap; all other instructions are re-executed from the beginning.

The Illegal Trap (ILL) results when a privileged instruction occurs while the processor is in the User mode. Traps are also provided for undefined opcodes (UND), for attempted division by zero (DVZ), and for the occurrence of an exceptional condition in an FPU or Custom Slave instruction (FPU). The Trace Trap is enabled by setting the T bit in the PSR and is used for program debugging (see Section 4.5.2).

All traps except the Trace trap occur as an integral part of the execution of an instruction, and are serviced before interrupts. The return address pushed by any trap except the Trace trap is the address of the first byte of the instruction during which the trap occurred; the return address of a Trace trap is the address of the next instruction to be traced. (See Section 4.5.3).

4.3.4 Memory-Mapped I/O

The architecture of Series 32000 implements a *memory-mapped I/O* system, in which peripheral devices are treated as a specified section of memory. The basic motivation of a memory-mapped system is to allow the use of the full range of the microprocessor's instructions and addressing modes for I/O operations.

Each device interface is organized as a set of registers (or ports) that responds to read and write commands to locations in the normal address space of the microprocessor. For example, a memory store becomes an I/O *write* if a peripheral device is addressed; a load from memory becomes an I/O *read*. A compare with memory is a very powerful instruction that can take a group of input sourced data and successively compare their magnitude with a value in a register. Also, data in an external device register can be tested or modified directly, without bringing it into memory or disturbing the general registers.

Memory-mapped I/O allows I/O operations to be performed directly in a high level language, i.e., an I/O device may be declared as a data structure and then manipulated with the use of pointers. In an isolated I/O system, assembly language subroutines for I/O must be written and then called by the HLL. Memory-mapped I/O also insures that the I/O space is protected by the same memory management facilities that are used to protect critical areas of memory.

4.4 Slave Processors

A *slave processor* is an auxiliary processing unit that operates in coordination with Series 32000 CPUs, allowing architectural capabilities which, in view of the limitations in

contemporary integration technology, could not otherwise be provided. Communication between the master CPU and the slave processors takes place by means of a very fast, well defined, and self-contained protocol which is transparent to the programmer.

Series 32000 includes two slave processors: the NS32081 *Floating Point Unit (FPU)* and the NS32082 *Memory Management Unit (MMU)*. In addition, the CPUs provide the capability of communicating with a user-defined, generalized *"Custom" Slave Processor*.

4.4.1 Overview

Series 32000 CPUs recognize three groups of instructions as being executable by external Slave Processors: 1) Memory Management Instructions, 2) Floating Point Instructions, and 3) Custom Instructions.

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID byte followed by an Operation Word. The ID Byte identifies the instruction, as being a Slave Processor instruction, specifies which Slave Processor will execute it, and determines the format of the following Operation Word of the instruction. The Operation Word specifies the size and number of operands, the addressing modes used to access them, and the type of operation to be performed.

In all slave operations, the CPU fetches the instruction, performs any address calculation that may be needed, and then routes the instruction with the appropriate data to the slave processor for execution. The actual data manipulation is handled by the Slave Processor. If the necessary slave processor chip is not in the system, the CPU generates a software trap, allowing the instruction to be emulated with software routines.

Though the slave processor is external to the host CPU, all of the CPU's registers and facilities (such as effective address calculation, memory bus interface, etc.) can be considered an integral part of the system.

A four-bit CFG register, located in the control section of all Series 32000 CPUs, indicates to the CPU the presence of Slave Processors in the system configuration (see Figure 2-4). The F, M, and C bits indicate the availability of the FPU, the MMU and a Custom Slave Processor, The I bit indicates the presence of the NS32202 Interrupt Control Unit (see Section 4.3.2). These four bits must be set by the user during system initialization with the Set Configuration Instruction (SETCFG).

There are no restrictions on the number of slaves that can be used in the system, so long as only one slave of each kind is on the bus. Thus, four or five slave processors, each with a different instruction set, could work alongside the CPU on the same bus.

The slave processor concept has two main advantages for software development. First, the slave processors are so designed that when integration technology advances to the point where slave processor hardware can be incorporated within the CPU chip, no software modifications will be required—the same programs will simply execute much faster. Secondly, the programmer has the option of building an entry-level system without slaves by using software emulators. Later, higher performance systems can be built by simply adding the slave chips and removing the emulators.

4.4.2 MMU

The MMU provides dynamic address translation, virtual memory management, memory protection, and both hardware and software debugging support.

The MMU address translation and virtual memory mechanisms are described in Chapter 3; Section 4.5 of this chapter covers the debugging facilities of the MMU. In addition, six instructions are provided for manipulating the MMU's status. The Read Address Validate (RDVAL) instruction and the Write Address Validate (WRVAL) instruction provide read and write address translation validation for the user mode. The Load MMU Register (LMR) instruction allows the programmer to store data into any of the MMU registers. The Store MMU Register (SMR) instruction aliows any register to be read.

The MOVSU and MOVUS instructions permit the operating system to transfer data to and from user space. Without these instructions, the operating system would have no way of accessing data in the user's address space. Many microprocessors that distinguish supervisor mode from user mode lack this instruction, and the design of operating systems for these machines is adversely affected.

4.4.3 FPU

The FPU extends the Series 32000 instruction set with very high-speed floating-point operations for both single- and double-precision operands, as well as 8, 16 and 32-bit fixed-point calculations.

The FPU contains eight 32-bit data registers and a 32-bit Floating Point Status Register (FSR) which contains mode control information, the floating point error bits and trap enables. The data registers contain 32-bit single precision operands; for 64-bit double precision operands, two registers are concatenated.

Unlike other microprocessors which support floating point operations, the architecture of Series 32000 makes available to the FPU all Series 32000 addressing modes. For example, the Scaled Index mode permits an array of floating point data elements to be addressed by its logical index, rather than its physical address. Also, any instructions can be register-to-register, register-to-memory, or memory-to-memory.

The FPU executes 18 instructions which supplement the integral arithmetic instructions and provide conversion from one precision type to another. Three separate processors in the chip manipulate the mantissa, sign, and exponent, respectively, under the control of microcode stored on the chip. See Chapter 2, Section 2.3.2 for more about FPU operators.

Traps are provided for overflow, underflow, divide by zero, reserved operand, invalid operations, illegal instructions and inexact results. All traps can be individually enabled or disabled by the programmer.

4.4.4 Custom Slaves

The user-defined Custom Slave instruction set can be used to control any generic external chip. This chip is assumed to need some opcodes for arithmetic-like calculations, some opcodes for data moves, and some opcodes for examining and modifying status registers. The instruction set defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave, and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

4.5 Debugging Facilities

Debugging is one of the most difficult stages in program development. Though structured design techniques and modular programming have helped to reduce program debugging time, 20% of software development effort remains committed to this enterprise. Clearly, any debugging assistance provided by the hardware is of particular value. The support provided by Series 32000 is unique for microprocessors.

4.5.1 Overiew

Hardware support is provided for two operations that are crucial to program debugging: *flow tracing* and *breakpointing*. The implementation of these operations uses two sets of registers on the MMU—the Breakpoint and Flow Tracing registers—and the Breakpoint Trap instruction.

4.5.2 Breakpoint Trap and MMU Breakpoint Registers

Setting breakpoints is a technique for halting a program's execution at a particular instruction or data access for purposes of examining the program's state and thereby determining the cause of improper program behavior.

With Series 32000, breakpoints may be set either when a specified address is accessed or after a specified number of such accesses have been made. Also, more than one breakpoint address may be simultaneously selected, allowing a halt to be implemented after either fork of a conditional branch. These facilities are provided by the Breakpoint Trap instruction (BPT) and three dedicated registers located on the MMU.

The Breakpoint Trap instruction is a one byte instruction which replaces the first byte of the opcode of the instruction that is to be breakpointed. To allow breakpoints to be set in PROM, as well as RAM, two Breakpoint registers, BPR_0 and BPR_1 are provided. These registers hold the doubleword addresses of two selected breakpoints which are compared with the contents of the address bus for every memory cycle. When a breakpoint address appears in the program, and when other conditions specified by the contents of the register are met, a non-maskable interrupt occurs.

Because these registers are located in the MMU, they may be selected to look at either the virtual addresses from the CPU or the physical addresses from the MMU. In addition, the Breakpoint registers may be designated to operate when the indicated address is either written to or read from, or when there is an instruction fetch.

A third register on the MMU, the Breakpoint Count register, specifies the number of matches of the BPR_0 register breakpoint condition to pass over before a breakpoint occurs. This is useful for selecting a particular iteration in a loop instruction. See Figure 4-8 for a schematic representation of the operation of the three Breakpoint registers. In this example, the program contains a loop which will be executed 100 times. For purposes of debugging, the breakpoint is set to occur on the last time through the loop. This is done by setting BPR_0 to the address of the particular instruction, and setting the BC register to 99, this being one less than the number of times the loop will be executed in the program.

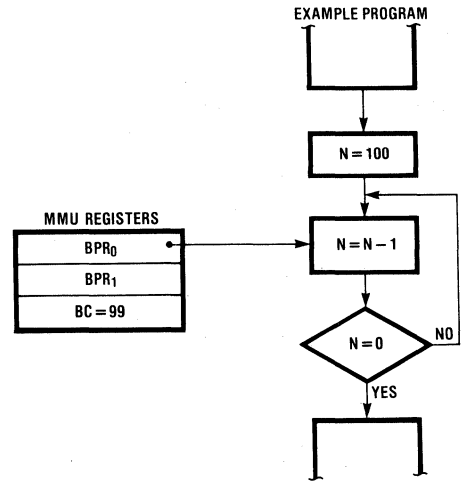


Figure 4-8. Breakpointing

In most other microprocessors, breakpointing is provided by a trap or breakpoint instruction which single steps the CPU. This can result in myriad problems for a virtual memory system. First and foremost is the fact that all addresses emanating from the CPU are virtual addresses. It is often necessary when debugging supervisor-mode software to be able to set breakpoints as absolute addresses; i.e., as addresses in physical memory. This is not possible with CPU-based debugging techniques, since the CPU has no concept of the distinction between the two types of addresses. Also, the setting of breakpoints with special instructions that overlay existing code can cause much additional overhead for the memory manager. For these and other reasons, the designers of Series 32000 have chosen to implement debug support on the MMU.

4.5.3 MMU Flow Tracing Registers

Flow tracing provides a chronicle of the actions and results of individual steps in a program during its execution. It allows the program's recent history to be examined at specified instructions or breakpoints in order to determine the reason for any undesired program behavior.

Series 32000 supports program flow tracing with four dedicated registers in the MMU and the Trace trap bit in the CPU's PSR. If flow tracing is activated (by means of a bit in the MSR), two 32-bit Program Flow registers (PF_0 and PF_1) will always hold the addresses of the last two instructions which were executed out of sequence. The two 16-bit Sequential Count registers (SC_0 and SC_1) will keep a record of the number of sequential instruction fetches between each change in program flow. All four of these registers may be cleared by the Load Memory Register (LMR) instruction. Figure 4-9 shows an example of the use of the Flow Tracing registers to determine which of two paths through a program were taken prior to the execution of the instruction pointed to by Breakpoint Register 0.

The user can select an instruction or series of instructions to trace by means of the Trace trap, which is enabled by setting a bit in the PSR. When the Trace trap is enabled at the beginning of an instruction, a trace trap will occur at the end of that instruction, and user software may then be employed to investigate the contents of the CPU registers. The trap will occur after each instruction, so long as the bit is set.

This trap is implemented in such a way that one and only one trace trap has the lowest priority of any exception, any other trap or interrupt request which occurs during a traced instruction is allowed to complete its entire service procedure before the Trace trap occurs. Also, unlike other traps, where the address of the first byte of the instruction during which the trap occurred is pushed onto the stack, the Trace trap insures that the return address to be pushed is that of the next instruction to be traced.

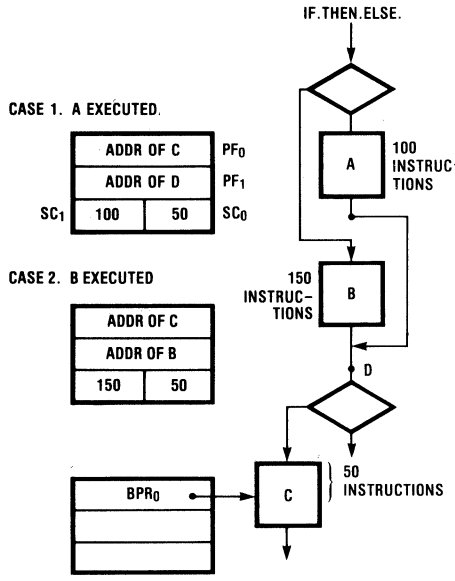


Figure 4-9. Flow Tracing

Benefits of Demand Paged Virtual Memory

Chapter 1
An overview of the past, present, and future.

Microprocessors were originally designed as general-purpose, software-programmable devices that would provide an alternative to prohibitively expensive special-purpose chips. The limitations of early technology, while equal to the modest performance requirements of original applications, generally limited the size of memory to 16K bytes.

1.1 Microprocessor memory architecture was primitive at first. The address on the microprocessor's address bus went directly to main memory[1]—an address was an address.

Even today, most microprocessor systems based on this architecture are either limited to 64K bytes of *total* memory, or stuck with a small number of 64K byte segments (typically, fewer than 128).

In a segmented address space, the address space necessarily consists of a small number of distinct uniform address spaces. Since an address pointer can only be incremented from the top of one segment to the bottom of another segment, all data structures larger than the maximum segment size must be broken down to fit into two or more segments. Consequently, a program that needs 100,000 bytes of data in main memory requires the user to split the total data into two segments, neither of which can be larger than 65,536 bytes. And, to allow the system to access the data, the user must create complex instructions that enable the system to figure out where in the two segments the data it needs actually resides, and how to efficiently access that data when it overlaps both segments.

Altogether, the complications inherent in segmentation (usually requiring hard-to-manage overlays) present pitfalls to the user. The development of the 8086 from the 8080, though a heroic attempt to expand address space while preserving some measure of software compatibility, resulted in the most striking example of the problems of segmented architecture.

1.2 Programs share many traits in common with human beings. For one thing, they follow Parkinson's Law.

Just as work expands to fill the time available in which to do it, programs tend to expand, over their lifetime, to fill the memory available to them. The memory requirements of typical applications today commonly strain the capacity of minis and even mainframes. Programs such as high-level language compilers, or the recently enhanced version of VisiCalc™, need over 250K bytes of main memory alone; graphics programs can require over a million bytes of secondary storage.[2]

With the advent of extremely fast VLSI (very-large-scale-integration) technology, it is now possible to close the performance and capacity gap between a mainframe and a microprocessor-based system. But for a small microprocessor-based system to expand to the full capability of a mini or a mainframe, its architecture must optimize performance without compromising user protection. It must accommodate the largest applications, yet remain cost-effective. Also, to put off obsolescence and minimize development costs, it must be able to utilize the enormous software inventory available on minis and mainframes.

1.3 Increasingly, systems designers and programmers are coming to the conclusion that Series 32000 will be the foundation for the next generation of high-performance, low-cost computers.

Why? Because, among its virtues, Series 32000 features a totally new, totally practical microprocessor architecture—not simply an enhancement of an existing one.

Unlike any other commercial processor (micro, mini, *or* mainframe) it is designed to fully support the use of high-level languages and modular-software programming.

It introduces a powerful, highly symmetrical, instruction set that includes over 100 genuine two-operand instruction types, but avoids special-case instructions that compilers cannot use.

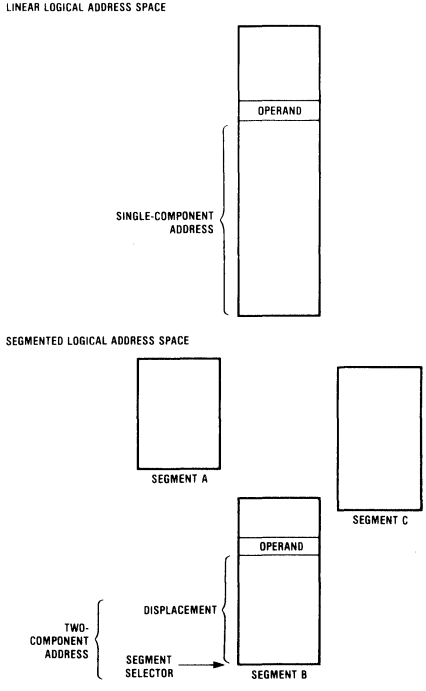
It is the first commercial microprocessor capable of implementing Demand Paged Virtual Memory as a means of solving large-memory-management problems. Its architecture also supports uniform addressing—addresses start at location zero and proceed uniformly until the entire virtual address space[3] is filled. As a consequence, the memory configuration of a Series 32000-based system is completely flexible. A designer can maximize the use of the system's main and virtual memory resources, and achieve a heretofore-unrealizable level of performance at a minimal cost *(Figure 1)*.

---

[1] *Also called semiconductor*, or *real* memory.

[2] Also called *peripheral*, or *mass* storage.

[3] Also called the *logical* address space.



TL/EE/8666-1

Figure 1: Uniform vs. segmented architecture

Chapter 2
With virtual memory, a system appears to have more memory than it actually does.

Although memory hardware is becoming less expensive, few systems can afford to have megabytes of main memory. A megabyte of secondary disk storage is likely to remain considerably less expensive than a megabyte of RAM for some time. To circumvent this practical limitation on the size of main memory, Series 32000's sophisticated architecture was deliberately designed to support the implementation of virtual memory. This memory strategy uses a secondary storage device, such as the usually standard hard disk, as an adjunct to the main memory under the control of an operating system.

2.1 In a virtual-memory system, every address used by the CPU is called a *virtual address* and each virtual address is subject to dynamic address translation[4]—a mapping function that translates a virtual address in virtual memory into a *physical address* in main memory.

All virtual addresses are translated into physical addresses by a common formula, and are all given access protection. If a datum is not in main memory when an instruction calls for it, the address translation subsystem must so inform the CPU by requesting an *instruction abort*. The CPU then aborts the current instruction, and transfers control to an operating system routine that interprets the cause of the abort. If the abort was caused by a reference to a location available only in secondary storage, the CPU first transfers the contents of the page containing the location from secondary storage to main memory, and then retries the aborted instruction. This translation process is called *swapping*.

If this process can be done automatically, then as far as the user is concerned the combination of main memory and secondary storage becomes one immense, contiguous memory resource. Consequently, the user is able to take full advantage of extremely large operating-system software and applications programs without worrying about the actual hardware limitations of the system.

2.2 From the standpoint of system designs and manufacturing, the price/performance tradeoffs that virtual memory affords are particularly important, and readily apparent.

For example, a small business system may need several million bytes of memory in order to run word processing, financial planning, and data-base management programs efficiently. Yet a manufacturer may wish to ship only 1 Mbyte of main memory in order to keep the price of the system competitive. Since this hypothetical system probably includes a hard disk as standard equipment, implementing virtual memory is a viable technique for expanding the address space. The result is that the user *sees* a system which performs like one with, say 5 Mbytes of main memory, yet *pays* for only 1 Mbyte.

It must be emphasized that for a memory management strategy to truly implement virtual memory, the user must be presented with the illusion that all of the addressable memory is available for use at any given time. To the extent that the user is aware of the memory-management strategy, the benefit of "virtual-ness" disappears.

2.3 The first wave of 16-bit microprocessors were not designed with virtual memory in mind.

As a result, designers must overcome a great many obstacles, to adapt earlier designs for use in virtual memory systems.

Early 68000-based virtual memory-management systems, for example, required *two* 68000s. One executed the programs, but any time it needed to access a new virtual address, the other had to stop the first and repair the "page fault".

The designers of the 68010, which attempts to support virtual memory with a single CPU, were forced to interrupt the internal microcode machine between two microcycles of the executing instruction, and to save 26 words of "invisible" internal state on the external stack before freeing the CPU to perform the memory-management tasks.

Series 32000 eliminates the need for any such "kludges."

[4] Also called *memory mapping*, or *address relocation*.

Chapter 3
The two predominant approaches to virtual memory-management: *segmentation* and *demand paging*.

Virtual memory works because programs, instead of addressing locations in memory at random, tend to stay localized for long periods. If a system has enough main memory to hold these "locales," a program will run smoothly...for a while.

Eventually, though, a program will call for data not present in main memory. The CPU must then obtain that data by swapping one of the current physical-memory regions of data with the desired externally stored region. This swapping task must be accomplished as quickly as possible—preferably "on demand"—to carry out a program instruction. However, the swapped region should contain as little unusable data as possible.

3.1 In a uniform memory space, the size of the swap is based on a fixed-sized unit called a *page,* which can be any size the system designer specifies. In a segmented memory system, the size of the swap is determined by the size of the segment, which varies.

Segmentation is a comparatively simple form of memory management. Most segmented systems have few segments. Each segment must therefore cover a large portion of the virtual address space. As large segments get swapped in and out of main memory, the available memory space tends to become "fragmented" into many small pieces, until not enough contiguous main memory is available to contain one large segment. To avoid this problem, and eliminate the wasteful allocation of large portions of main memory to unused virtual addresses, segments are generally allowed to be of variable size.

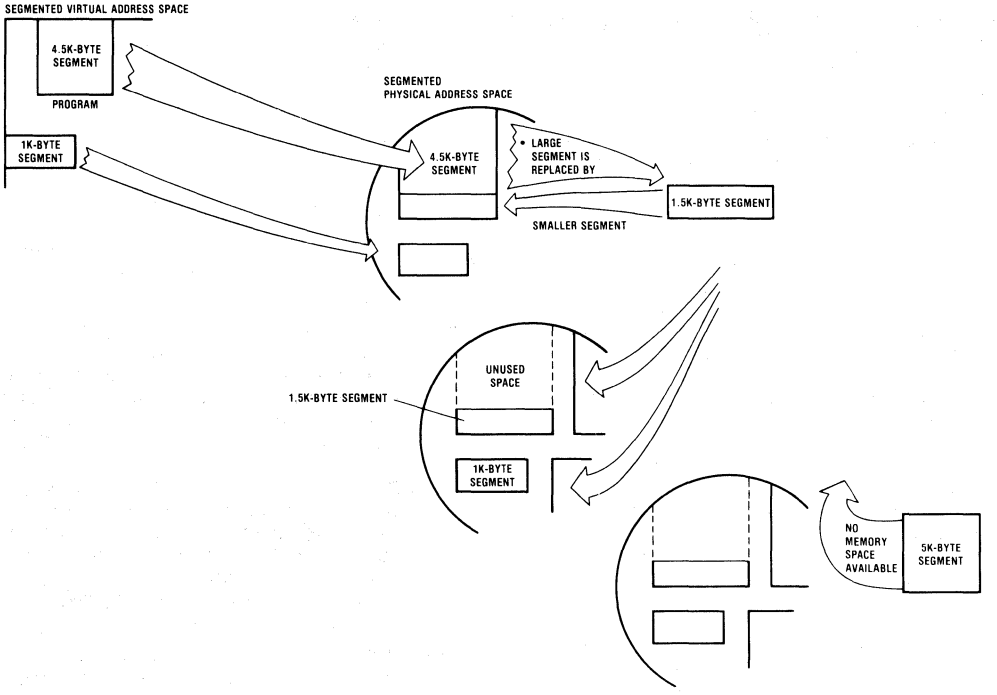
Specific segments of virtual address space are generally associated with specific aspects of a running task. In an 80286, the four segments are "hard-wired" to be a code segment, a stack segment, a data segment, and an alternate data segment. This approach allows swapping and access protection in minimal hardware, but creates havoc in a virtual memory-management system because the segments coincide with the pages.

Because of the variable size of segments and their association with a running task, the implementation of segmented virtual-memory management systems is difficult. No matter how much, or how little, of each segment is needed, all of it must be swapped; a part of a segment cannot be swapped independently *(Figure 2)*.

3.2 In a Demand Paged Virtual Memory system, on the other hand, there is no need for the virtual memory-management strategy to deal with large, odd-sized blocks of main memory, or to take into account any information about how any memory will be used by the running task.

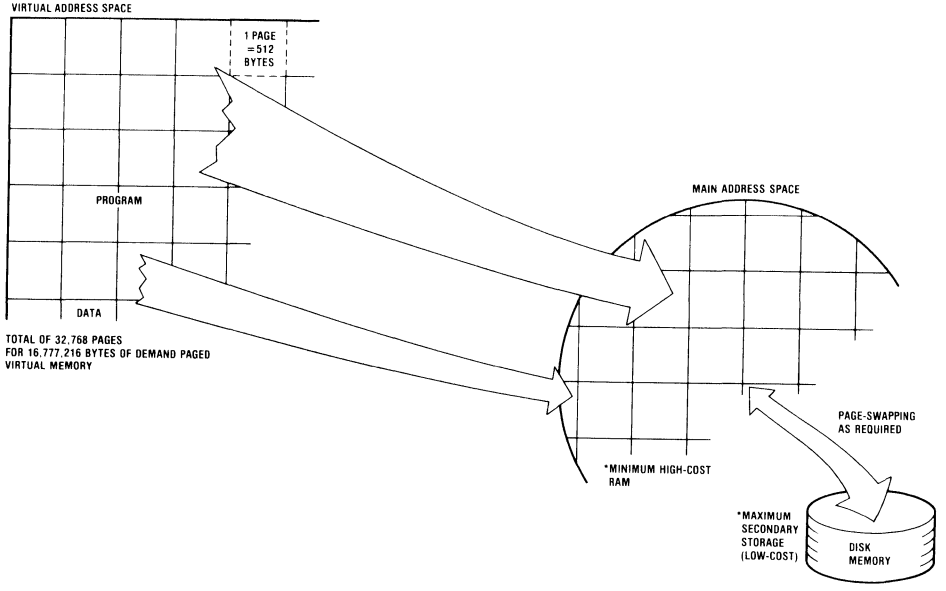
Series 32000 does provide the protection advantages of segmentation, without the segment-size disadvantages, by permitting "segments" to be constructed out of an arbitrary number of fixed-size pages that can be independently swapped. But Series 32000 does not *require* segmentation as a built-in feature of its architecture.

The formidable advantage of demand paging over segmentation, in general, is the simplicity with which pages can be swapped in and out of main memory. The result is a particularly low-overhead memory allocation algorithm that improves system performance *(Figure 3)*.



TL/EE/8666-2

Figure 2: Fragmentation in segmented address space schemes



TL/EE/8666-3

Figure 3: Demand Paged Virtual Memory

Chapter 4
In Series 32000, dynamic address translation is performed automatically by the NS32082 Memory Management Unit (MMU).

The NS32082 MMU is an auxiliary (slave) processing unit that operates in coordination with the master Series 32000 CPUs. Communication between the two units takes place by means of a very fast, well-defined, self-contained protocol that is transparent to the user.[5] Together, the CPU and MMU allow architectural capabilities that otherwise, because of the limitations of contemporary integration technology, would be impossible on a single chip *(Figure 4)*.

The MMU provides a Series 32000-based system with dynamic address translation[6], memory management, memory protection, and both hardware and software debugging support.

4.1 Series 32000 has a virtual address space of 16 Mbytes divided into 32,768 pages, each with a fixed size of 512 bytes. The physical address space is the same size, and is also divided into similarly-sized pages.

This scheme is an ideal one for managing a virtual-memory system because the fixed-sized pages are easy to swap via disk: page numbers can map directly to disk sectors, and paging is transparent to the user-program. Studies have also shown that pages of approximately this size exhibit a good tradeoff between "locality" and "graininess."

4.2 In dynamic address translation, the MMU keeps track of each virtual address requested by the CPU and its corresponding value in main memory at all times.

To do this, it uses two levels of page tables containing pointers that indicate where to go in main memory.

Among its set of registers, the MMU contains two Page Table Base registers: PTB_0 in System mode, and PTB_1 in User mode. (A system program can force the MMU to use PTB_0 in both modes, if desired.) Either register's contents points to a location in main memory that holds a page table. (It is the job of the operating system to load the PTB registers and build the corresponding table.)

The total size of the page table that each PTB register points to is 1,024 bytes, divided into 256 entries, each 32 bits wide. Each one of the 256 entries in each *page* table points to a *pointer* table.

Each pointer table is divided into 128 entries, each 32 bits wide: thus, each pointer table fits on a 512-byte page. Each of the 128 entries in a pointer table points to a 512-byte page in virtual memory.

4.3 Surprisingly, the page and pointer tables do not require large amounts of memory. Practically speaking, each program or task can have its own page table.

An entire 16-Mbyte virtual memory map will use only one 1024-byte page table (resident in main memory) to point to 256 pointer tables, of 512 bytes each, for a maximum of 132,096 bytes devoted to mapping. (The pointer tables need not be in main memory—they can be brought in from secondary storage on demand.)

Changing the page table is simply a matter of changing the MMU register that points to the location in memory that holds the current page table. Therefore, each program or task can have its own map from virtual memory to main memory, and its own virtual address space of 16 Mbytes *(Figure 5)*.

4.4 Each entry in a page table, or one of the pointer tables, has the same basic format.

Bits 9 through 23 specify the starting physical address of the specified page. Bits zero through four contain the following status bits:

0 The Valid bit (V)—indicates whether the entry may be used for address translation.

1–2 The Protection Level field (PL)—indicates the level of protection provided for the page.

3 The Referenced bit (R)—indicates whether the page has been accessed. (It is automatically set when the corresponding page has been accessed for reading or writing.)

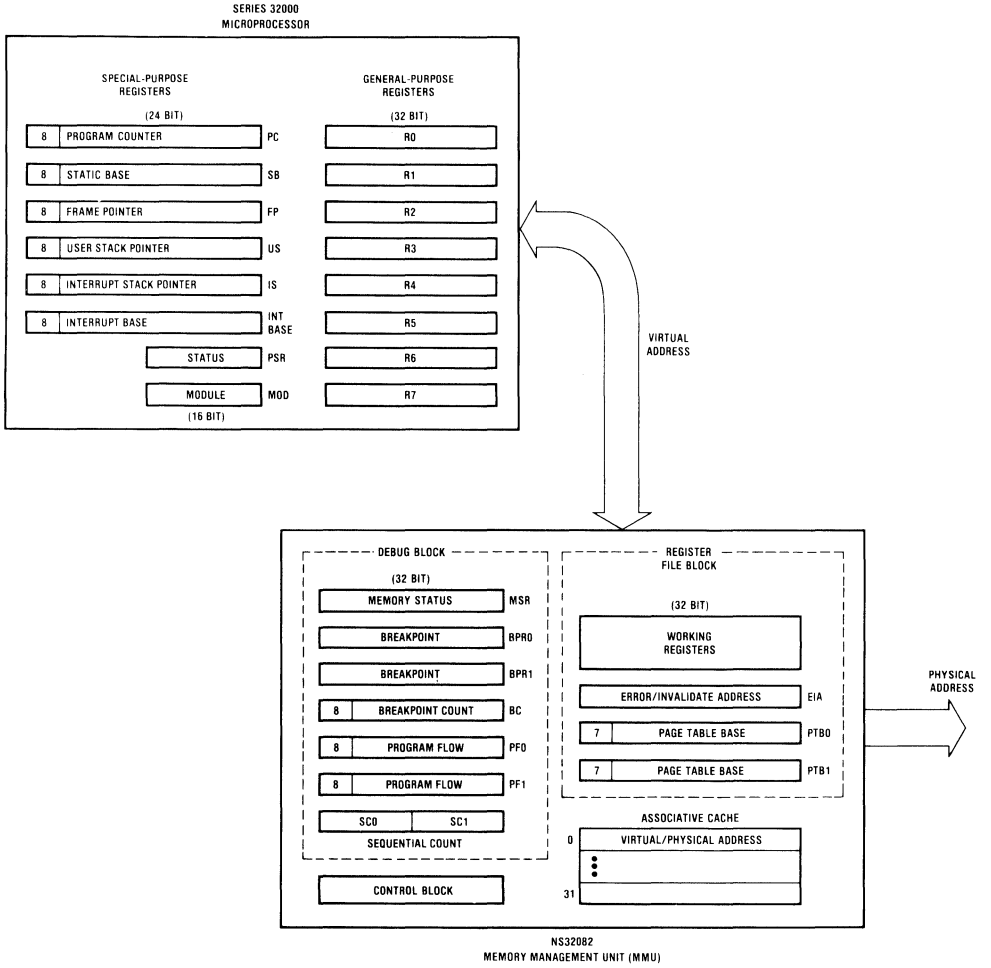
4 The Modified bit (M)—indicates whether the page has been modified during its residence in main memory. (It is automatically set when the corresponding page is written to.)

4.5 To implement dynamic address translation, the page selector field of an address is used to index the page and pointer tables.

The 24 bits of a virtual address may be thought of as consisting of two fields: the page selector field (the high-order fifteen bits), and the offset field (the lower nine bits).

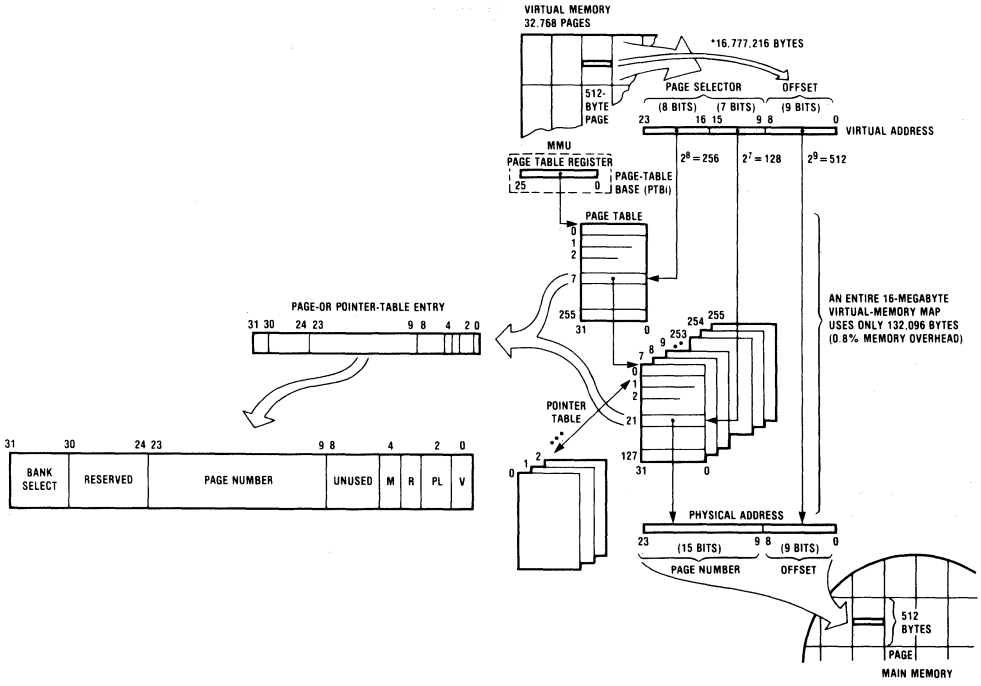
[5] Because the operation of the NS32082 MMU is controlled by these in-line instructions, integrating the MMU's memory management capabilities with any future CPU in Series 32000 will entail only a few, localized software modifications.

[6] See section 2.1.



TL/EE/8666-4

Figure 4: Series 32000 CPU and MMU registers



TL/EE/8666-5

Figure 5: Dynamic address translation

The fifteen-bit page selector field is again divided into subfields: the upper eight bits, which index an entry in the page table, and the lower seven bits, which index an entry in the appropriate pointer table.

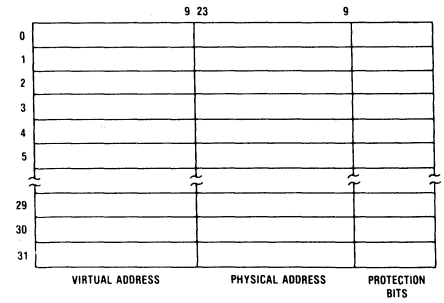
The lower nine bits of the virtual address become the lower nine bits of the physical address, and index the location of a byte within a page.

The result? The 24-bit *virtual* address becomes a 24-bit *physical* address, which is the address actually used to refer to memory.

4.6 Because the mapping tables are too large to store in the MMU, they must be stored in main memory. Reading a table entry from memory would normally take at least two memory accesses per memory access generated by a program—a clearly unacceptable delay.

To speed up the process of dynamic address translation, the NS32082 MMU utilizes an associative on-chip translation cache.

The cache contains the 32 most recently accessed virtual addresses and their translated physical addresses *(Figure 6)*.



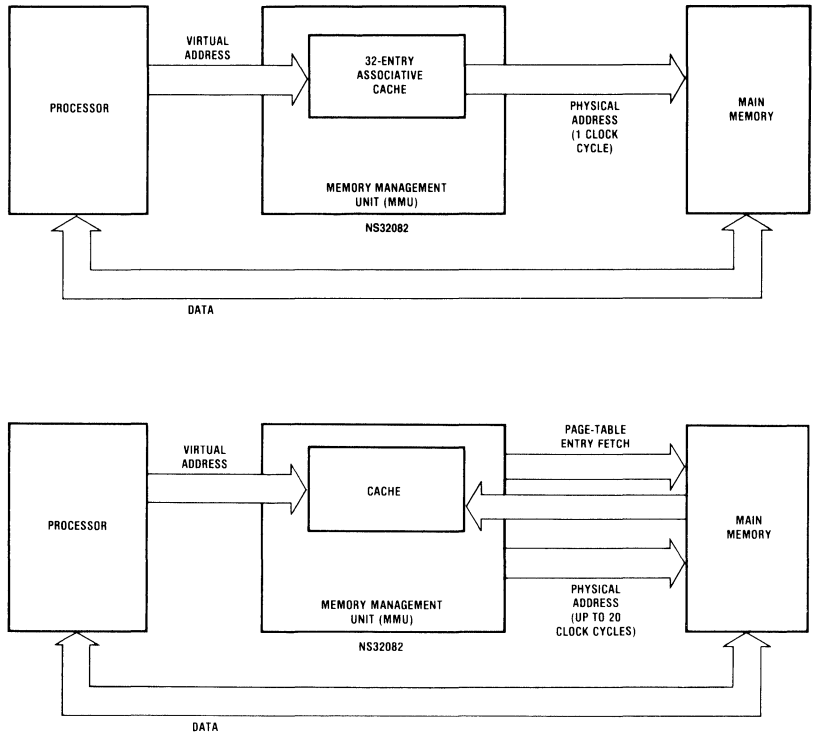
TL/EE/8666-6

Figure 6: Associate cache

When the CPU passes a virtual address to the MMU, the MMU first attempts to match the virtual address with an entry in the cache. If the address requested by the CPU matches one of the 32 cache entries, the MMU will then check the protection level and, if access is permitted, immediately make the physical address available for memory

reference. This virtual-to-physical address translation can occur in just one clock cycle (100 nanoseconds with a 10-MHz microprocessor clock).

If, however, the requested address is not present in the cache, the MMU must fetch both page and pointer table entries from memory before address translation can be performed—a process that may take an average of twenty clock cycles (2 microseconds with a 10-MHz microprocessor clock) *(Figure 7)*.



TL/EE/8666-7

Figure 7: Memory access through the NS32082 MMU

4.7 The NS32082 MMU's net performance depends directly on the frequency with which the cache contains the necessary table entries: observations show that in typical programs the cache contains the entries over 98 percent of the time

Therefore, the operation of the MMU is transparent to the user-program.

When the MMU's cache is full, room must be made for a newly translated page address. To do this efficiently, however, requires an algorithm that decides which is the best page address to remove, in the sense that its absence will have the smallest adverse effect on the running program. The NS32082 MMU uses what is called the Least Recently Used (LRU) algorithm: it evicts the least recently accessed page address in the cache on the basis of that page having a high *a priori* probability of not being in the running program's working set.

The MMU's ability to achieve a 98 percent "hit rate" (comparable to that of the VAX-11 ™) is directly related to its use of this very fast algorithm.

4.8 The powerful Referenced bit (R), in conjunction with the Modified bit (M) in the table entries, also directly influence the MMU's net performance.

The *Referenced bit* is set by the hardware when the page is referenced (read or written) by an instruction. By periodically checking and clearing this bit in all page and pointer table entries, the operating system can monitor the frequency with which pages are being used, and select pages to be swapped out according to the LRU algorithm. If a page about to be swapped out has not been modified since it was read in (a likely occurrence if the page contains code rather than data), it is unnecessary to write it back into secondary memory, since an accurate copy already exists there. If, however, the page has been modified since being read in, the copy in secondary storage is no longer accurate and must be rewritten.

The *Modified bit* is set to "1" by the hardware whenever a page is written to while resident in main memory. When the page is to be released, the operating system can check this bit to see if the copy on disk must be updated. If the bit is "1," the page must be written to secondary storage; if it is "0," then the page has not been modified since it was read in, and can simply be overlaid.

4.9 Two other MMU registers facilitate the interaction of the MMU, CPU, and operating system.

The *Error/Invalidate Address* (EIA) register provides a "window" between the MMU cache and the CPU that allows the software to remove an entry from the cache. (If it were changed in memory alone, the MMU would continue to use the old value stored in the cache.) The EIA register also returns the address that caused an MMU exception, for use in case of error or a required virtual-memory swap.

The EIA's high-order bit indicates which PTB is being used for translation. Changing a PTB value automatically removes all cache entries based on that PTB.

The *Memory Status Register* (MSR) holds the many indicators that allow software to monitor and control the MMU's actions. It is accessible only in the Supervisor mode.

4.10 The NS32082 MMU can abort an instruction during execution by the CPU—to load a page from secondary storage into main memory—and immediately retry the instruction.

This feature is unique to Series 32000 and is fully implemented in hardware. No complex restart routine or externally saved internal state is required[7].

After fetching and decoding an instruction, the CPU sends the virtual address of the operand to the MMU. The *Valid bit* (V) in a page or pointer table entry indicates whether or not the corresponding page is present in main memory. Whenever an address is generated by the CPU, and passed to the MMU for translation into a physical address, the MMU checks the Valid bit of the table entry specified by the incoming virtual address. If the Valid bit is "1," the page is assumed to be present in main memory, and address translation proceeds directly.

If the Valid bit is "0," the page is assumed not to be in main memory, and a *page fault* occurs. A page fault is a hardware-generated trap that is used to tell the operating system to read the missing page in from secondary storage. The page fault occurs in the MMU, which generates an ABORT signal to the CPU that immediately halts execution of the current instruction. (A memory-access abort will also occur if the CPU tries to access a protected section of memory.)

One of the problems in implementing virtual memory systems is that an instruction may generate a page fault at any time during the course of its execution. If the instruction occupies several bytes, it may overlap a page boundary, and the act of fetching an instruction may itself cause a page fault. The process of fetching the source or destination operand may cause a page fault as well.

In Series 32000, when a page fault occurs, for *any* reason, the MMU sends the ABORT signal to the CPU. To permit the instruction to be *restarted*, the CPU not only halts the execution of the instruction, it also returns any register that was altered by the instruction to the state it was in before the aborted instruction began. At the same time, the program counter is automatically saved, as are the processor-status register and the stack pointer, among other registers, so that, as soon as the operating system completes the page swap, the CPU automatically retries the aborted instruction.

4.11 The exception to this process is in the case of *string instructions*, which get special treatment during an abort.

Since it would be extremely undesirable to have a long string instruction repeated from the beginning if an abort occurred in the middle of the string, Series 32000 CPUs allow an aborted string instruction to be re-executed from the point at which the page fault occurred.

[7] Motorola's 68451 MMU, for example, which also has 32 on-chip translation registers, must interrupt the 68010 CPU if the required informationis not in one of its registers—a procedure that can easily take 50 to 100 times the less-than-2μs required by the NS32082 MMU.

Chapter 5
In a Series 32000-based system, each page can have protection attributes to limit the ways in which the page can be accessed. This provides the basis for memory protection within the virtual address space.

The protection features actually implemented in Series 32000 architecture can be divided into three groups:

1. *5.1 Supervisor/User mode.* The CPU has two operating modes: Supervisor mode, in which the entire instruction set is available, and User mode, in which only a restricted subset of instructions are available. Supervisor mode is intended for operating systems and other trusted programs, User mode for programs that are not trusted.
2. *5.2 Separate address spaces for each task.* Each task running on a Series 32000-based system has a unique collection of pages that constitutes its address space: access to another task's address space is impossible.
3. *5.3 Protection bits along with the physical addresses in the page- and pointer-table entries.*

To keep order in today's multi-tasking, multi-user, and multi-processor systems, the protection bits define whether a page can be read, but not written into; read and written into; or, neither read nor written. How the protection bits are interpreted depends on the operating mode of the CPU: a given setting of the Protection Level (PL) field will be interpreted differently in Supervisor mode than in User mode, as shown below.

PL	SUPERVISOR MODE	USER MODE
00	read only	no access
01	read/write	no access
10	read/write	read only
11	read/write	read/write

As a result, the operating system can treat a collection of pages with the same protection level as a segment. For example, a constants segment might be a set of pages containing data with the read-only protection level, so users could not modify the data. In this way, page-based dynamic address translation provides a mechanism for implementing segmentation.

Inter-task protection is accomplished by giving each task its own set of page tables, so that each task has its own address space, which provides flexibility and virtual memory for each task. By changing the single register that points to the page table, the user can switch to the new task's address space.

Chapter 6
At best, debugging is one of the most difficult stages of program development. In a virtual memory system, it could be a nightmare.

The debugging facility provided by the NS32082 MMU is unique and unsurpassed among microprocessor families. It would prove invaluable even if the MMU did nothing else but provide for the two crucial, program-debugging operations: *breakpointing*, and *flow tracing*.

6.1 In most other microprocessors, breakpointing is provided by a trap, or breakpoint instruction.

This can result in a myriad of problems for a virtual memory system. It is often necessary, when debugging Supervisor-mode software, to be able to set breakpoints as absolute addresses, i.e., as addresses in physical memory. This is not possible with CPU-based debugging techniques, since all addresses emanating from the CPU are virtual addresses—the CPU has no concept of the distinction between the two types of addresses. Moreover, the setting of breakpoints with special instructions that overlay existing code can cause a great deal of additional overhead for the memory manager.

6.2 For these reasons, among others, the designers of Series 32000 chose to implement debugging support on the MMU.

To implement breakpointing and flow tracing, the NS32082 MMU uses two sets of registers—the Breakpoint and Flow Tracing registers—and one instruction, the Breakpoint Trap instruction.

6.3 Setting breakpoints is a technique for halting a program's execution at a particular instruction or data access for the purpose of examining the program's state, and thereby determining the cause of improper program behavior.

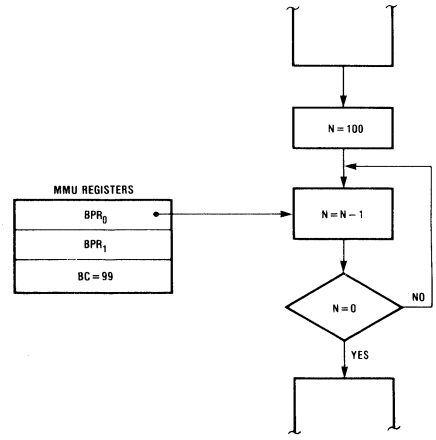
With Series 32000, breakpoints may be set either when a specified address is accessed, after a specified address ns accessed, or after a specified number of such accesses have been made. Also, more than one breakpoint address may be selected simultaneously, allowing a halt to be implemented after either fork of a conditional branch. These facilities are provided by the Breakpoint Trap instruction (BPT) and three dedicated registers located on the MMU.

The Breakpoint Trap instruction is a one-byte instruction that replaces the first byte of the opcode of the instruction that is to be breakpointed. To allow breakpoints to be set in PROM, as well as RAM, two breakpoint registers, BPR_0 and BPR_1 are provided. These registers hold the double word addresses of two selected breakpoints, which are compared with the contents of the address bus at every memory cycle. When a breakpoint address appears in the program, and when other conditions specified by the contents of the register are met, a non-maskable interrupt occurs.

Because these registers are located in the MMU, they may be set to look at either the virtual addresses from the CPU or the physical addresses from the MMU. They may also be set to operate when the indicated address is either written to or read from, or when there is an instruction fetch.

A third register on the MMU, the Breakpoint Count register, specifies the number of matches of the BPR_0 register breakpoint condition to be passed over before a breakpoint occurs. This is useful for selecting a particular interaction in a loop instruction.

The breakpointing process occurs parallel to the execution of the running program, and exacts a negligible performance penalty. Consider how a programmer might want to use software breakpointing or tracing to debug a virtual-memory system in which the memory area where the Breakpoint Trap instruction is located might have been swapped out onto disk. This task would be all but impossible without hardware support such as that provided in the NS32082 MMU *(Figure 8)*.



TL/EE/8666-8

Figure 8: Breakpointing

6.4 Flow tracing provides a chronicle of the actions and results of individual steps in a program during its execution.

It allows the program's recent history to be examined at specified instructions or breakpoints in order to determine the cause of any undesired program behavior.

Series 32000 supports program flow tracing with four dedicated registers in the MMU, and the Trace trap bit in the CPU's PSR. If flow tracing is activated (by means of a bit in the MSR), two Program Flow registers (PF_0 and PF_1) will always hold the addresses of the last two instructions which were executed out of sequence. The two 16-bit Sequence Count registers (SC_0 and SC_1) will keep a record of the number of sequential instructions executed between each change in program flow.

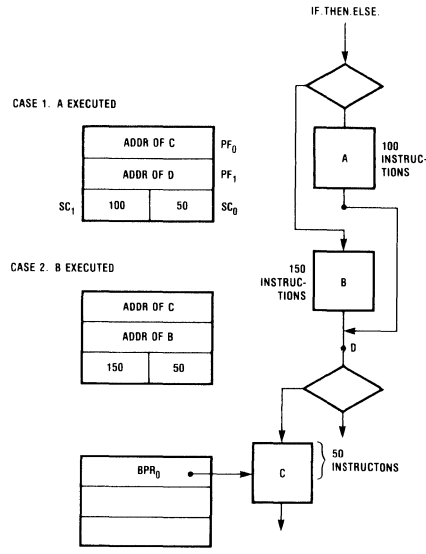
The MMU thus performs the following steps every time a branch, call, return, interrupt, or other non-sequential instruction is executed:

—Store PF_0 into PF_1
—Store new program-counter value into PF_0
—Store SC_0 into SC_1
—Clear SC_0

6.5 The user can also select an instruction, or series of instructions, to trace by means of the Trace trap, which is enabled by setting a bit in the PSR.

When the Trace trap is enabled at the beginning of an instruction, a trace trap will occur at the end of that instruction, and user software may then be employed to investigate the contents of the CPU registers. The trap will occur after each instruction, so long as the bit is set.

In Series 32000, the Trace trap is implemented in such a way that one and only one Trace trap will be taken for each instruction. The Trace trap always has the lowest priority of any exception. Any other trap, or any interrupt request that occurs during a traced instruction, is allowed to complete its entire service procedure before the Trace trap occurs. Also, unlike other traps, where the address of the first byte of the instruction during which the trap occurred is pushed on the stack, the Trace trap insures that the return address to be pushed is that of the next instruction to be traced *(Figure 9)*.

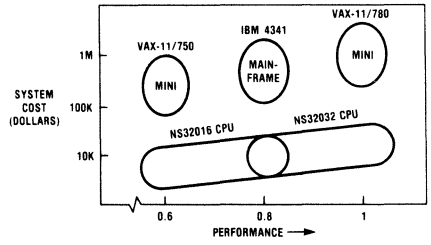


TL/EE/8666-9

Figure 9: Flow tracing

Chapter 7
Conclusion.

The simplicity and efficiency of Demand Paged Virtual Memory, as implemented in Series 32000, offers features formerly available only in much larger systems—but in a combination not available on any one system, nor anywhere near microprocessor prices *(Figure 10)*.



TL/EE/8666-10

Figure 10: Comparison, Series 32000 to minis and mainframes

SOURCES

"Demand Paged Virtual Memory with Series 32000," Richard Mateosian.

"Extend μP capabilities with a memory-management IC," Robert D. Grappel (Hemenway Corp.). *EDN*, February 3, 1983.

"Introduction to Series 32000 Architecture."

"Virtual Memory," Carol Anne Ogdin.

"Virtual memory in microprocessor systems," Subhash Bal and Gary Martin. *Electronic Products* Magazine, September 30, 1982.

Section 2

Central Processing Units

Section 2 Contents

NS32332-10, -12, -15 32-Bit Advanced Microprocessor with Virtual Memory 2-3
NS32132-6, -8, -10 High-Performance Microprocessors 2-76
NS32032-6, -8, -10 High-Performance Microprocessors 2-146
NS32C016-6, -10, -15 High-Performance Microprocessors 2-211
NS32016-6, -8, -10 High-Performance Microprocessors 2-275
NS32008-6, -8, -10 High-Performance Microprocessors 2-339

National Semiconductor Corporation

PRELIMINARY

NS32332-10/NS32332-12/NS32332-15
32-Bit Advanced Microprocessor with Virtual Memory

General Description

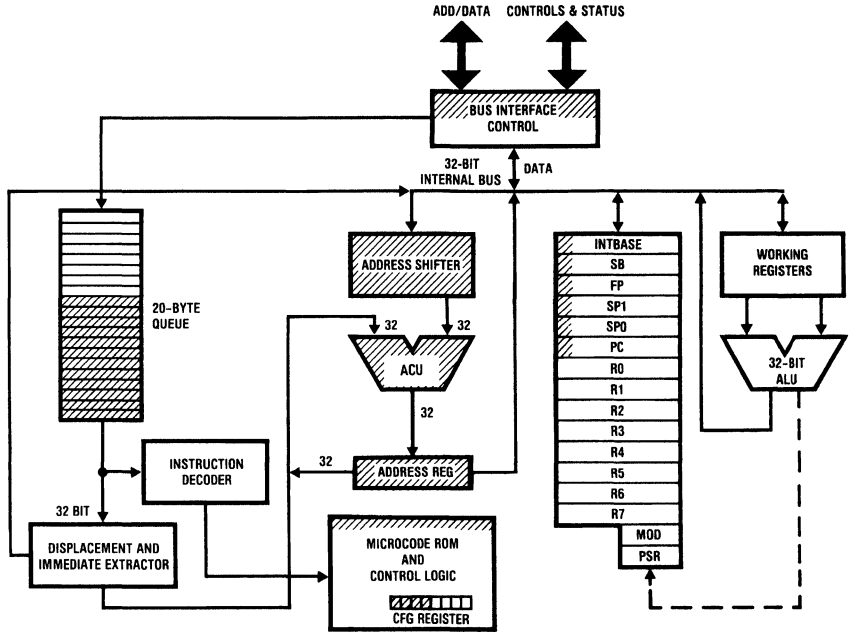
The NS32332 is a 32-bit, virtual memory microprocessor with 4 GByte addressing and an enhanced Series 32000® microarchitecture. It is fully object code compatible with other Series 32000 microprocessors, and it has the added features of 32-bit addressing, higher instruction execution throughput, cache support, and expanded bus handling capabilities. The new bus features include bus error and retry support, dynamic bus sizing, burst mode memory accessing, and enhanced slave processor communication protocol. The higher clock frequency and added features of the NS32332 enable it to deliver 2 to 3 times the performance of the NS32032.

The NS32332 microprocessor is designed to work with both the 16- and 32-bit slave processors of the Series 32000 family.

Features

- 32-bit architecture and implementation
- 4 Gbyte uniform addressing space
- Software compatible with the Series 32000 Family
- Powerful instruction set
  - — General 2-address capability
  - — Very high degree of symmetry
  - — Address modes optimized for high level languages
- Supports both 16- and 32-bit Slave Processor Protocol
  - — Memory management support via NS32082 or NS32C382
  - — Floating point support via NS32081 or NS32310
- Extensive bus feature
  - — Burst mode memory accessing
  - — Cache memory support
  - — Dynamic bus configuration (8-, 16-, 32-bits)
  - — Fast bus protocol
- High speed XMOS™ technology
- 84 Pin grid array package

Block Diagram



TL/EE/8673-1

FIGURE 1

*Shaded areas indicate enhancements from the NS32032.

Table of Contents

1.0 PRODUCT INTRODUCTION

1.1 NS32332 Key Features

2.0 ARCHITECTURAL DESCRIPTION

2.1 Programming Model

2.1.1 General Purpose Registers
2.1.2 Dedicated Registers
2.1.3 The Configuration Register (CFG)
2.1.4 Memory Organization
2.1.5 Dedicated Tables

2.2 Instruction Set

2.2.1 General Instruction Format
2.2.2 Addressing Modes
2.2.3 Instruction Set Summary

3.0 FUNCTIONAL DESCRIPTION

3.1 Power and Grounding
3.3 Clocking
3.3 Resetting
3.4 Bus Cycles

3.4.1 Cycle Extension
3.4.2 Burst Cycles
3.4.3 Bus Status
3.4.4 Data Access Sequences
3.4.4.1 Bit Accesses
3.4.4.2 Bit Field Accesses
3.4.4.3 Extending Multiple Accesses
3.4.5 Instruction Fetches
3.4.6 Interrupt Control Cycles
3.4.7 Dynamic Bus Configuration
3.4.8 Bus Exceptions
3.4.8.1 Bus Retry
3.4.8.2 Bus Error
3.4.8.3 Fatal Bus Error
3.4.9 Slave Processor Communication
3.4.9.1 Slave Processor Bus Cycles
3.4.9.2 Slave Operand Transfer Sequence

3.5 Memory Management Option

3.5.1 The FLT (Float) Pin
3.5.2 Aborting Bus Cycles
3.5.2.1 The Abort Interrupt
3.5.2.2 Hardware Considerations

3.0 FUNCTIONAL DESCRIPTION (Continued)

3.6 Bus Access Control
3.7 Instruction Status
3.8 NS32332 Interrupt Structure

3.8.1 General Interrupt/Trap Sequence
3.8.2 Interrupt/Trap Return
3.8.3 Maskable Interrupts (The INT Pin)
3.8.3.1 Non-Vectored Mode
3.8.3.2 Vectored Mode: Non-Cascaded Case
3.8.3.3 Vectored Mode: Cascaded Case
3.8.4 Non-Maskable Interrupt (The NMI Pin)
3.8.5 Traps
3.8.6 Prioritization
3.8.7 Interrupt/Trap Sequences: Detailed Flow
3.8.7.1 Maskable/Non-Maskable Interrupt Sequence
3.8.7.2 Trap Sequence: Traps Other than Trace
3.8.7.3 Trace Trap Sequence
3.8.7.4 Abort Sequence

3.9 Slave Processor Instructions

3.9.1 16-Bit Slave Processor Protocol
3.9.2 32-Bit Fast Slave Protocol
3.9.3 Floating Point Instructions
3.9.4 Memory Management Instructions
3.9.5 Custom Slave Instructions

4.0 DEVICE SPECIFICATIONS

4.1 Pin Descriptions
4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics

4.4.1 Definitions
4.4.2 Timing Tables
4.4.2.1 Output Signals: Internal Propagation Delays
4.4.2.2 Clocking Requirements
4.4.2.3 Input Signal Requirements
4.4.3 Timing Diagrams

Appendix A: Instruction Formats
B: Interfacing Suggestions

List of Illustrations

CPU Block Diagram 1

The General and Dedicated Registers 2-1
Processor Status Register 2-2
CFG Register 2-3
Module Descriptor Format 2-4
A Sample Link Table 2-5
General Instruction Format 2-6
Index Byte Format 2-7
Displacement Encodings 2-8

Recommended Supply Connections 3-1
Clock Timing Relationships 3-2
Power-on Reset Requirements 3-3
General Reset Timing 3-4
Recommended Reset Connections, Non-Memory Managed System 3-5a
Recommended Reset Connections, Memory Managed System 3-5b
Read-cycle Timing 3-6
Write-cycle Timing 3-7
Bus Connections 3-8
RDY Pin Timing 3-9
Extended Cycle Example 3-10
Burst Cycles; Normal Termination of Burst 3-11a
Burst Cycles; External Termination of Burst 3-11b
$\overline{\text{BOUT}}$ Timing Resulting from a Bus Width Change 3-12
Memory Interface 3-13
Bus Width Changes 3-14
Bus Cycle Retry; Bus Cycle Not Retried 3-15a
Bus Cycle Retry; Bus Cycle Retried 3-15b
Bus Error During Read or Write Cycle 3-16
Slave Processor Connections 3-17
CPU Read from Slave Processor 3-18
CPU Write to Slave Processor 3-19
Read (Write) Cycle with Address Translation 3-20
$\overline{\text{FLT}}$ Timing 3-21
$\overline{\text{HOLD}}$ Timing, Bus Initially Idle 3-22
$\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle 3-23
$\overline{\text{ILO}}$ Timing 3-24
Non-Aligned Write Cycle—$\overline{\text{MC}}$/$\overline{\text{EXS}}$ Timing 3-25
Interrupt Dispatch Table 3-26
Interrupt/Trap Service Routine Calling Sequence 3-27
Return from Trap (RETTn) Instruction Flow 3-28
Return from Interrupt (RETI) Instruction Flow 3-29
Service Sequence 3-30
Slave Processor Protocol 3-31
Fast Slave Protocol 3-32
ID and Opcode Format for Fast Slave Protocol 3-33
Slave Processor Status Word Format 3-34

List of Illustrations (Continued)

Connection Diagram, Pin Grid Array Package 4-1
Timing Specification Standard (Signal Valid After Clock Edge) 4-2
Timing Specification Standard (Signal Valid Before Clock Edge) 4-3
NS32332 Read Cycle Timing 4-4
NS32332 Write Cycle Timing 4-5
NS32332 Burst Cycle Timing 4-6
NS32332 Bus Retry During Normal Bus Cycle 4-7
$\overline{BRT}$ Activated, but No Bus Retry 4-8
Bus Retry During Burst Bus Cycle 4-9
$\overline{BRT}$ Activated During Burst Bus Cycle, but No Bus Retry 4-10
Bus Error During Normal Bus Cycle 4-11
Bus Error During Burst Bus Cycle 4-12
Timing of Interlocked Bus Transactions 4-13
Floating by $\overline{HOLD}$ Timing (CPU not Idle Initially) 4-14
Floating by $\overline{HOLD}$ Timing (CPU Initially Idle) 4-15
Release from $\overline{HOLD}$ 4-16
$\overline{FLT}$ Initiated Cycle Timing 4-17
Release from $\overline{FLT}$ Timing 4-18
Slave Processor Write Timing 4-19
Slave Processor Read Timing 4-20
$\overline{DT}/\overline{SDONE}$ Timing (32-Bit Slave Protocol) 4-21
$\overline{SPC}$ Timing (16-Bit Slave Protocol) 4-22
Clock Waveforms 4-23
Relationship of $\overline{PFS}$ to Clock Cycles 4-24
Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch 4-25
Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$ 4-26
Abort Timing, $\overline{FLT}$ Not Applied 4-27
Abort Timing, $\overline{FLT}$ Applied 4-28
Power-on Reset 4-29
Non-Power-on Reset 4-30
U/S Relationship to Any Bus Cycle, Guaranteed Valid Interval 4-31
$\overline{INT}$ Interrupt Signal Detection 4-32
$\overline{NMI}$ Interrupt Signal Timing 4-33

Processor System Connection Diagram B-1

List of Tables

NS32332 Addressing Modes 2-1
Series 32000 Instruction Set Summary 2-2
Bus Access Types 3-1
Access Sequences 3-2
Interrupt Sequences 3-3

1.0 Product Introduction

The Series 32000 Microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from a minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operations. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes. Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types. The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set. While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations. The Series 32000 CPUs represent two-operand machines with each operand addressable by all addressing modes. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management. Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing. The NS32332 has 32-bit address pointers that can address up to 4 gigabytes without requiring any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support. Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept. The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

1.1 NS32332 KEY FEATURES

The NS32332 is a 32-bit CPU in the Series 32000 family. It is totally software compatible with the NS32032, NS32016, and NS32008 CPUs but with an enhanced internal implementation.

The NS32332 design goals were to achieve two to three times the throughput of the NS32032 and to provide the full 32-bit addressing inherent in the architecture.

The basic approaches to higher throughput were: fewer clock cycles per instruction, better bus use, and higher clock frequency.

An examination of the block diagram of the NS32332 shows it to be identical to that of the NS32032, except for enhanced bus interface control, a 20-byte (rather than 8-byte) instruction prefetch queue, and special hardware in the address unit. The new addressing hardware consists of a high-speed ALU, a barrel shifter on one of its inputs, and an address register. Of the throughput improvement not due to increased clock frequency, about half is derived from the new address unit hardware, about 30% from the bus enhancements, about 15% from the larger prefetch queue, and the rest from microcode improvements.

Other important aspects of the enhanced bus interface circuitry of the NS32332 are a burst access mode, designed to work with nibble and static column RAMs, read and write timing designed to support caches, and support for bus error processing.

An enhanced slave processor communication protocol is designed to achieve improved performance with the NS32382 MMU and NS32310 FPC, while still working directly with the existing NS32082 MMU and NS32081 FPU.

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture has 8 general purpose and 8 dedicated registers. All registers are 32 bits wide except the STATUS and MODULE register. These two registers are each 16 bits wide.

2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

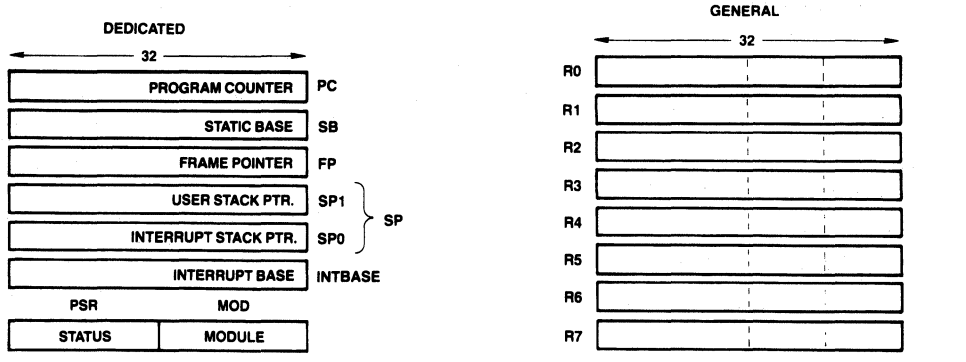
2.1.2 Dedicated Registers

The eight dedicated registers of the processor are assigned specific functions.

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section.

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used

2.0 Architectural Description (Continued)



TL/EE/8673-2

FIGURE 2-1. The General and Dedicated Registers

primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 the SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1.

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.

FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer.

SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module.

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table.

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

PSR: The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.

TL/EE/8673-3

FIGURE 2-2. Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Sec. 3.8.5).

L: The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the processor is said to be in Supervisor Mode; when U = 1 the processor is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

2.0 Architectual Description (Continued)

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Sec. 3.8.5.). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I = 1, then all interrupts will be accepted (Sec. 3.8.). If I = 0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)*

Within the Control section of the CPU is the CFG Register, which declares the presence and type of external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3.*

*The NS32332 CPU has four new bits in the CFG Register, namely P, FC, FM and FF.

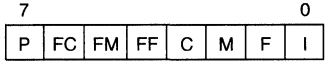


FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{INT}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

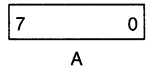
The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

The FF, FM, FC bits define the Slave Communication Protocol to be used in FPU, MMU and Custom Slave instructions (Sec. 3.4.9). If these bits are not set, the corresponding instructions will use the 16-bit protocol (32032 compatible). If these bits are set, the corresponding instructions will use the new (fast) 32-bit protocol.

The P bit improves the efficiency of the Write Validation Buffer in the CPU. It is set if the Virtual Memory has page size(s) larger than or equal to 4 Kbytes. It is reset otherwise. In Systems where the MMU is not present, the P bit is not used.

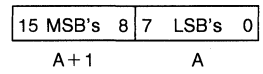
2.1.4 Memory Organization

The main memory is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at 2^{32} - 1. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.



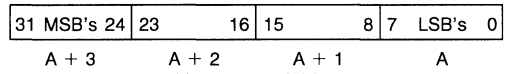
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Sec. 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.



Word at Address A

Two contiguous words are called a double word. Except where noted (Sec. 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



Double Word at Address A

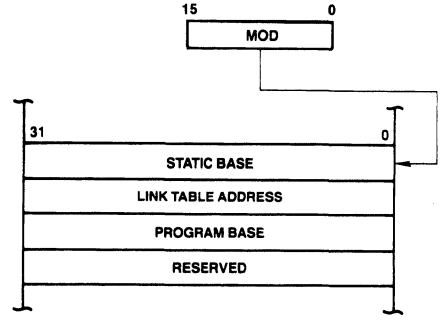
Although memory is addressed as bytes, it is actually organized as double-words. Note that access time to a word or a double-word depends upon its address, e.g. double-words that are aligned to start at addresses that are multiples of four will be accessed more quickly than those not so aligned. This also applies to words that cross a double-word boundary.

2.1.5 Dedicated Tables

Two of the dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically up-dated by the Call External Procedure instructions (CXP and CXPD).



TL/EE/8673–4

FIGURE 2-4. Module Descriptor Format

The format of a Module Descriptor is shown in *Figure 2-4.* The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.

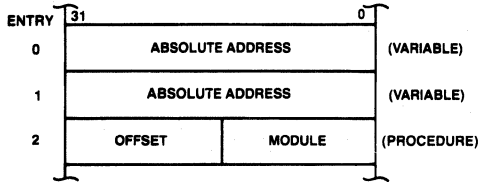
2.0 Architectual Description (Continued)

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-5*. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/8673-5

FIGURE 2-5. A Sample Link Table

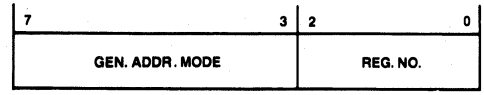
2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-7*.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected address modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded with the top bits of that field, as shown in *Figure 2-8*, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first. Note that this is different from the memory representation of data (Sec. 2.1.4).



TL/EE/8673-7

FIGURE 2-7. Index Byte Format

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Sec. 2.2.3).

2.2.2 Addressing Modes

The CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

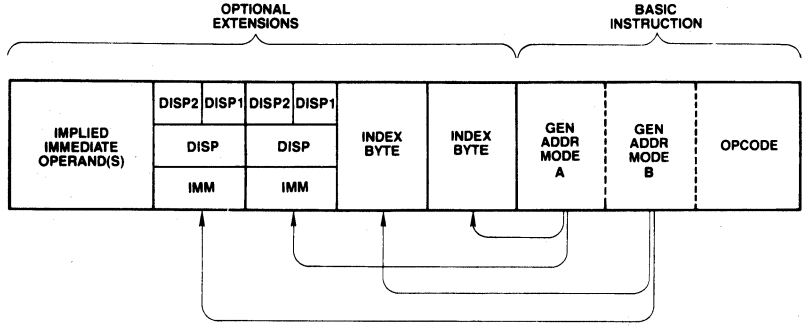
Addressing modes are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

Addressing Modes fall into nine basic types:

Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

Memory Space. Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

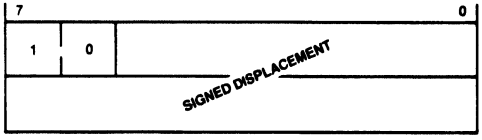


TL/EE/8673-6

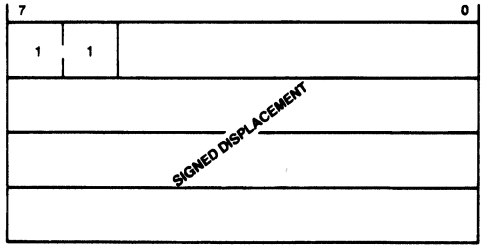
2.0 Architectual Description (Continued)

BYTE DISPLACEMENT: RANGE −64 TO +63

WORD DISPLACEMENT: RANGE −8192 TO +8191



DOUBLE WORD DISPLACEMENT: RANGE $-(2^{29}-2^{24})$ to $+(2^{29}-1)$*



TL/EE/8673-8

FIGURE 2-8. Displacement Encodings

***Note: The pattern "11100000" for the most significant byte of the displacement is reserved by National for future enhancements. Therefore, it should never be used by the user program. This causes the lower limit of the displacement range to be $-(2^{29}-2^{24})$ instead of -2^{29}.**

Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode. Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the Series 32000 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0–R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

2.0 Architectual Description (Continued)

TABLE 2-1
NS32332 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2× Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4× Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn. 'Mode' and 'n' are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

TABLE 2-2
Series 32000 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move Effective Address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDI	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to Extended Integer.
7	DEIi	gen,gen	Divide Extended Integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add Packed.
6	SUBPi	gen,gen	Subtract Packed.

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare Multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical Exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

2.0 Architectural Description (Continued)

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical Shift, left or right.
6	ASHi	gen,gen	Arithmetic Shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to Bit Field Pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 - Comparison Value

R3 - Translation Table Pointer

R2 - String 2 Pointer

R1 - String 1 Pointer

R0 - Limit Count

Options on all string instructions are:

B (Backward): Decrement string pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Descriptions
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare translating, String 1 bytes.
5	SKPSi	options	Skip over String 1 entries
	SKPST	options	Skip, translating bytes for Until/While.

2.0 Architectural Description (Continued)

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor Call.
1	FLAG		Flag Trap.
1	BPT		Breakpoint Trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save General Purpose Registers.
1	RESTORE	[reg list]	Restore General Purpose Registers.
2	LPRi	areg,gen	Load Dedicated Register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store Dedicated Register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust Stack Pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set Configuration Register. (Privileged)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a Floating Point value.
9	MOVLF	gen,gen	Move and shorten a Long value to Standard.
9	MOVFL	gen,gen	Move and lengthen a Standard value to Long.
9	MOVif	gen,gen	Convert any integer to Standard or Long Floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
12	REMf	gen,gen	Remainder.
12	SQRTf	gen,gen	Square Root.
12	POLYf	gen,gen	Polynomial Step.
12	DOTf	gen,gen	Dot Product.
12	SCALBf	gen,gen	Binary Scale.
12	LOGBf	gen,gen	Binary Log.
12	ATAN2f	gen,gen	Arctangent.
12	SICOSf	gen,gen	Sine and Cosine.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

2.0 Architectural Description (Continued)

MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load Memory Management Register. (Privileged)
14	SMR	mreg,gen	Store Memory Management Register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUi	gen,gen	Move a value from Supervisor Space to User Space. (Privileged)
8	MOVUSi	gen,gen	Move a value from User Space to Supervisor Space. (Privileged)

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No Operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom Calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.7	CCAL4c	gen,gen	
15.7	CCAL5c	gen,gen	
15.7	CCAL6c	gen,gen	
15.7	CCAL7c	gen,gen	
15.7	CCAL8c	gen,gen	
15.7	CCAL9c	gen,gen	
15.5	CMOV0c	gen,gen	Custom Move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
15.5	CMOV3c	gen,gen	
15.7	CMOV4c	gen,gen	
15.7	CMOV5c	gen,gen	
15.7	CMOV6c	gen,gen	
15.7	CMOV7c	gen,gen	
15.5	CCMPc	gen,gen	Custom Compare.
15.5	CCMP1c	gen,gen	
15.1	CCV0ci	gen,gen	Custom Convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load Custom Status Register.
15.1	SCSR	gen	Store Custom Status Register.
15.0	CATST0	gen	Custom Address/Test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load Custom Register. (Privileged)
15.0	SCR	creg,gen	Store Custom Register. (Privileged)

3.0 Functional Description

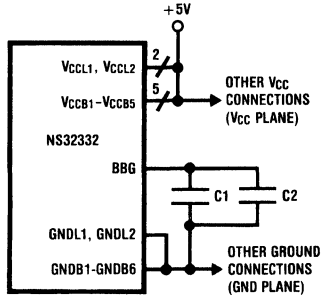
The following is a functional description of the NS32332 CPU.

3.1 POWER AND GROUNDING

The NS32332 requires a single 5-volt power supply, applied on 7 pins. The Logic Voltage pins ($V_{CC}L1$ and $V_{CC}L2$) supply the power to the on-chip logic. The Buffer Voltage pins (V_{CCB1} to V_{CCB5}) supply the power to the output drivers of the chip. The Logic Voltage pins and the Buffer Voltage pins should be connected together by a power (V_{CC}) plane on the printed circuit board.

The NS32332 grounding connections are made on 8 pins. The Logic Ground pins (GNDL1 and GNDL2) are the ground pins for the on-chip logic. The Buffer Ground pins (GNDB1 to GNDB6) are the ground pins for the output drivers of the chip. The Logic Ground pins and the Buffer Ground pins should be connected together by a ground plane on the printed circuit board.

In addition to V_{CC} and Ground, the NS32332 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Figure 3.1*) from the BBG pin to Ground.



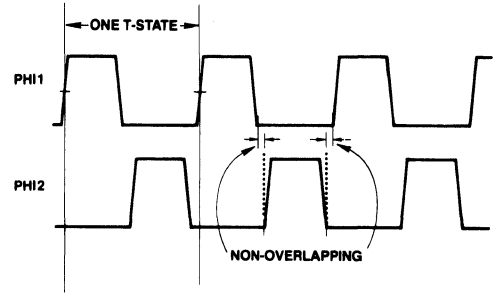
TL/EE/8673-11

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32332 inputs clocking signals from the Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 22) and PHI2 (pin 23). Their relationship to each other is shown in *Figure 3-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See Sec. 4 for complete specifications of PHI1 and PHI2.



TL/EE/8673-9

FIGURE 3-2. Clock Timing Relationships

As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

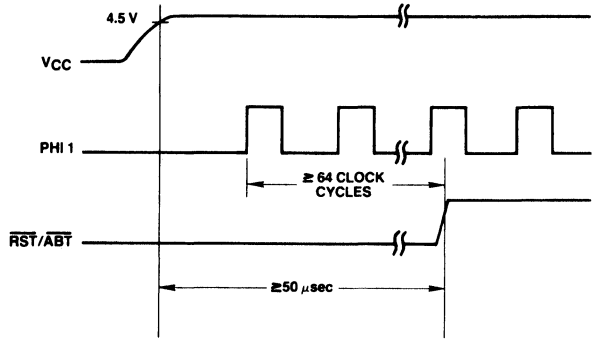
3.3 RESETTING

The $\overline{RST}/\overline{ABT}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.2.

The $\overline{DT}/\overline{SDONE}$ pin is sampled on the rising edge of the reset signal to select the data timing during write cycles. If $\overline{DT}/\overline{SDONE}$ is sampled high, AD0–AD31 are floated during state T2 and the data is output during state T3. This mode must be selected if an MMU is used (Section 3.5). If $\overline{DT}/\overline{SDONE}$ is sampled low, the data is output during state T2. See *Figure 3-7*.

The CPU may be reset at any time by pulling the $\overline{RST}/\overline{ABT}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, $\overline{RST}/\overline{ABT}$ must be held low for at least 50 μsec after V_{CC} is stable. This is to ensure that all



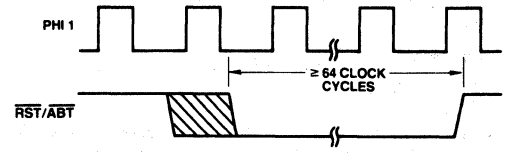
TL/EE/8673-10

FIGURE 3-3. Power-on Reset Requirements

3.0 Functional Description (Continued)

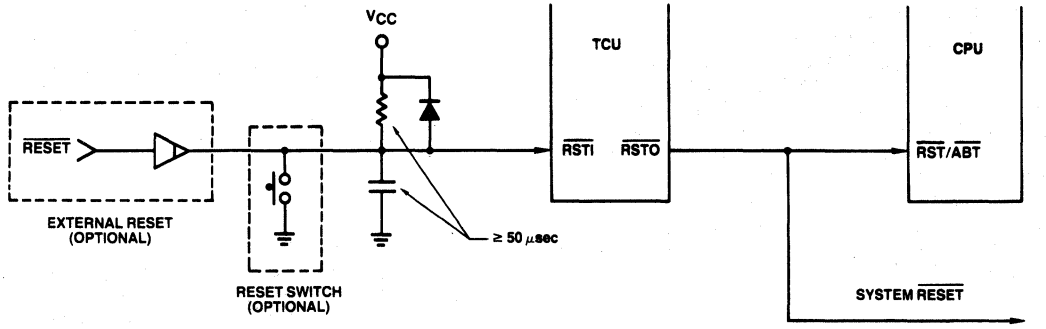
on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4*.

The Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32332 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



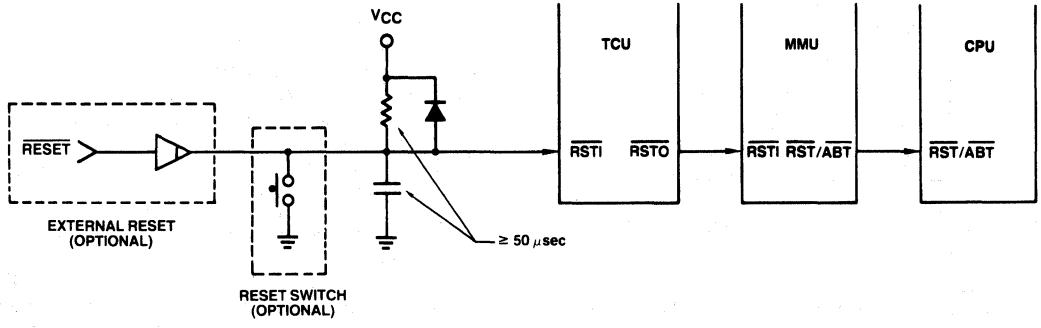
TL/EE/8673–12

FIGURE 3-4. General Reset Timing



TL/EE/8673–13

FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System



TL/EE/8673–14

FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

3.4 BUS CYCLES

The NS32332 CPU will perform Bus cycles for one of the following reasons:

1) To write or read data to or from memory or peripheral interface device. Peripheral input and output are memory mapped in the Series 32000 family.
2) To fetch instructions into the 20-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4) To transfer information to or from a Slave Processor.

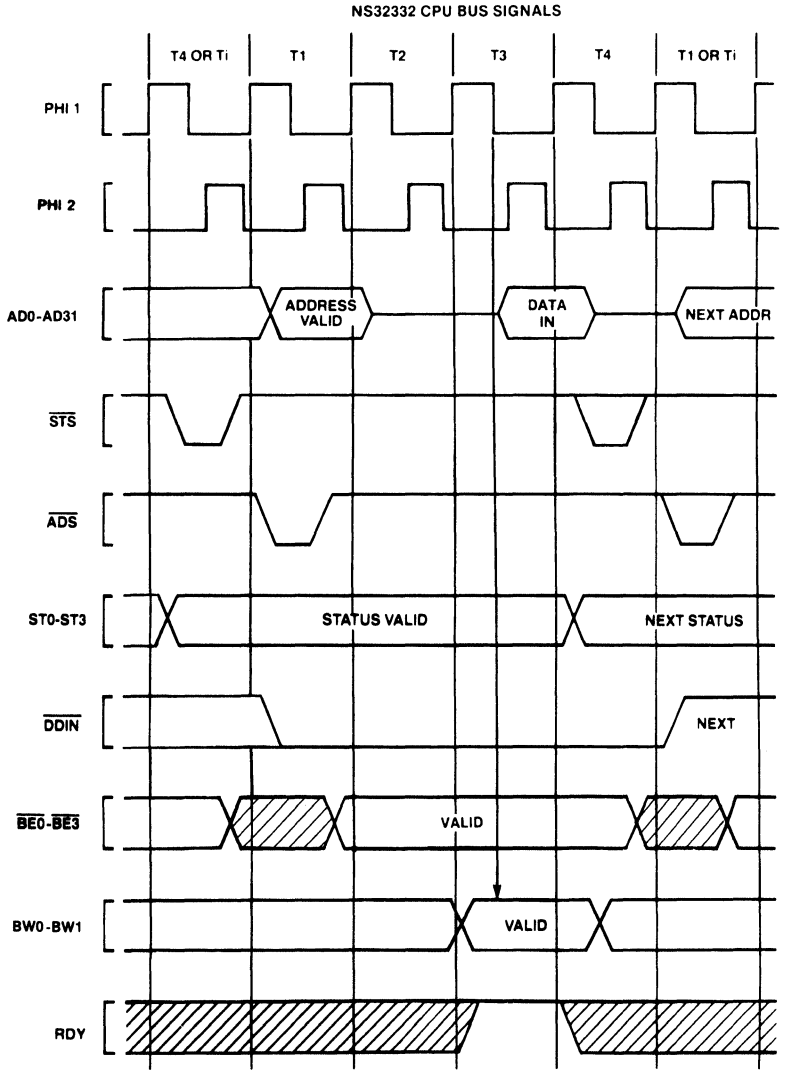
In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Sec. 4. The only external difference between them is the 4-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Sec. 3.4.6).

For case 1 (only Read) and case 2, 3, the NS32332 supports Burst cycles which are suitable for memories that can handle "nibble mode" accesses. (Sec. 3.4.2).

The sequence of events in a non-Slave, non-Burst Bus cycle is shown in *Figure 3-6* for a Read cycle, and *Figure 3-7* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).

A full speed Bus cycle is performed in four cycles of the PHI1 clock, labeled T1 through T4. Clock cycles not associated with a Bus cycle are designated Ti (for idle).

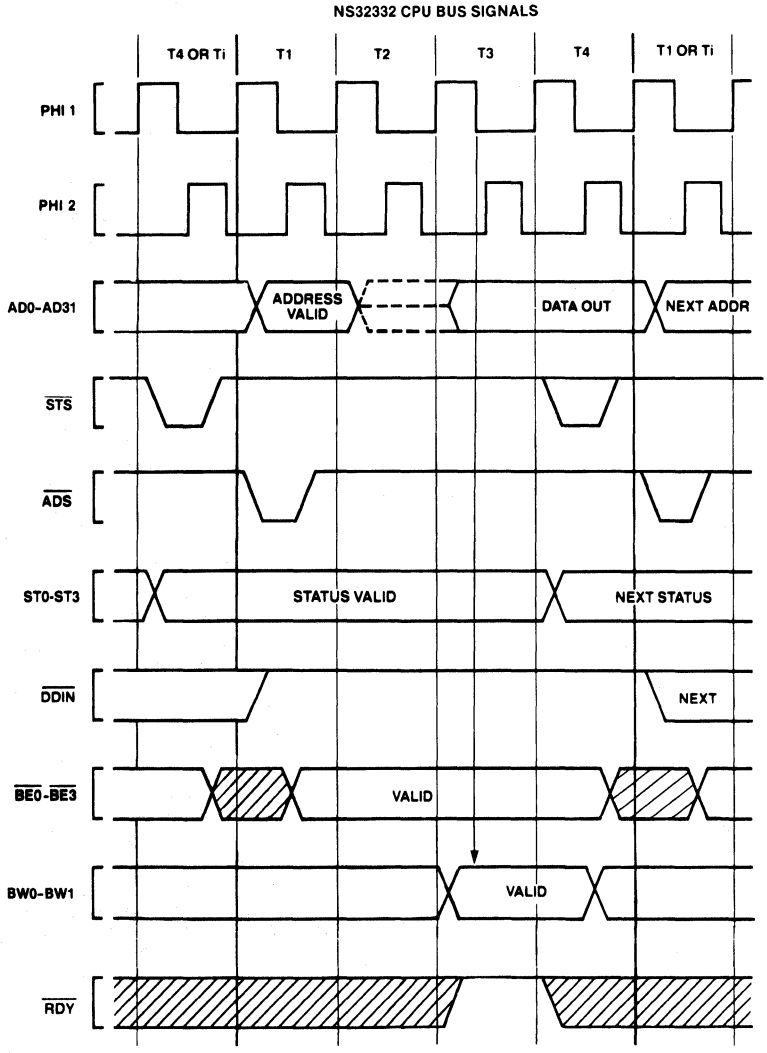
3.0 Functional Description (Continued)



TL/EE/8673-15

FIGURE 3-6. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/8673-16

FIGURE 3-7. Write Cycle Timing

3.0 Functional Description (Continued)

During T4 or Ti which preceed T1 of the current Bus cycle, the CPU applies a Status Code on pins ST0-ST3. It also provides a low-going pulse on the $\overline{STS}$ pin to indicate that the status code is valid.

The $\overline{ADS}$ signal has the dual purpose of informing the external circuitry that a Bus cycle is starting and of providing control to an external latch for demultiplexing address bits 0–31 from AD0–AD31 pins. (See *Figure 3-8.*)

During this time, the control signal $\overline{DDIN}$, which indicates the direction of the transfer, and $\overline{BE0}$–$\overline{BE3}$ which indicate which of the four bus bytes to be referenced, become valid. Note that during Instruction Fetch cycles $\overline{BE0}$–$\overline{BE3}$ are all active, but in operand Read or Write cycles they indicate the byte(s) to be referenced.

Note: If a burst cycle occurs during an operand read, all the memory banks should be enabled, during the burst cycle, regardless of $\overline{BE}$n. The CPU $\overline{BE}$n lines, in this case, are valid in the middle of T3 of the burst cycle—thus, there may not be enough time to selectively enable the different memory banks, unless a WAIT state is added. See *Figure 4-6.*

During T2 the CPU floats AD0–AD31 lines unless $\overline{DT/SDONE}$ is sampled low on the rising edge of reset and the bus cycle is a write cycle. T2 is a time window to be used for virtual to physical address translation by the Memory Management Unit, if virtual memory is used in the system.

The T3 state provides for access time requirements and it occurs at least once in a bus cycle. In the middle of T3 on the falling edge of PHI1, the RDY line is sampled to determine whether the bus cycle will be extended (Sec. 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD31) is sampled on the falling edge of PHI2 of the last T3 state. See Sec. 4. Data must, however, be held at least until the beginning of T4. The T4 state finishes the Bus cycle. Data from the CPU during Write cycles remains valid throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).

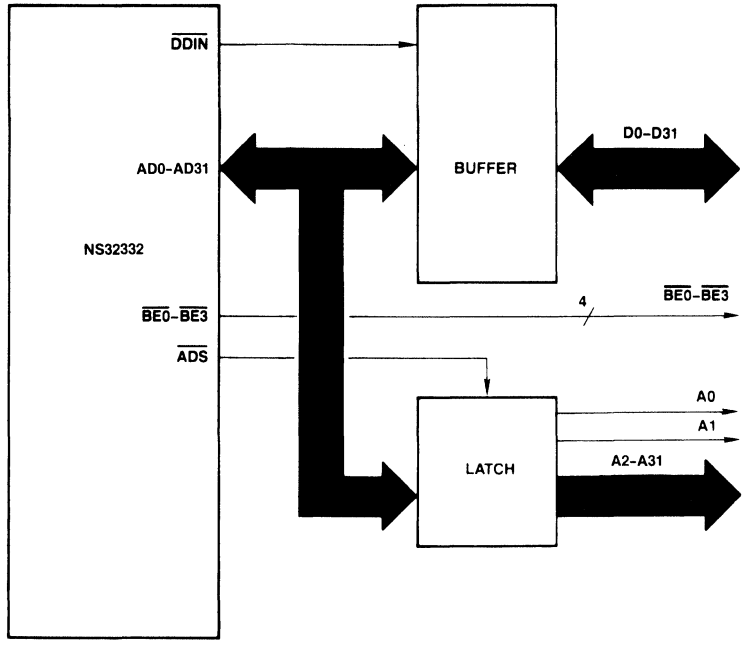
3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32332 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

In the middle of T3 on the falling edge of PHI1, the RDY line is sampled by the CPU. If RDY is high, the next T-state will be T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted and the RDY line will again be sampled on the falling edge of PHI1. Each additional T3 state after the first is referred to as a "WAIT STATE". See *Figure 3-9.*

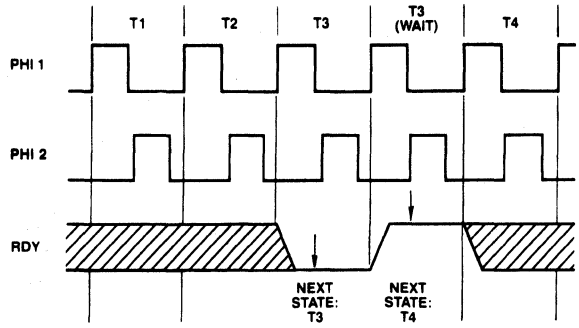
Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the RDY pin.



TL/EE/8673-17

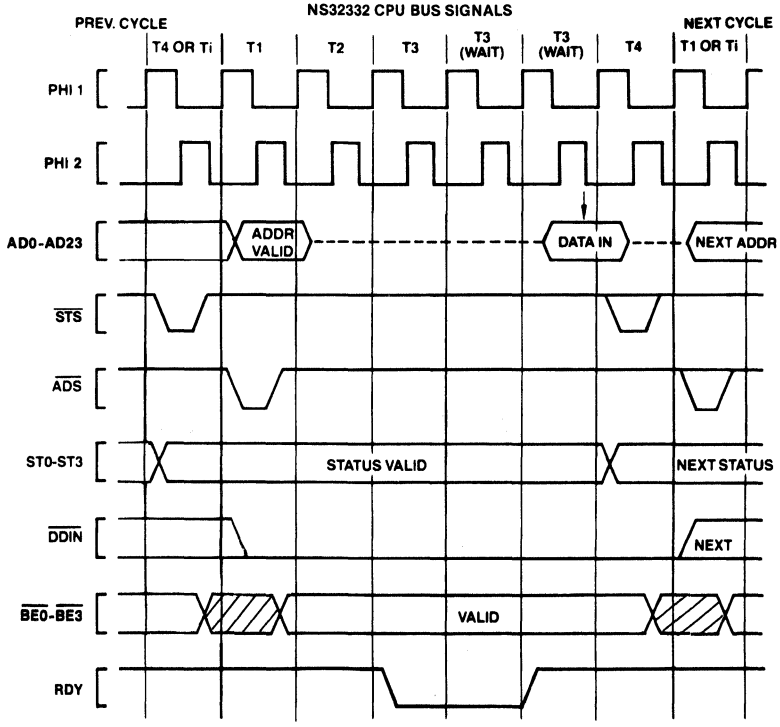
FIGURE 3-8. Bus Connections

3.0 Functional Description (Continued)



TL/EE/8673-18

FIGURE 3-9. RDY Pin Timing



TL/EE/8673-19

FIGURE 3-10. Extended Cycle Example

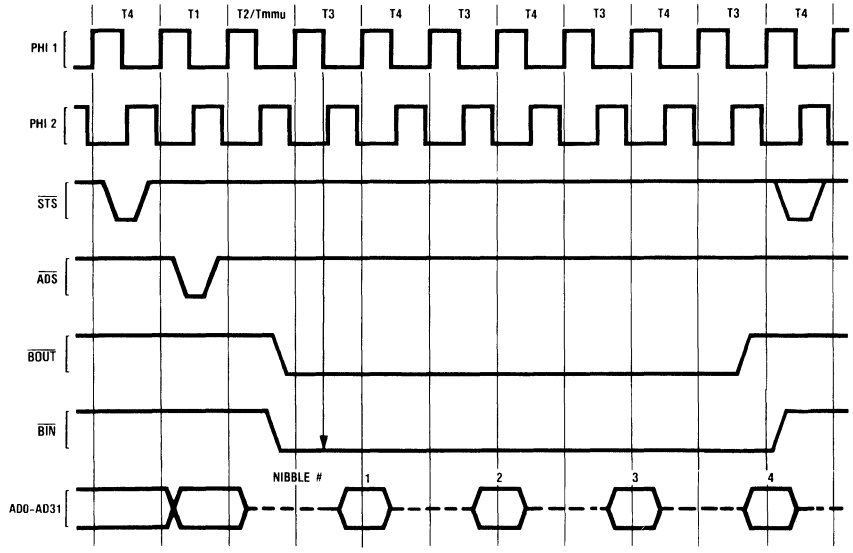
3.0 Functional Description (Continued)

3.4.2 Burst Cycles

The NS32332 is capable of performing Burst cycles in order to increase the bus throughput. Burst is available in instruction Fetch cycles and operand Read cycles only. Burst is not supported in operand Write cycles or Slave cycles.

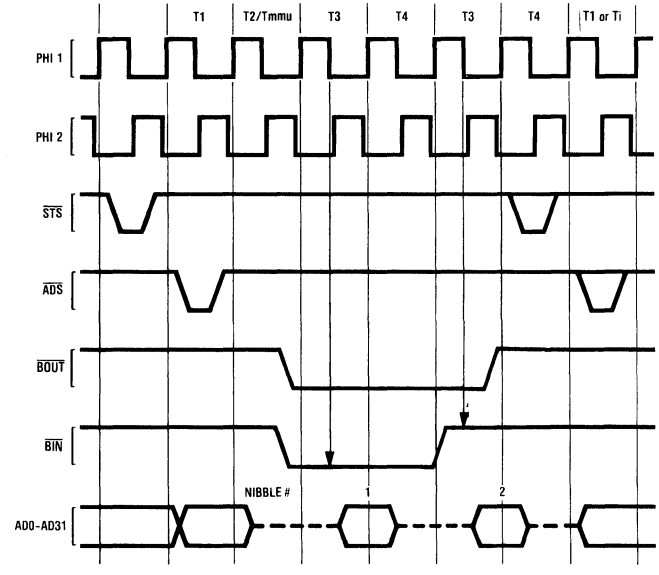
The sequence of events for Burst cycles is shown in *Figure 3-11*. The cases shown assume that the selected memory is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).

A Burst cycle is composed of two parts. The first part is a regular cycle (i.e. T1 through T4), in which the CPU outputs the new status and asserts all the other relevant control signals discussed in Sec. 3.4. In addition, the Burst Out Signal ($\overline{BOUT}$) is activated by the CPU indicating that the CPU can perform Burst cycles. If the selected memory allows



TL/EE/8673-20

(a) Normal Termination of Burst



TL/EE/8673-21

(b) External Termination of Burst

FIGURE 3-11. Burst Cycles (For Read Only)

3.0 Functional Description (Continued)

Burst cycles, it will notify the CPU by activating the burst in signal ($\overline{BIN}$). $\overline{BIN}$ is sampled by the CPU in the middle of T3 on the falling edge of PHI1. If the memory does not allow burst ($\overline{BIN}$ high), the cycle will terminate through T4 and $\overline{BOUT}$ will go inactive immediately. If the memory allows burst ($\overline{BIN}$ low), and the CPU has not deasserted $\overline{BOUT}$, the second part of the Burst cycle will be performed (see *Figure 3-11*) and $\overline{BOUT}$ will remain active until termination of the Burst.

The second part consists of up to 3 nibbles. In each nibble, a data item is read by the CPU. The duration of each nibble is 2 clock cycles labeled T3 and T4.

The Burst chain will be terminated in the following cases:

1. The CPU has reached a 16 byte boundary i.e. the byte address of the current nibble is x...x1111 (binary).

Note: In 16-bit bus systems (see Sec. 3.4.7) the Burst chain will be terminated by the CPU on an 8-byte boundary i.e. address x..x111 (binary) and in 8-bit bus system on a 4-byte boundary i.e. address x...x11 (binary).

2. $\overline{BIN}$, sampled in the current nibble's last T3, is not active any more. (See *Figure 3.11b*).
3. Bus Error or Bus Retry occurs (see Sec. 3.4.8).

Case 2 enables the Burst termination externally.

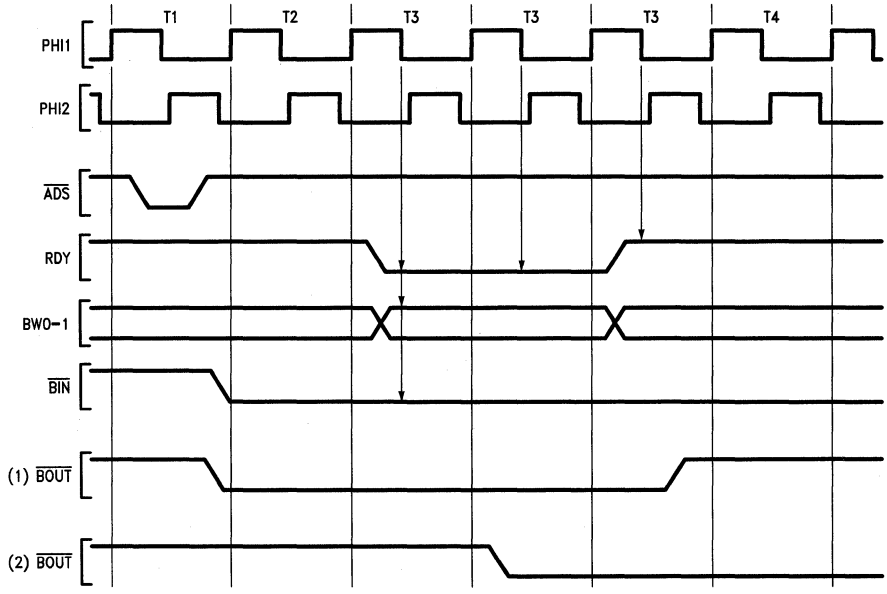
Any nibble's T3 may be extended with WAIT states using the RDY line as described in Sec. 3.4.2.

The control signals $\overline{BOUT}$, ST0–ST3, $\overline{DDIN}$ and $\overline{BE0}$–$\overline{BE3}$ remain stable during the Burst chain.

$\overline{BOUT}$ is initially set by the CPU according to the known bus width. Its state may change in a subsequent T3 as a result of a change in the bus width. *Figure 3-12* shows the resulting $\overline{BOUT}$ timing.

Note: If the selected memory is capable of handling burst transfers, it should activate $\overline{BIN}$ regardless of the state of $\overline{BOUT}$.

The reason is that $\overline{BOUT}$ may be activated by the CPU after the $\overline{BIN}$ sampling time. The $\overline{BOUT}$ signal indicates when the CPU is going to burst, and should not be interpreted as a 'Burst Request' signal.



TL/EE/8673–88

Note 1: CPU deasserts $\overline{BOUT}$.

Note 2: CPU asserts $\overline{BOUT}$.

FIGURE 3-12. $\overline{BOUT}$ Timing Resulting from a Bus Width Change

3.0 Functional Description (Continued)

3.4.3 Bus Status

The NS32332 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why is it idle.

Referring to *Figures 3-6* and *3-7*, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before $\overline{ADS}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 – The bus is idle because the CPU does not yet need to perform a bus access.

0001 – The bus is idle because the CPU is executing the WAIT instruction.

0010 – (Reserved for future use.)

0011 – The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 – Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{NMI}$) it will read from address $FFFFFF00_{16}$, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on $\overline{INT}$) it will read from address $FFFFFF00_{16}$, expecting a vector number to be provided from the Master Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded Interrupt Control Unit. The address provided is the address of ICU's Hardware Vector register. See Sec. 3.4.6.

0110 – End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Sec. 3.4.6.

0111 – End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Sec. 3.4.6.

1000 – Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.

The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

1110 – Read Slave Processor Status.

The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Sec. 3.9.1.

1111 – Broadcast Slave ID.

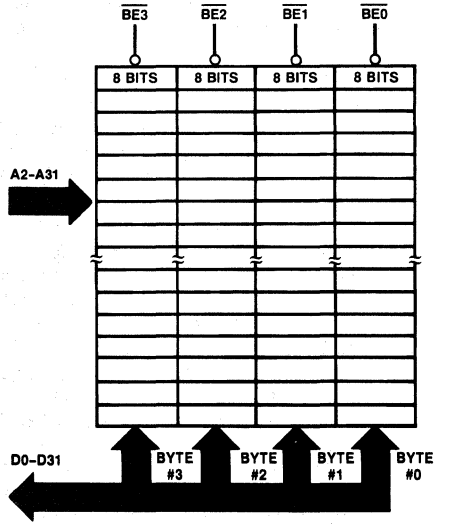
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.0 Functional Description (Continued)

3.4.4 Data Access Sequences

The 32-bit address provided by the NS32332 is a byte address; that is, it uniquely identifies one of up to 4 billion eight-bit memory locations. An important feature of the NS32332 is that the presence of a 32-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32332 provides special control signals. Byte Enable ($\overline{BE0}$–$\overline{BE3}$) which facilitate individual byte accessing on a 32-bit bus.

Memory is organized as four eight-bit banks, each bank receiving the double-word address (A2–A31) in parallel. One bank, connected to Data Bus pins AD0–AD7 is enabled when $\overline{BE0}$ is low. The second bank, connected to data bus pins AD8–AD15 is enabled when $\overline{BE1}$ is low. The third and fourth banks are enabled by $\overline{BE2}$ and $\overline{BE3}$, respectively. See *Figure 3-13.*



TL/EE/8673–22

FIGURE 3-13. Memory Interface

Since operands do not need to be aligned with respect to the double-word bus accessed performed by the CPU, a given double-word access can contain one, two, three, or four bytes of the operand being addressed, and these bytes can begin at various positions, as determined by A1, A0. Table 3-1 lists the 10 resulting access types.

TABLE 3-1
Bus Access Types

Type	Bytes Accessed	A1,A0	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$
1	1	00	1	1	1	0
2	1	01	1	1	0	1
3	1	10	1	0	1	1
4	1	11	0	1	1	1
5	2	00	1	1	0	0
6	2	01	1	0	0	1
7	2	10	0	0	1	1
8	3	00	1	0	0	0
9	3	01	0	0	0	1
10	4	00	0	0	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment. Table 3-2 lists the bus cycles performed for each situation.

3.4.4.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.4.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.4.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

3.4.5 Instruction Fetches

Instructions for the NS32332 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the twenty-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Sec. 3.4.3).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always type 10 Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle depends on the destination address.

If a non-sequential fetch is followed by additional sequential fetches which are burst continuation of the non-sequential fetch, then the Status Bus (ST0–ST3) remains the same.

Note: During instruction fetch cycles, $\overline{BE0}$–$\overline{BE3}$ are all active regardless of the alignment.

3.4.6 Interrupt Control Cycles

Activating the $\overline{INT}$ or $\overline{NMI}$ pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0–ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine.

3.0 Functional Description (Continued)

TABLE 3-2
Access Sequences

Cycle	Type	Address	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	Data Bus Byte 3	Byte 2	Byte 1	Byte 0
A. *Word at address ending with 11*									BYTE 1	BYTE 0 ← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	1	A + 1	1	1	1	0	X	X	X	Byte 1
B. *Double word at address ending with 01*							BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
C. *Double word at address ending with 10*							BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
D. *Double word at address ending with 11*							BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
E. *Quad word at address ending with 00*							BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	10	A	0	0	0	0	Byte 3	Byte 2	Byte 1	Byte 0
Other bus cycles (instruction prefetch or slave) can occur here.										
2.	10	A + 4	0	0	0	0	Byte 7	Byte 6	Byte 5	Byte 4
F. *Quad word at address ending with 01*							BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	9	A + 4	0	0	0	1	Byte 6	Byte 5	Byte 4	X
4.	1	A + 7	1	1	1	0	X	X	X	Byte 7
G. *Quad word at address ending with 10*							BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	7	A + 4	0	0	1	1	Byte 5	Byte 4	X	X
4.	5	A + 6	1	1	0	0	X	X	Byte 7	Byte 6
H. *Quad word at address ending with 11*							BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
Other bus cycles (instruction prefetch or slave) can occur here.										
1.	4	A + 4	0	1	1	1	Byte 4	X	X	X
2.	8	A + 5	1	0	0	0	X	Byte 7	Byte 6	Byte 5

X = Don't Care

3.0 Functional Description (Continued)

TABLE 3-3
Interrupt Sequences

Cycle	Status	Address	$\overline{DDIN}$	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	Data Bus Byte 3	Byte 2	Byte 1	Byte 0
					A. Non-Maskable Interrupt Control Sequences						
Interrupt Acknowledge											
1	0100	$FFFFFF00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
None: Performed through Return from Trap (RETT) instruction.											
					B. Non-Vectored Interrupt Control Sequences						
Interrupt Acknowledge											
1	0100	$FFFFFE00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
1	0110	$FFFFFE00_{16}$	0	1	1	1	0	X	X	X	X
					C. Vectored Interrupt Sequences: Non-Cascaded.						
Interrupt Acknowledge											
1	0100	$FFFFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Range: 0-127
Interrupt Return											
1	0110	$FFFFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Same as in Previous Int. Ack. Cycle
					D. Vectored Interrupt Sequences: Cascaded						
Interrupt Acknowledge											
1	0100	$FFFFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0101	Cascade Address	0			See Note		Vector, range 9–255; on appropriate byte of data bus.			
Interrupt Return											
1	0110	$FFFFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: Same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address)											
2	0111	Cascade Address	0			See Note		X	X	X	X

X = Don't Care

Note: $\overline{BE0}$-$\overline{BE3}$ signals will be activated according to the cascaded ICU address. The cycle type can be 1, 2, 3 or 4, when reading the interrupt vector. The vector value can be in the range 0–255.

3.0 Functional Description (Continued)

3.4.7 Dynamic Bus Configuration

The NS32332 interfaces to external data buses with 3 different widths: 8-bit, 16-bit and 32-bit. The NS32332 can switch from one bus width to another dynamically i.e. on a cycle by cycle basis.

This feature allows the user to include in his system different bus sizes for different purposes, like 8-bit bus for bootstrap ROM and 32-bit bus for cache memory, etc.

In each memory cycle, the bus width is determined by the inputs BW0 and BW1.

Four combinations exist:

BW1	BW0	
0	0	reserved
0	1	8-bit bus
1	0	16-bit bus
1	1	32-bit bus

The dynamic bus configuration is not applicable for slave cycles (see Sec. 3.4.1).

The BW0–BW1 lines are sampled by the CPU in T3 with the falling edge of PHI1 (see *Figure 3-14*).

If the bus width didn't change from the previous memory cycle, the CPU terminates the cycle normally.

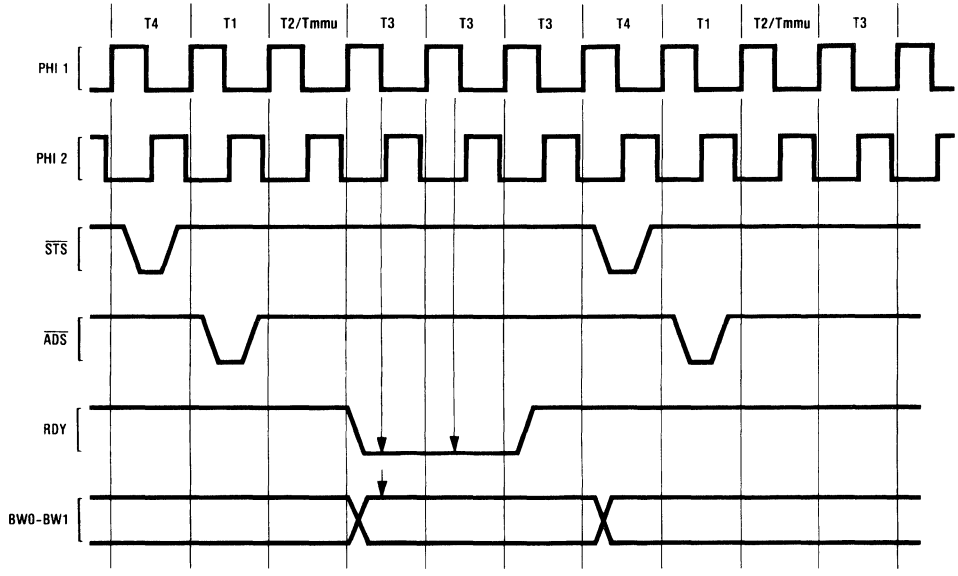
If the bus width of the current cycle is different from the bus width of the previous cycle, then two WAIT states (see Sec. 3.4.1) must be inserted in order to let the CPU switch to the new width.

The additional 2 WAIT states count from the moment BW0 BW1 change. This can be overlapped with the wait states due to slow memories.

In write cycles, the appropriate data will be present on the appropriate data lines. The CPU presents the data during T3 in a way that would fit any bus width.

If the operand being written is a byte, it will be duplicated on the 4 bytes AD0–AD31 depending on the operand address:

AddressA0 – 1 =	00	XX	XX	XX	OP
	01	XX	XX	OP	OP
	10	XX	OP	XX	OP
	11	OP	XX	OP	OP



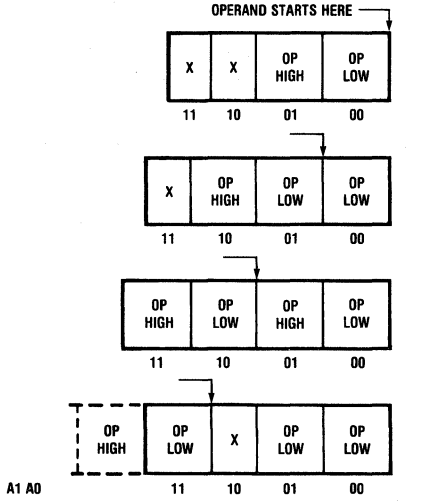
TL/EE/8673–23

FIGURE 3-14. Bus width changes. Two wait states are required after the signals BW0–BW1 change.

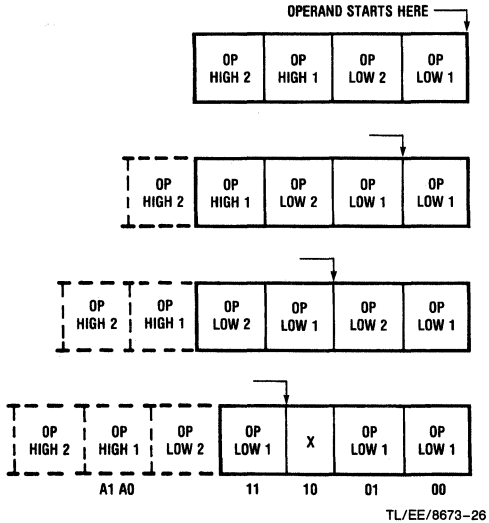
3.0 Functional Description (Continued)

If the operand being written is a word, 4 cases exist. The operand address can be x...x00 (binary) or x...x01 (binary) or x...x10 or x...x11 (binary).

See the duplications for each case:



TL/EE/8673-25



TL/EE/8673-26

If the operand being written is a double word 4 cases exist: The operand address can be x...x00 (binary) or x...x01 (binary) or x...x10 (binary) or x...x11 (binary).

See the duplications for each case:

Note that the organization of the operand described applies to the initial part of the operand cycle. For instance, if the CPU writes a double word operand to a 16-bit bus and the operand address is x...x11 (binary) it needs three memory cycles.

The description above applies to the first cycle. In the other 2 memory cycles belonging to the same operand, the data will be presented on the data bus lines to fit 16-bit bus width and take into account the operand length.

Example:

The CPU has to write a double word DDCCBBAA to address HEX 987653 which is in a 16-bit bus area. In the first cycle, the CPU does not know the width until T3 so it generates a cycle to address 987653 which activates the $\overline{BE3}$ line and puts on the data bus AA XX AA AA (X = don't care). After this cycle, the CPU knows it has a 16-bit bus and it generates a cycle to address 987654 which activates the BE0, BE1 and BE2 lines and puts on the data bus XX XX CC BB. The last cycle will address 987656, activate BE2, and put on the data bus XX XX XX DD. The BE0–BE3 lines are always activated as if the bus is 32-bit wide, regardless of BW0–BW1 state.

The CPU does not support a change of the bus width during a sequence of several memory references belonging to the same operand e.g. nonaligned double word. In other words, any operand should not be split between two memory spaces having different bus widths.

Instruction Fetches do not fall in this category and an Instruction Fetch can have its own bus width regardless of the bus width in the previous cycle.

3.4.8 Bus Exceptions

Any bus cycle may have a bus error during its execution. The error may be corrected during the current cycle or may be incorrectable. The NS32332 can handle both types of errors by means of BUS RETRY and BUS ERROR.

3.4.8.1 Bus Retry

If a bus error can be corrected, the CPU may be requested to repeat the erronous bus cycle. The request is done by asserting the $\overline{BRT}$ (Bus Retry) signal.

The CPU response to Bus Retry depends on the cycle type:

Instruction Fetch Cycle—If the RETRY occurs during an instruction fetch, the fetch cycle will be retried as soon as possible. If the RETRY is requested during a burst chain, the burst is stopped and the fetch is retried. The only delay in retrying the instruction fetch may result from pending operand requests (and, of course, from hold or wait requests).

Operand Read Cycle—If the RETRY occurs on an operand read, the bus cycle is immediately repeated. If the data read is "multiple" e.g. non-aligned, only the problematic part will be repeated. For instance, if the cycle is a non-aligned double word and the second half failed, only the second part will be repeated. The same applies for a RETRY occurring during a burst chain. The repeated cycle will begin where the read operand failed (rather than the first address of the burst) and will finish the original burst.

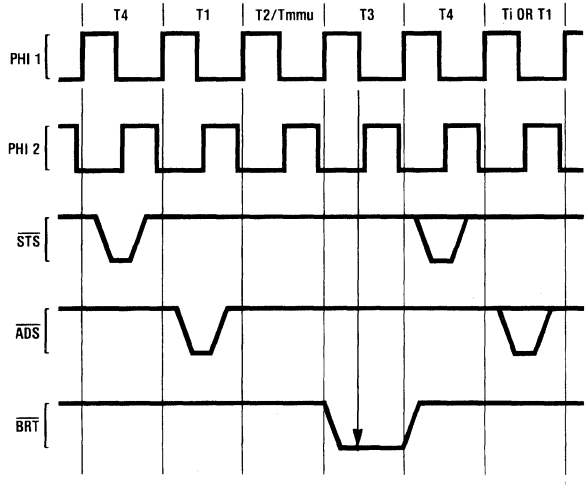
3.0 Functional Description (Continued)

Operand Write Cycle—If the RETRY occurs on a write, the bus cycle is immediately repeated. If the operand write is "multiple" e.g. non-aligned, only the problematic part will be repeated. For instance, if the cycle is a non-aligned double word and the second half failed, only the second part will be repeated.

A Bus Retry is requested by activating the $\overline{BRT}$ line (see *Figure 3-15*). $\overline{BRT}$ is sampled by the CPU during T3 on the falling edge of PHI1. If $\overline{BRT}$ is not active, the cycle will be terminated in a regular way. If $\overline{BRT}$ is active, $\overline{BRT}$ will be sampled again during T4 on the falling edge of PHI1. If $\overline{BRT}$ is not active, the cycle will be terminated in a regular way. If $\overline{BRT}$ is active, T4 will be followed by an idle state and the cycle will be repeated, i.e. a new T4 for setting the Status Bus and issuing $\overline{STS}$ and then T1 through T4 will be performed.

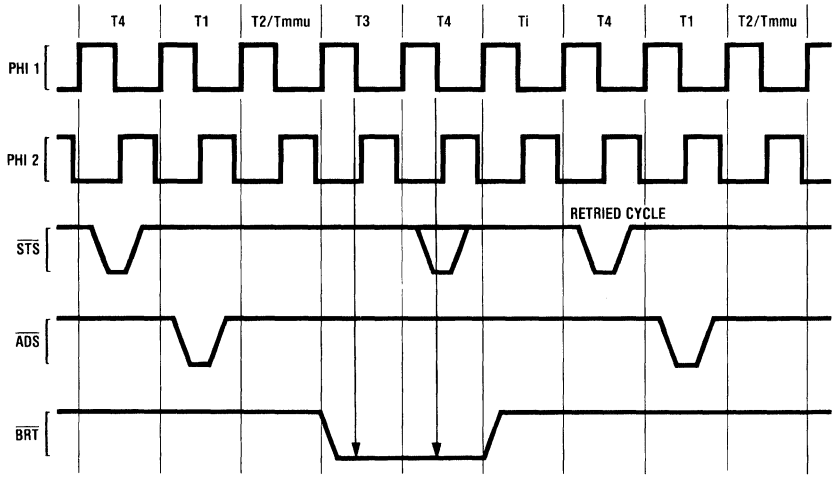
Although the decision about Retry is taken by the CPU on T4, $\overline{BRT}$ must have an early activation in T3 as described above in order to prevent the internal pipeline to advance. Holding the pipeline allows the repeated cycle to override the original one. If $\overline{BRT}$ is activated only in T3 and not in T4, there might be one cycle penalty in the performance of the execution unit in operand read cycles.

Retry is applicable for regular memory cycles and burst cycles, but not for Slave cycles.



TL/EE/8673-27

(a) Bus Cycle Not Retried



TL/EE/8673-28

(b) Bus Cycle Retried

FIGURE 3-15. Bus Cycle Retry

3.0 Functional Description (Continued)

3.4.8.2 Bus Error

If a Bus Error is incorrectable the CPU may be requested to abort the current process and branch to an appropriate routine to handle the error. The request is performed by activating the $\overline{BER}$ signal.

$\overline{BER}$ is sampled by the CPU during T4 on the falling edge of PHI1. If $\overline{BER}$ is active the bus will go to Tidle after T4 and the CPU will jump to the Bus Error handler (see Sec. 3.8).

The CPU response to Bus Error depends on the cycle type:

Instruction Fetch Cycles—If the bus error occurs on an instruction fetch, additional fetches are inhibited including the one which failed. If, after inhibiting instruction fetches, some operand cycles are still pending within the CPU, they are executed normally, delaying the access to the bus error exception. If and when the internal instruction queue becomes empty, the CPU will enter the BUS ERROR exception. This arrangement enables the CPU to ignore bus errors which belong to fetch ahead cycles if these fetches are not to be used as a result of a jump.

Operand Read Cycles—If the bus error occurs on an operand read, the bus error is immediately accepted, and the CPU enters the BUS ERROR exception.

Operand Write Cycles—If the bus error occurs on an operand write, the exception is immediately accepted.

Note 1: When a bus error occurs, the instruction that caused the error is generally not re-executable.

The process that was being executed should either be aborted or should be restarted from the last checkpoint.

Note 2: Bus error has top priority and is accepted even during the acknowledge sequence of another CPU exception (i.e. Abort, Interrupt, etc.).

It is the responsibility of the user software to detect such an occurence and to take the appropriate corrective actions.

3.4.8.3 Fatal Bus Error

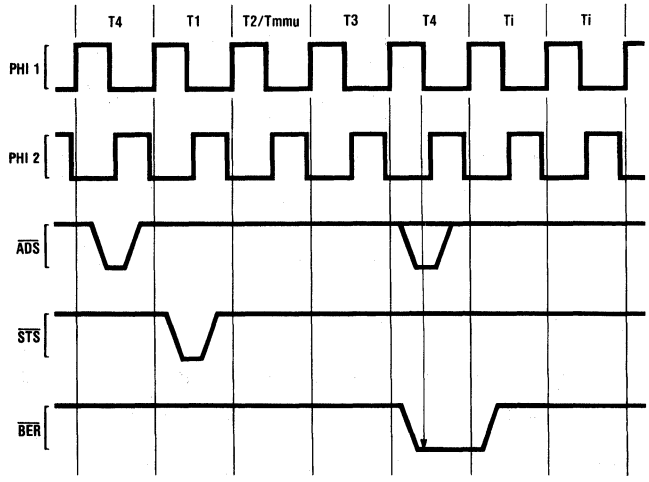
As previously mentioned, the CPU response to a bus error is to interrupt the current activity and enter the error routine.

An exception to this rule occurs when a bus error is signalled to the CPU during the acknowledge of a previous bus error. In this case the second error is interpreted by the CPU as a fatal bus error.

The CPU will respond to this event by halting execution and floating $\overline{ADS}$, $\overline{BE0}$–$\overline{BE3}$, $\overline{DDIN}$, $\overline{STS}$ and AD0–AD31.

The Halt condition is indicated by the setting of ST0–ST3 to zero and by the assertion of $\overline{MC}/\overline{EXS}$.

The CPU can exit this condition only through a hardware reset.



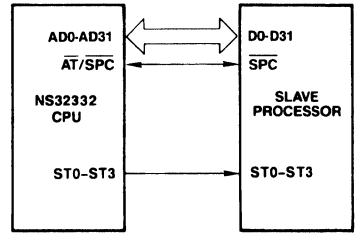
TL/EE/8673–30

FIGURE 3-16. Bus Error During Read or Write Cycle

3.0 Functional Description (Continued)

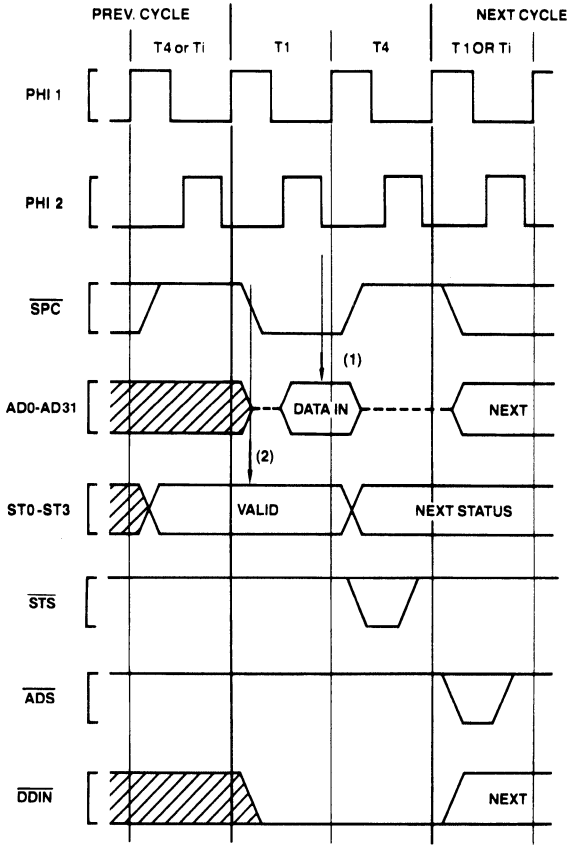
3.4.9 Slave Processor Communication

The $\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{SPC}$). In a Slave Processor bus cycle, data is transferred on the Data Bus and the status lines (ST0–ST3) are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.



TL/EE/8673–31

FIGURE 3-17. Slave Processor Connections



TL/EE/8673–32

Notes:

(1) CPU samples Data Bus here.

(2) Slave Processor samples CPU Status here.

FIGURE 3-18. CPU Read from Slave Processor

3.0 Functional Description (Continued)

3.4.9.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-18* and *3-19*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

The CPU does not pulse the address ($\overline{ADS}$) and status ($\overline{STS}$) strobes during a slave protocol. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.9.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more slave operand cycles. The NS32332 supports two slave protocols which can be selected by the configuration register (CFG).

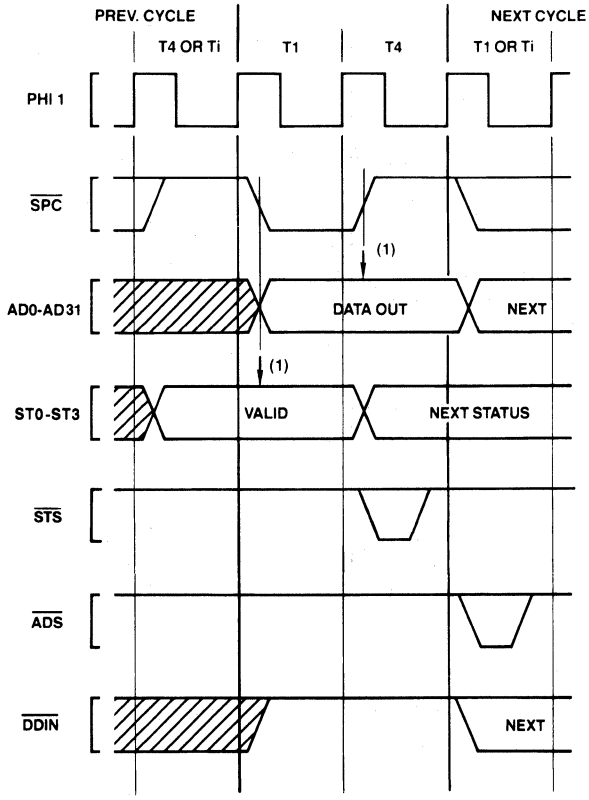
1. The regular Slave protocol is fully compatible with NS32032, NS32016 and NS32008 slave protocols.

   In this protocol the NS32332 uses only the two least significant bytes of the data bus for slave cycles. This allows the NS32332 CPU to work with the current slaves (like NS32082, NS32081 etc.)

   A byte operand is transferred on the least significant byte of the data bus (AD0–AD15).

   A double word is transferred in a consecutive pair of bus cycles least significant word first. A quadword is transferred in two pairs of slave cycles.
2. The fast slave protocol is unique to the NS32332 CPU. In this protocol the NS32332 uses the full width of the data bus (AD0–AD31) for slave cycles.

   A byte operand is transferred on the least significant byte of the data bus (AD0–AD7), a word operand is transferred on bits AD0–AD15 and a double word operand is transferred on bits AD0–AD31. A quad word is transferred in two pairs of slave cycles with other bus cycles possibly occurring between them.



TL/EE/8673–33

Note:

(1) Arrows indicate points at which the Slave Processor samples.

FIGURE 3-19. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32332 CPU, in conjunction with the Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

When an MMU is used, the states T2 and TMMU are overlapped. During this time the CPU places AD0–AD31 into the TRI-STATE mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{PAV}$. *Figure 3-20* shows the Bus Cycle timing with address translation.

Note 1: If an NS32382 MMU is used, the bus lines AD0–AD31 can be used during T2 by the CPU to output data during write cycles, since the NS32382 uses a separate physical address bus—this data timing can be selected by forcing $\overline{DT/SDONE}$ low during reset as shown in *Figure B-2* in Appendix B.

$\overline{DT/SDONE}$ must be forced high during reset if an NS32082 MMU is used since, in this case, no separate physical address bus is provided.

Note 2: If an NS32082 MMU is used, in order for it to operate properly, it must be set to the 32-Bit mode by forcing a A24/$\overline{HBF}$ low during reset. In this mode the bus lines AD16–AD24 are floated after the MMU address has been latched, since they are used by the CPU to transfer data.

3.5.1 The $\overline{FLT}$ (Float) Pin

The $\overline{FLT}$ signal is used by the CPU for address translation support. Activating $\overline{FLT}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the MMU in order to update its Translation Lookaside Buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-21 shows the effect of $\overline{FLT}$. Upon sampling $\overline{FLT}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD31, and $\overline{DDIN}$ to the TRI-STATE condition ("floating").
2) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{RST/ABT}$ description.)

The above conditions remain in effect until $\overline{FLT}$ again goes high. See Sec. 4.

3.5.2 Aborting Bus Cycles

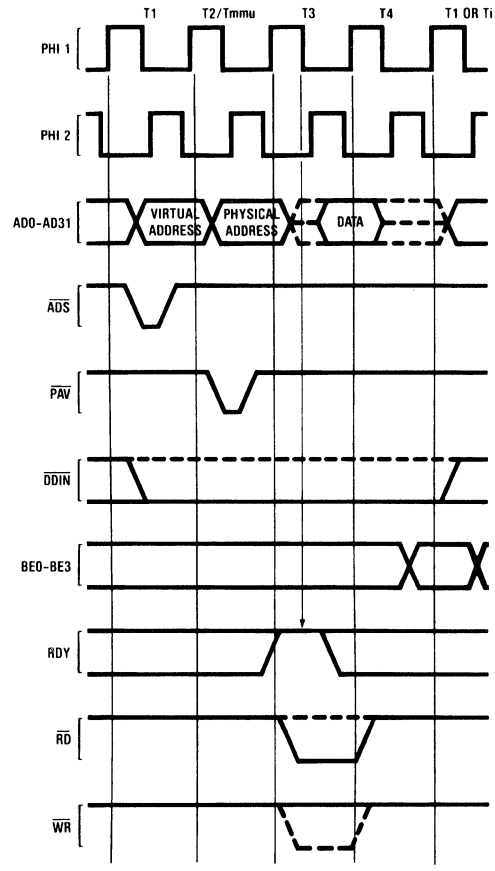
The $\overline{RST/ABT}$ pin, apart from its Reset function (Sec. 3.3), also serves as the means to "abort", or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the $\overline{RST/ABT}$ pin is held active for only one clock cycle.

If $\overline{RST/ABT}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. Since it is the MMU $\overline{PAV}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.
2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later.



TL/EE/8673–87

FIGURE 3-20. Read (Write) Cycle with Address Translation

3.5.2.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

3.5.2.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the Memory Management Unit.

1) If $\overline{FLT}$ has not been applied to the CPU, the Abort pulse must occur during Tmmu.

3.0 Functional Description (Continued)

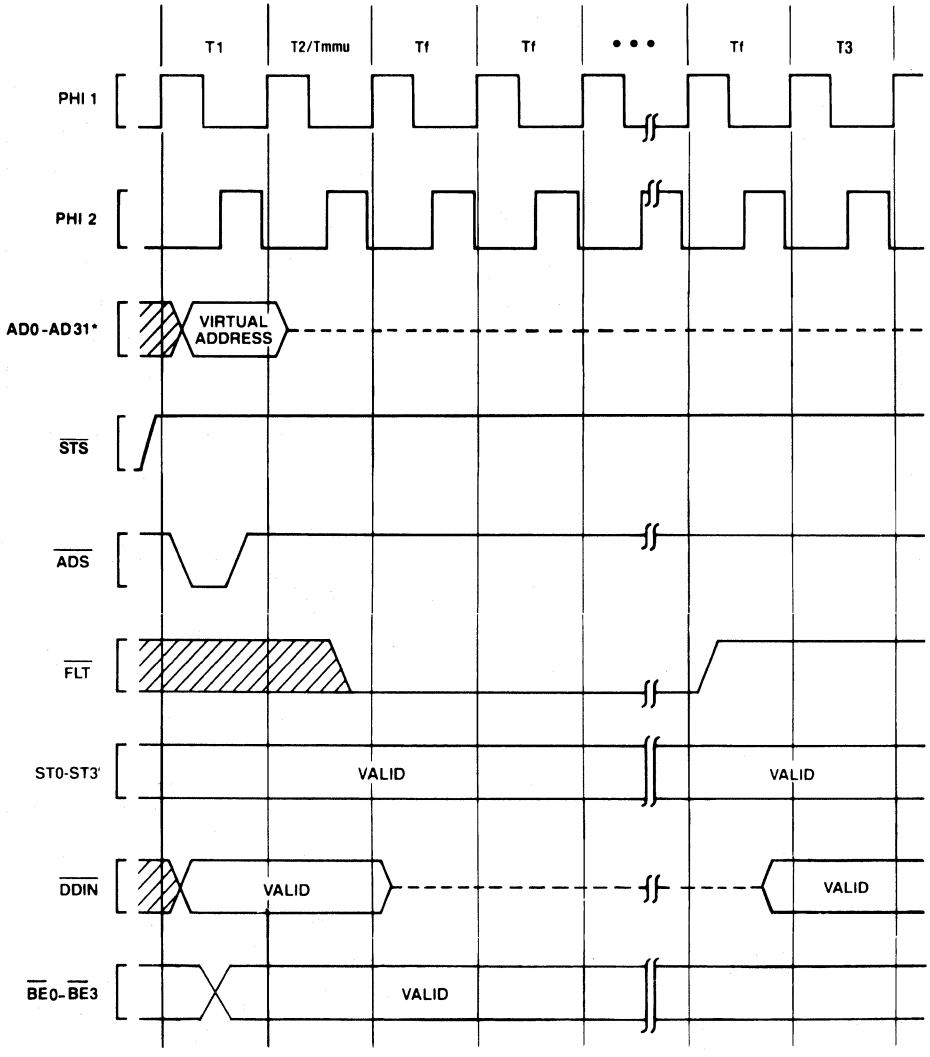
2) If $\overline{FLT}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{FLT}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{FLT}$ is removed.

3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If $\overline{RST}/\overline{ABT}$ is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

3.6 BUS ACCESS CONTROL

The NS32332 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{HOLD}$ (Hold Request) and $\overline{HLDA}$ (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of $\overline{HLDA}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the



*See MMU data sheet for details on physical address timing and MMU initiated Bus cycles.

TL/EE/8673-34

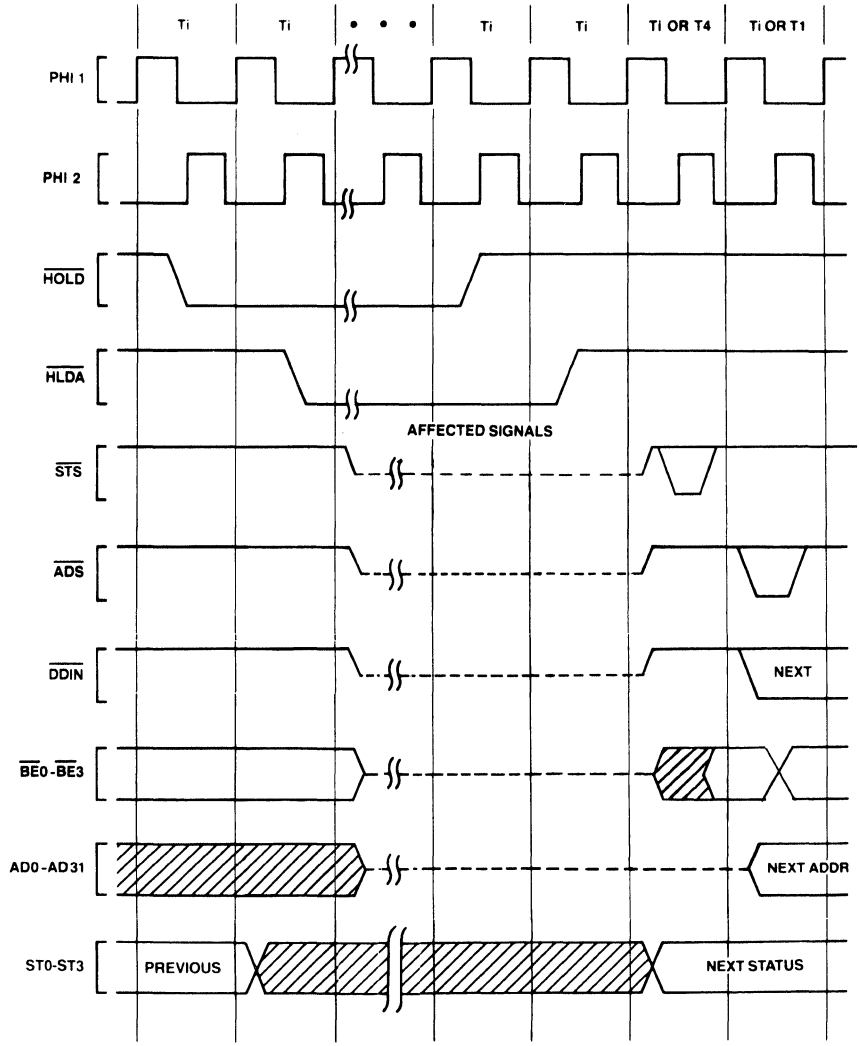
FIGURE 3-21. $\overline{FLT}$ Timing

3.0 Functional Description (Continued)

AD0–AD31, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{BE0}$–$\overline{BE3}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{HOLD}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{HLDA}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{HOLD}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-22* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-23* shows the sequence if the CPU is using the bus at the time that the $\overline{HOLD}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{HLDA}$ signal is connected in a daisy-chain through the MMU, so that the MMU can release the bus if it is using it.



TL/EE/8673–35

FIGURE 3-22. $\overline{HOLD}$ Timing, Bus Initially Idle

3.0 Functional Description (Continued)

3.7 INSTRUCTION STATUS

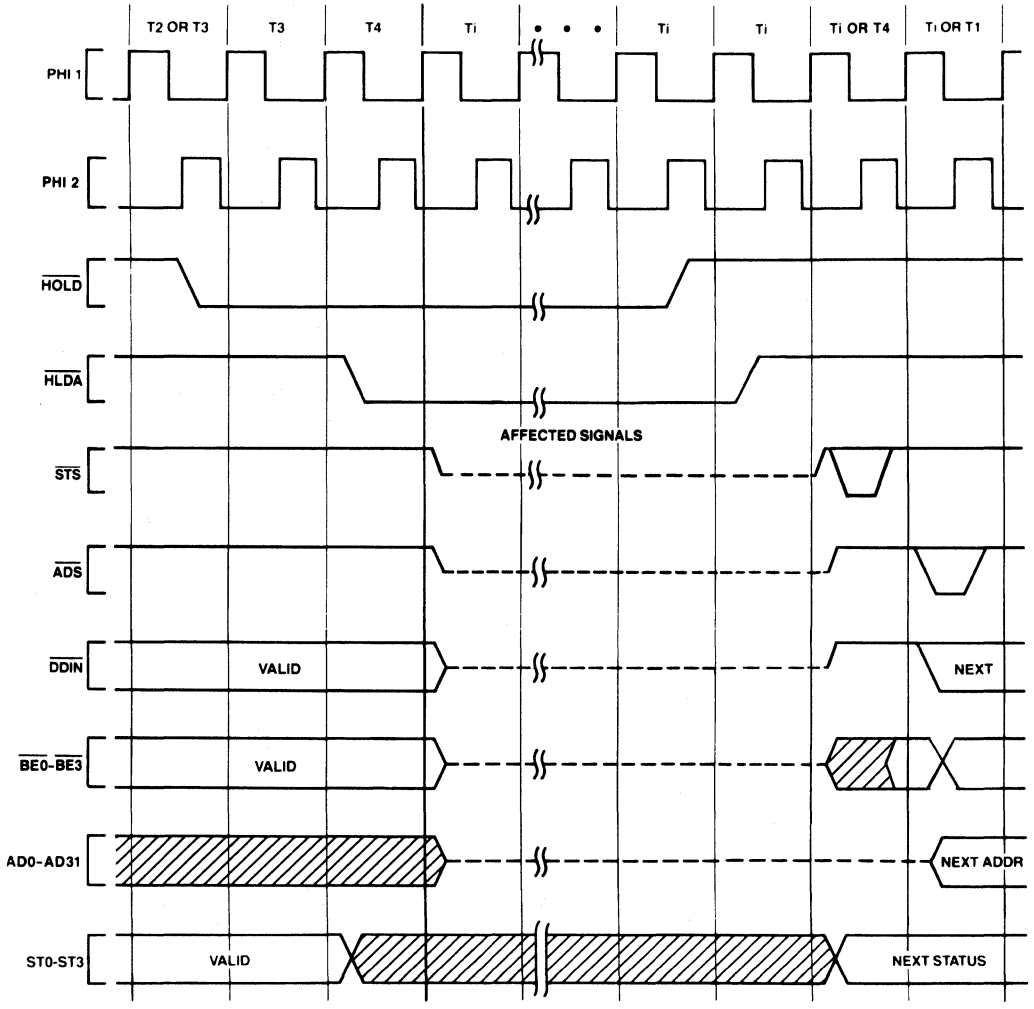
In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32332 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes.

U/$\overline{S}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection, and debugging purposes. U/$\overline{S}$ line is updated every T4.

$\overline{ILO}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing.

While $\overline{ILO}$ is active, the CPU inhibits instruction fetches. In order to prevent MMU cycles during $\overline{ILO}$, the CPU executes a dummy Read cycle with status code 1011 (RMW) prior to activating $\overline{ILO}$. Thereafter, $\overline{ILO}$ is activated and the Read is performed again but with status code 1010 (operand transfer). Refer to *Figure 3-24*.



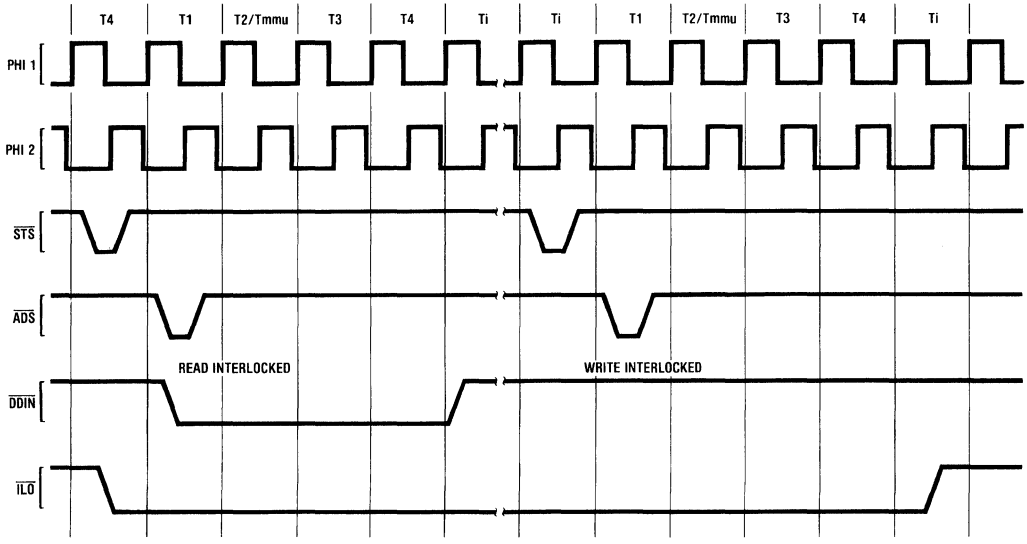
TL/EE/8673–36

FIGURE 3-23. $\overline{HOLD}$ Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

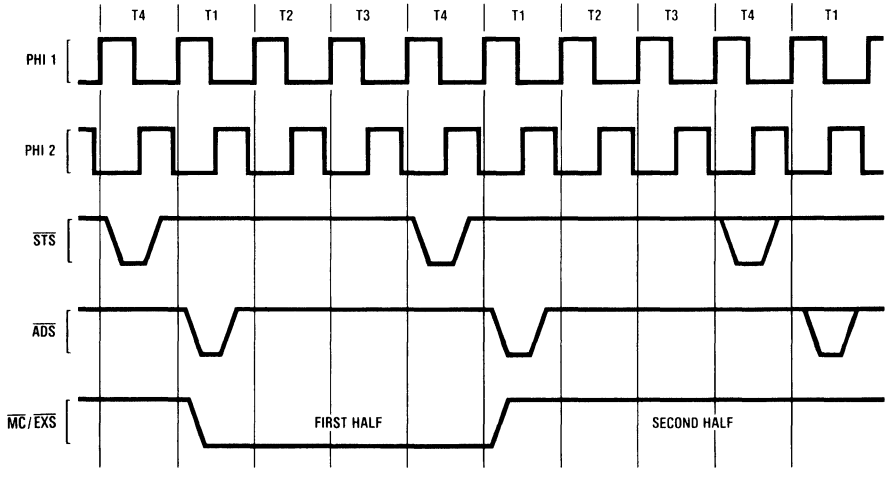
$\overline{MC}/\overline{EXS}$ (Multiple Cycle/Exception Status) is activated during the access of the first part of an operand that crosses a double-word address boundary. The activation of this signal is independent of the selected bus width. Its timing is shown in *Figure 3-25*. The MMU or other external circuitry can use it as an early indication of a CPU access to an operand that crosses a page boundary.

$\overline{MC}/\overline{EXS}$ is also activated during the first non-sequential instruction fetch (status code 1001) following an abort, and when the CPU enters the idle state (Status Code 0000) following a fatal bus error.



TL/EE/8673-37

FIGURE 3-24. $\overline{ILO}$ Timing



TL/EE/8673-38

FIGURE 3-25. Non-aligned Write Cycle—$\overline{MC}/\overline{EXS}$ Timing

3.0 Functional Description (Continued)

3.8 NS32332 INTERRUPT STRUCTURE

$\overline{INT}$, on which maskable interrupts may be requested,

$\overline{NMI}$, on which non-maskable interrupts may be requested, and

$\overline{RST}/\overline{ABT}$, which may be used to abort a bus cycle and any associated instruction. See Sec. 3.5.2.

In addition there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

1) Adjustment of Registers.

Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2) Saving Processor Status.

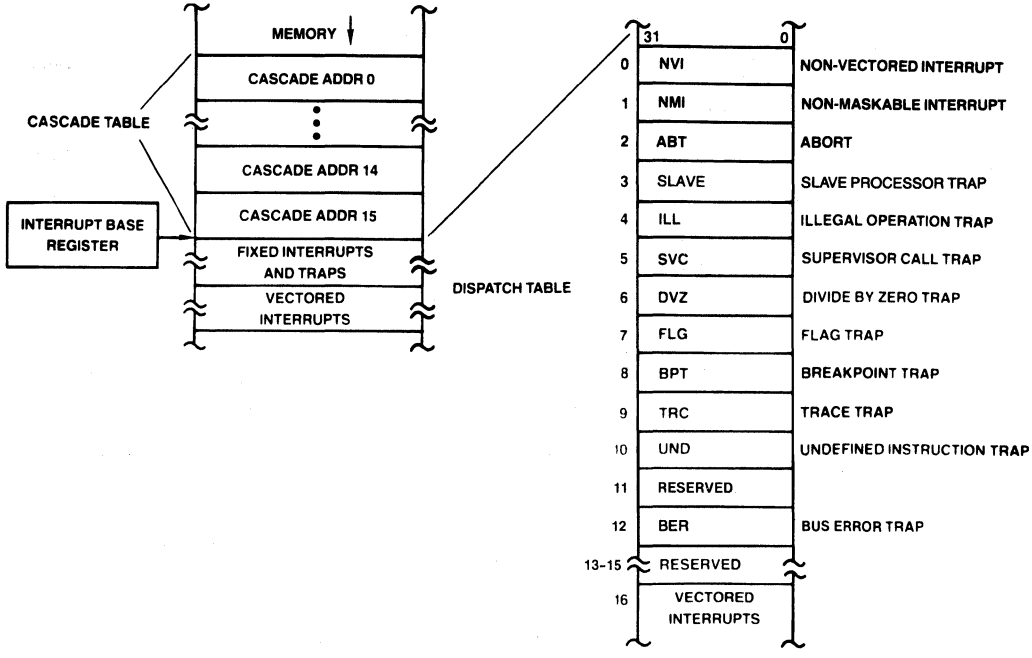
The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

3) Vector Acquisition.

A Vector is either obtained from the Data Bus or is supplied by default.

4) Service Call.

The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-26*. A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.



TL/EE/8673-39

FIGURE 3-26. Interrupt Dispatch Table

3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-27*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Sec. 3.8.7.1.
Abort Interrupt:	Sec. 3.8.7.4.
Traps (except Trace):	Sec. 3.8.7.2.
Trace Trap:	Sec. 3.8.7.3.

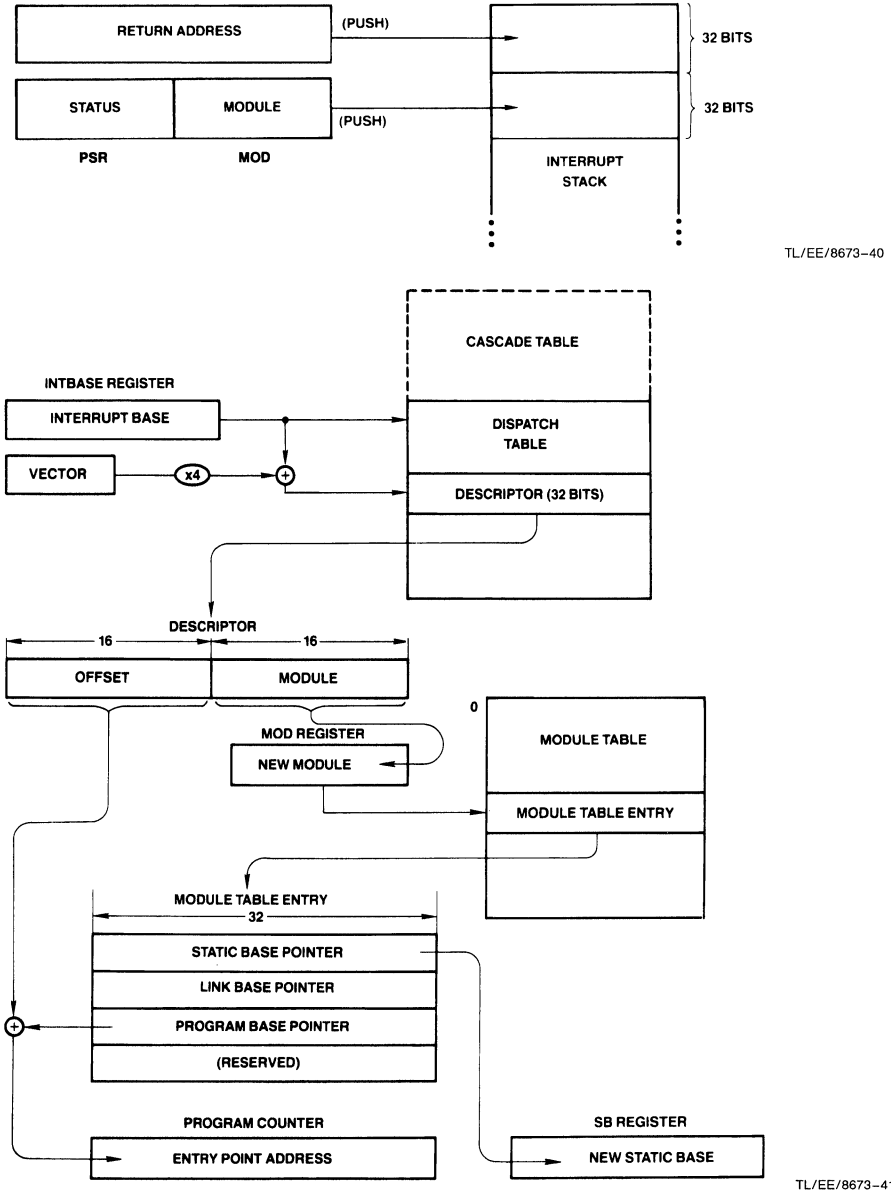


FIGURE 3-27. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction *(Figure 3-28)* restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-29*.

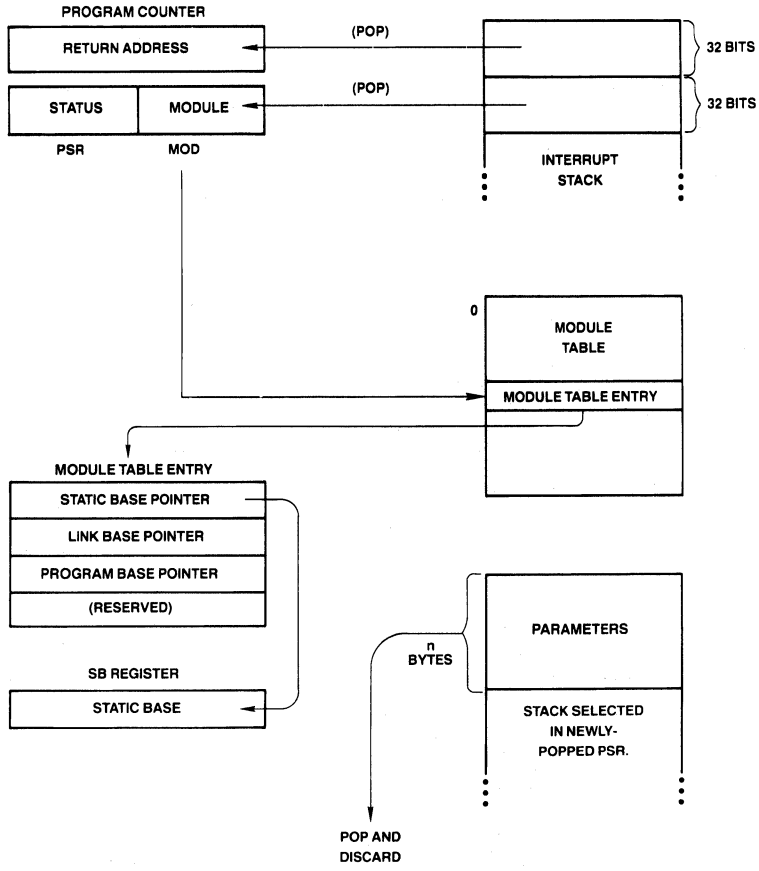
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$, $\overline{\text{NMI}}$ or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = C) or Vectored (bit I = 1).

3.8.3.1 Non-Vectored Mode

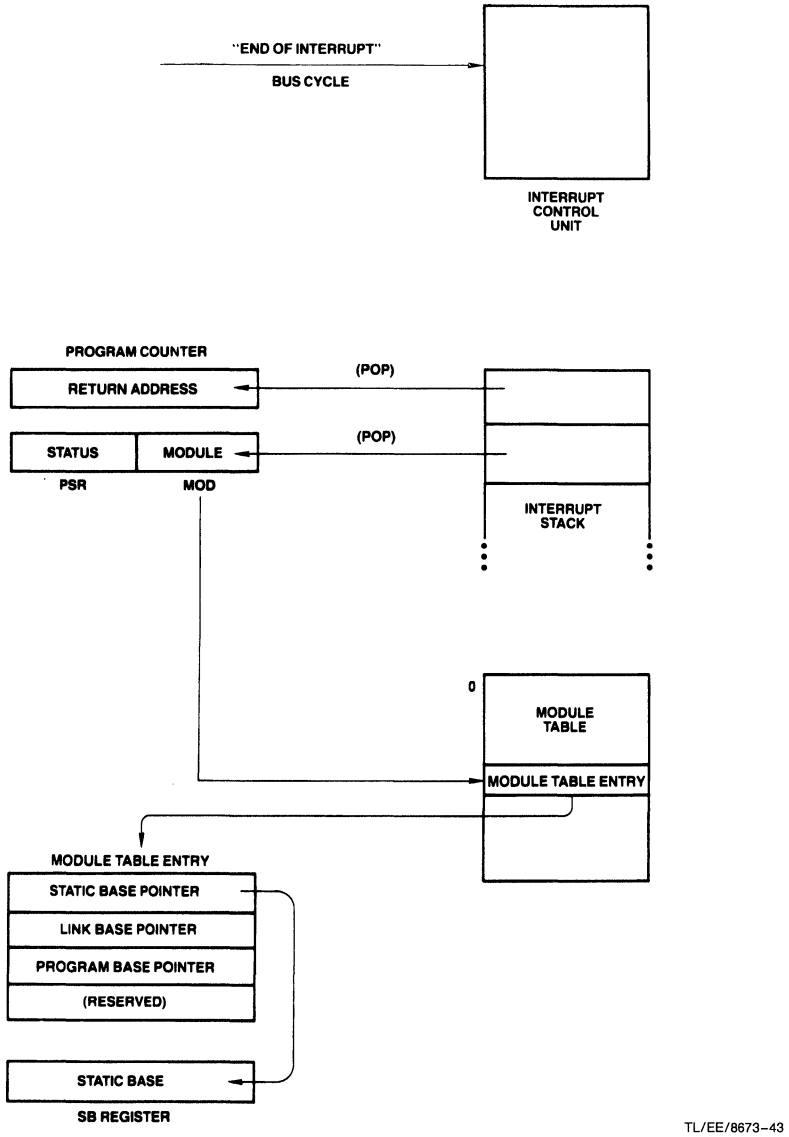
In the Non-Vectored mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.



TL/EE/8673–42

FIGURE 3-28. Return from Trap (RETT n) Instruction Flow

3.0 Functional Description (Continued)



TL/EE/8673-43

FIGURE 3-29. Return from Interrupt (RETI) Instruction Flow

3.0 Functional Description (Continued)

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize many interrupt requests. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.3) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.8.3.3 Vectored Mode: Cascaded Case

In order to allow more levels of interrupt, provision is made in the CPU to transparently support cascading. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{INT}$ pin. Refer to the ICU data sheet for details.

In a system which uses cascading, two tasks must be performed upon initialization:

1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number on which it receives the cascaded requests.
2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-26 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range -16 to -1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Sec. 3.4.3), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Sec. 3.4.3), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.3), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the interrupt mask register of the interrupt controller.

However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.

3.8.4 Non-Maskable Interrupt (The $\overline{NMI}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the $\overline{NMI}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.3) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is $FFFFFF00_{16}$. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Sec. 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32332 CPU are:

Trap (SLAVE): An exceptional condition was detected by the Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

Trap (BER): A bus error condition was encountered.

3.0 Functional Description (Continued)

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

Note: A slight difference exists between the NS32332 and previous Series 32000 CPUs when tracing is enabled.

The NS32332 always clears the P bit in the PSR before pushing the PSR on the stack. Previous CPUs do not clear it when a trap (ILL) occurs.

The result is that an instruction that causes a trap (ILL) exception is traced by previous Series 32000 CPUs, but is never traced by the NS32332.

3.8.6 Prioritization

The NS32332 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1) Traps other than Trace (Highest priority)
2) Abort
3) Bus Error
4) Non-Maskable Interrupt
5) Maskable Interrupts
6) Trace Trap (Lowest priority)

3.8.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-30*. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{INT}$ or $\overline{NMI}$ pins, respectively), see Sec. 3.8.7.1 For Abort Interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{NMI}$ pin receives a falling edge, or the $\overline{INT}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master, Sec. 3.4.3). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address $FFFFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.3). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address $FFFFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.3).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE $+4*$ Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Sec. 3.4.3).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-30*.

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector* 4 + INTBASE Register contents.

2) Move the Module field of the Descriptor into the MOD Register.

3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.

4) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.

5) Flush queue: Non-sequentially fetch first instruction of Interrupt routine.

6) Push MOD Register into the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)

7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-30. Service Sequence

Invoked during all interrupt/trap sequences.

3.8.7.2 Trap Sequence: Traps Other Than Trace

1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2) Set "Vector" to the value corresponding to the trap type.

SLAVE:	Vector = 3.
ILL:	Vector = 4.
SVC:	Vector = 5.
DVZ:	Vector = 6.
FLG:	Vector = 7.
BPT:	Vector = 8.
UND:	Vector = 10.

3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.

3.0 Functional Description (Continued)

4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Return Address" to the address of the first byte of the trapped instruction.
6) Perform Service (Vector, Return Address), *Figure 3-30.*

3.8.7.3 Trace Trap Sequence

1) In the Processor Status Register (PSR), clear the P bit.
2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4) Set "Vector" to 9.
5) Set "Return Address" to the address of the next instruction.
6) Perform Service (Vector, Return Address), *Figure 3-30.*

3.8.7.4 Abort Sequence

1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
2) Clear the PSR P bit.
3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Vector" to 2.
6) Set "Return Address" to the address of the first byte of the aborted instruction.
7) Perform Service (Vector, Return Address), *Figure 3-30.*

3.8.7.5 Bus Error Sequence

1) The same as Abort sequence above, but set vector to 12.

3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32332 CPU recognizes three groups of instructions being executable by external Slave Processor:

Floating Point Instruction Set

Memory Management Instruction Set

Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

In addition, each slave instruction will be performed either through the regular (32032 compatible) slave protocol or through a fast slave protocol according to the relevent bit in the configuration register (Sec. 2.1.3).

A combination of one slave communicating with an old protocol and another with a new protocol is allowed, e.g. 16-bit FPU (32081) and 32-bit MMU (32382) or vice versa.

3.9.1 16-Bit Slave Processor Protocol (32032 Compatible)

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.
2) It specifies which Slave Processor will execute it.
3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-31.* While applying Status Code 1111 (Broadcast ID, Sec. 3.4.3), the CPU transfers the ID Byte on the least-significant byte of the Data Bus (AD0–AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.3). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The operation Word is swapped on the Data Bus, that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Address Mode fields within the Operation Word, the CPU starts fetching operand and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.3).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this $\overline{SPC}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Sec. 3.4.3).

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.3). This word has the format shown in *Figure 3-34.* If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the SLAVE vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.3).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.0 Functional Description (Continued)

Status Combinations:

Send ID (ID): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operaton Word.
3	OP	CPU Sends Required Operands
4	—	Slave Starts Execution. CPU Pre-fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-31. 16-Bit Slave Processor Protocol

3.9.2 32-Bit Fast Slave Protocol

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-32*. While applying Status code 1111 (Broadcast ID Sec. 3.4.2), the CPU transfers the ID Byte on bits AD24–AD31, the operation word on bits AD8–AD23 in a swapped order of bytes and a non-used byte xxxx0000 (x = don't care) on bit AD0–AD7 (*Figure 3-33*).

Using the addressing mode fields within the Operation word, the CPU fetches operands and sends them to the Slave Processor. Since the CPU is solely responsible for memory accesses, addressing mode extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand Sec. 3.4.2). After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SDONE}$ or $\overline{SPC}$ low for one clock cycle.

Unlike the old protocol, the SLAVE may request the CPU to read the status by activating the $\overline{SDONE}$ or $\overline{SPC}$ line for two clock cycles instead of one. The CPU will then read the slave status word and update the PSR Register, unless a trap is signalled. If this happens, the CPU will immediately abort the protocol and start a trap sequence using the SLAVE vector in the interrupt table.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills its queue before the Slave Processor finishes, the CPU will wait applying status code 0011 (waiting for Slave, Sec. 3.4.2).

Upon receiving the pulse on either $\overline{SDONE}$ or $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read the result from the Slave Processor and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

Status Combinations:

Send ID (IOD): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU sends ID and Operation Word.
2	OP	CPU sends required operands (if any).
3	—	Slave starts execution (CPU prefetches).*
4	—	Slave pulses $\overline{SDONE}$ or $\overline{SPC}$ low.
5	ST	CPU Reads Status word (only if $\overline{SDONE}$ or $\overline{SPC}$ pulse is two clock cycles wide).
6	OP	CPU Reads Results (if any).

FIGURE 3-32. 32-Bit Fast Slave Protocol

Certain Slave Processor instructions affect CPU PSR. For these instructions only the CPU will perform a Read Slave status cycle as described in 3.9.1.1 before reading the result. The relevent PSR bits will be loaded from the status word.

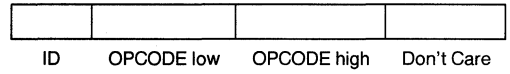


FIGURE 3-33. ID and Opcode Format for Fast Slave Protocol

3.9.3 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

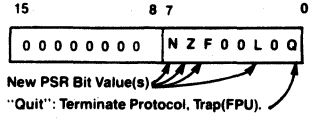
The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-34*).

3.0 Functional Description (Continued)

TABLE 3-4
Floating Point Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
REMf	read.f	rmw.f	f	f	f to Op.2	none
SQRTf	read.f	write.f	f	N/A	f to Op.2	none
POLYf	read.f	read.f	f	f	f to F0	none
DOTf	read.f	read.f	f	f	f to F0	none
SCALBf	read.f	rmw.f	f	f	f to Op.2	none
LOGBf	read.f	write.f	f	N/A	f to Op.2	none
ATAN2f	read.f	rmw.f	f	f	f to Op.2	none
SICOSf	read.f	rmw.f	f	N/A	f to Op.2 & F0	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note:
D = Double Word
i = Integer size (B,W,D) specified in mnemonic.
f = Floating Point type (F,L) specified in mnemonic.
N/A = Not Applicable to this instruction.



TL/EE/8673–44

FIGURE 3-34. Slave Processor Status Word Format

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

3.9.4 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Instruction Set Reference Manual and the MMU Data Sheet.

3.0 Functional Description (Continued)

TABLE 3-5
Memory Management Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Instruction Set Reference Manual and the Memory Management Unit Data Sheet.

D = Double Word

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.9.5 Custom Slave Instructions

Provided in the NS32332 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

3.0 Functional Description (Continued)

TABLE 3-6
Custom Slave Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL4c	read.c	rmw.c	c	c	c to Op.2	none
CCAL5c	read.c	rmw.c	c	c	c to Op.2	none
CCAL6c	read.c	rmw.c	c	c	c to Op.2	none
CCAL7c	read.c	rmw.c	c	c	c to Op.2	none
CCAL8c	read.c	read.c	c	c	c to c.reg	none
CCAL9c	read.c	read.c	c	c	c to c.reg	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op.2	none
CMOV4c	read.c	write.c	c	N/A	c to Op.2	none
CMOV5c	read.c	write.c	c	N/A	c to Op.2	none
CMOV6c	read.c	write.c	c	N/A	c to Op.2	none
CMOV7c	read.c	write.c	c	N/A	c to Op.2	none
CCMP0c	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to OP. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

Note:

D = Double Word

i = Integer size (B,W,D) specified in mnemonic.

c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.

* = Privileged instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 NS32332 PIN DESCRIPTIONS

The following is a brief description of all NS32332 pins. The descriptions reference portions of the Functional Description, Section 3.

Unless otherwise indicated, reserved pins should be left open.

4.1.1 Supplies

Logic Power $V_{CCL1, 2}$ +5V positive supply.

Buffers Power ($V_{CCB1, 2, 3, 4, 5}$): +5V positive supply.

Logic Ground (GNDL1, GNDL2): Ground reference for on-chip logic.

Buffer Grounds (GNDB1, GNDB2, GNDB3, GNDB4, GNDB5, GNDB6): Ground references for on-chip drivers.

Back Bias Generator (BBG): Output of on-chip substrate voltage generator.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals.

Ready (RDY): Active high. While RDY is not active, the CPU adds wait cycles to the current bus cycle. Not applicable for slave cycles.

Hold Request ($\overline{HOLD}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes.

Note: If the $\overline{HOLD}$ signal is generated asynchronously, it's set up and hold times may be violated. In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{HLDA}$ latency. This is to avoid speed degradations in cases of heavy $\overline{HOLD}$ activity (i.e. DMA controller cycles interleaved with CPU cycles.)

Interrupt ($\overline{INT}$): Active low. Maskable Interrupt request.

Non-Maskable Interrupt ($\overline{NMI}$): Active low. Non-Maskable Interrupt request.

Reset/Abort ($\overline{RST}$/$\overline{ABT}$): Active low. If held active for one clock cycle and released, this pin causes an ABORT. If held longer, it is interpreted as RESET.

Bus Error ($\overline{BER}$): Active low. When active, indicates that an error occurred during a bus cycle. It is treated by the CPU as the highest priority exception after RESET. Not applicable for slave cycles.

Bus Retry ($\overline{BRT}$): Active low, when active, the CPU will re-execute the last bus cycle. Not applicable for slave cycles.

Bus Width (BW1, BW0): Define the bus width (8, 16, 32) in every bus cycle. 01–8 bits, 10–16 bits, 11–32 bits. 00 is a reserved combination. Not applicable for slave cycles.

Burst in ($\overline{BIN}$): Active low. When active, the CPU may perform burst cycles.

Float ($\overline{FLT}$): Active low. Float command input. In non-memory managed systems, this pin should be tied to V_{CC} through a 10 kΩ resistor.

Data Timing/Slave Processor Control ($\overline{DT}$/$\overline{SDONE}$): Active low. Used by a 32-bit slave processor to acknowledge the completion of an instruction and/or indicate that the slave status should be read (Section 3.9.2). Sampled on the rising edge of reset to select the data timing during write cycles (Section 3.3).

4.1.3 Output Signals

Address Strobe ($\overline{ADS}$): Active low. Controls address latches, indicates the start of a bus cycle.

Data Direction in ($\overline{DDIN}$): Active low. Indicates the directions of data transfers.

Byte Enables ($\overline{BE0}$–$\overline{BE3}$): Active low. Enable the access of bytes 0–3 in a 32 bit system.

Status (ST0–ST3): Bus cycle status code, ST0 least significant. Encodings are:

0000 — Idle: CPU Inactive on Bus.
0001 — Idle: WAIT Instruction.
0010 — (Reserved).
0011 — Idle: Waiting for Slave.
0100 — Interrupt Acknowledge, Master.
0101 — Interrupt Acknowledge, Cascaded.
0110 — End of Interrupt, Master.
0111 — End of Interrupt, Cascaded.
1000 — Sequential Instruction Fetch.
1001 — Non-Sequential Instruction Fetch.
1010 — Data Transfer.
1011 — Read Read-Modify-Write Operand.
1100 — Read for Effective Address.
1101 — Transfer Slave Operand.
1110 — Read Slave Status Word.
1111 — Broadcast Slave ID.

Status Strobe ($\overline{STS}$): Active low. Indicates that a new status (ST0–ST3) is valid. Not applicable for slave cycles.

Multiple Cycle/Exception Status ($\overline{MC}$/$\overline{EXS}$): Active low. This signal is activated during the access of the first part of an operand that crosses a double word address boundary.

It is also activated during Abort Acknowledge and when a fatal bus error occurs.

Hold Acknowledge ($\overline{HLDA}$): Active low. Activated by the CPU in response to HOLD input. Indicates that the CPU has released the bus.

User/Supervisor (U/$\overline{S}$): User or Supervisor Mode status.

Interlocked Operation ($\overline{ILO}$): Active low. Indicates that an interlocked cycle is being performed.

Program Flow Status ($\overline{PFS}$): Active low. A pulse that indicates the beginning of an instruction execution.

Burst Out ($\overline{BOUT}$): Active low. When active, indicates that the CPU will perform burst cycles.

4.1.4 Input Output Signals

Address/Data 0–31 (AD0–AD31): Multiplexed address and data lines.

Slave Processor Control ($\overline{SPC}$): Active low. Used by the CPU as a data strobe output for slave processor transfers. Used by a 16-bit slave processor to acknowledge the completion of an instruction.

4.0 Device Specifications (Continued)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

4.2 ABSOLUTE MAXIMUM RATINGS

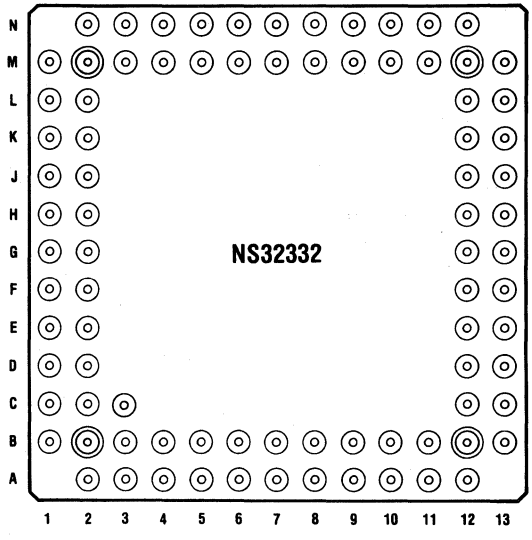
Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7V
Power Dissipation	3 Watt

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS T_A = 0° to +70°C, V_{CC} = 5V ±5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		V_{CC} +0.5	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	V_{CC} −0.5		V_{CC} +0.5	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CRT}	Clock Input Ringing Tolerance	PHI1, PHI2 pins only	−0.5		0.5	V
V_{OH}	Logical 1 Output Voltage	I_{OH} = −400 μA	2.4			V
V_{OL}	Logical 0 Output Voltage	I_{OL} = 2 mA			0.45	V
I_{ILS}	$\overline{SPC}$ Input Current (low)	V_{IN} = 0.4V, $\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	0 ≤ V_{IN} ≤ V_{CC}, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current (Output pins in TRI-STATE condition)	0.4 ≤ V_{OUT} ≤ V_{CC}	−20		20	μA
I_{CC}	Active Supply Current	I_{OUT} = 0, T_A = 25°C		320	450	mA

Connection Diagram*



Bottom View

NS32332 Pinout Descriptions
84 Pin Grid Array

Desc	Pin	Desc	Pin	Desc	Pin
GNDB1	B1	AD29	N6	$\overline{BOUT}$	E12
AD6	B2	AD30	M6	$\overline{SPC}$	D13
AD7	C1	AD31	N7	$\overline{MC}/\overline{EXS}$	D12
AD8	C2	VCCL1	M7	VCCB5	C13
AD9	D1	VCCL2	N8	$\overline{ADS}$	C12
AD10	D2	$\overline{INT}$	M8	GNDB6	B13
AD11	E1	$\overline{NMI}$	N9	$\overline{DDIN}$	A12
GNDB2	E2	*RESERVED	M9	$\overline{BE0}$	B12
AD12	F1	*RESERVED	N10	$\overline{BE1}$	A11
AD13	F2	*RESERVED	M10	$\overline{BE2}$	B11
AD14	G1	*RESERVED	N11	$\overline{BE3}$	A10
AD15	G2	$\overline{ILO}$	M11	$\overline{HLDA}$	B10
VCCB2	H1	VCCB4	N12	$\overline{HOLD}$	A9
AD16	H2	ST3	M13	RDY	B9
AD17	J1	ST2	M12	$\overline{DT}/\overline{SDONE}$	A8
AD18	J2	ST1	L13	PHI 2	B8
AD19	K1	ST0	L12	PHI 1	A7
GNDB3	K2	$\overline{STS}$	K13	BBG	B7
AD20	L1	GNDB5	K12	GNDL2	A6
AD21	L2	$\overline{PFS}$	J13	GNDL1	B6
AD22	M1	U/$\overline{S}$	J12	VCCB1	A5
AD23	N2	BW1	H13	AD0	B5
VCCB3	M2	BW0	H12	AD1	A4
AD24	N3	$\overline{BIN}$	G13	AD2	B4
AD25	M3	$\overline{FLT}$	G12	AD3	A3
AD26	N4	$\overline{RST}/\overline{ABT}$	F13	AD4	B3
AD27	M4	$\overline{BRT}$	F12	AD5	A2
GNDB4	N5	BER	E13	POSITION PIN	C3
AD26	M5				

FIGURE 4-1. Pin Grid Array Package

* AMP sockets are recommended for use with NS32332 CPU. AMP sockets are manufactured by AMP INCORPORATED, Harrisburg PA.

4.0 Device Specifications (Continued)

4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 and PHI2 and 0.8V or 2.0V on all other signals as illustrated below, unless specifically stated otherwise.

ABBREVIATIONS:

L.E. — leading edge
T.E. — trailing edge
R.E. — rising edge
F.E. — falling edge

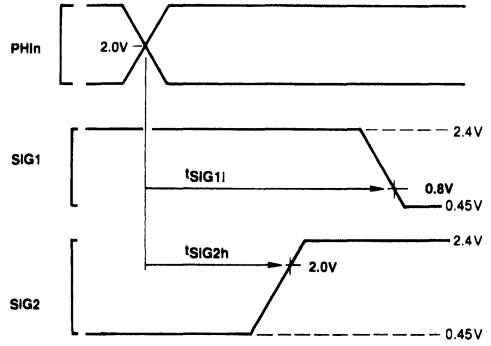


FIGURE 4-2. Timing Specification Standard (Signal Valid After Clock Edge)

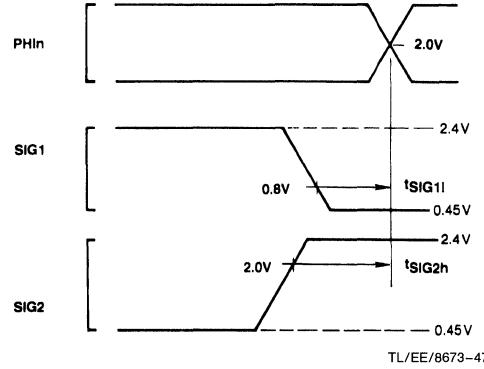


FIGURE 4-3. Timing Specification Standard (Signal Valid Before Clock Edge)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32332-10, NS32332-12, NS32332-15*

Maximum times assume capacitive loading of 100 pF.

***15 MHz Timing is Advance Information**

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-12		NS32332-15*		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4-5	Address bits 0–31 valid	after R.E., PHI1 T1		50				35	ns
t_{ALh}	4-5	Address bits 0–31 hold	after R.E., PHI1 T2/Tmmu	5				5		ns
t_{Dv}	4-5	Data valid (write cycle)	after R.E., PHI1 T3 or T2		50				38	ns
t_{Dh}	4-5	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0				0		ns
t_{ALADSs}	4-4	Address bits 0–31 setup	before $\overline{ADS}$ reaches 2.0V	25				20		ns
t_{ALADSh}	4-17	Address bits 0–23 hold	after $\overline{ADS}$ reaches 2.0V	10				10		ns
t_{ALf}	4-4	Address bits 0–31 floating (no MMU)	after R.E., PHI1 T2/Tmmu		25				18	ns
t_{ALMf}	4-17	Address bits 0–31 floating (by $\overline{FLT}$ line)	after R.E., PHI1 T2/Tmmu		25				23	ns
t_{STSa}	4-3,4-5	$\overline{STS}$ signal active (low)	after R.E., PHI1 T4 of previous bus cycle or Ti		35				25	ns
t_{STSia}	4-3,4-5	$\overline{STS}$ signal inactive	after R.E., PHI2 T4 of previous bus cycle or Ti		45				34	ns
t_{STSw}	4-3	$\overline{STS}$ pulse width	at 0.8V (both edges)	35				24		ns
t_{BErv}	4-4,4-6	$\overline{BEn}$ signals valid (Operand Read Cycles Only)	after R.E., PHI2, T4 or Ti		140				95	ns
t_{BEv}	4-5,4-6	$\overline{BEn}$ signals valid	after R.E., PHI2, T4 or Ti		85				58	ns
t_{BEh}	4-4	$\overline{BEn}$ signals hold	after R.E., PHI2, T4	0				0		ns
t_{STv}	4-5	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		50				35	ns
t_{STh}	4-5	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0				0		ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32332-10, NS32332-12, NS32332-15* (Continued)

***15 MHz Timing is Advance Information**

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-12		NS32332-15*		Units
				Min	Max	Min	Max	Min	Max	
t_{DDINv}	4-4	$\overline{DDIN}$ signal valid	after R.E., PHI1 T1		35				25	ns
t_{DDINh}	4-4	$\overline{DDIN}$ signal hold	after R.E., PHI1 next T1 or Ti	0				0		ns
t_{ADSa}	4-5	$\overline{ADS}$ signal active (low)	after R.E., PHI1 T1		35				26	ns
t_{ADSia}	4-5	$\overline{ADS}$ signal inactive	after R.E., PHI2 T1		45				29	ns
t_{ADSw}	4-5	$\overline{ADS}$ pulse width	at 0.8V (both edges)	35				24		ns
t_{MCa}	4-1,4-2	$\overline{MC}$ signal active (low)	after R.E., PHI1 T1		50				38	ns
t_{MCia}	4-1,4-2	$\overline{MC}$ signal inactive	after R.E., PHI1 T1 or T3 (burst)		50				38	ns
t_{ALf}	4-14	AD0–AD31 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		25				18	ns
t_{ADSf}	4-14, 4-15	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55				40	ns
t_{BEf}	4-14, 4-15	$\overline{BEn}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55				40	ns
t_{DDINf}	4-14, 4-15	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55				40	ns
t_{HLDAa}	4-14, 4-15	$\overline{HLDA}$ signal active (low)	after R.E., PHI1 T4		75				55	ns
t_{HLDAia}	4-16	$\overline{HLDA}$ signal inactive	after R.E., PHI1 Ti		75				55	ns
t_{ADSr}	4-16	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55				40	ns
t_{BEr}	4-16	$\overline{BEn}$ signals return from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55				40	ns
t_{DDINr}	4-16	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55				40	ns
t_{DDINf}	4-17	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		50				38	ns
t_{DDINr}	4-18	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		40				28	ns
t_{SPCa}	4-19	$\overline{SPC}$ output active (low)	after R.E., PHI1 T1		30				21	ns
t_{SPCia}	4-19	$\overline{SPC}$ output inactive	after R.E., PHI1 T4	2	35			2	26	ns
t_{SPCnf}	4-22	$\overline{SPC}$ output nonforcing	after R.E., PHI2 T4		10				8	ns
t_{Dv}	4-19	Data valid (slave processor write)	after R.E., PHI1 T1		50				38	ns
t_{Dh}	4-19	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0				0		ns
t_{PFSw}	4-24	$\overline{PFS}$ pulse width	at 0.8V (both edges)	70				45		ns
t_{PFSa}	4-24	$\overline{PFS}$ pulse active (low)	after R.E., PHI2		50				38	ns
t_{PFSia}	4-24	$\overline{PFS}$ pulse inactive	after R.E., PHI2		50				38	ns
t_{USv}	4-31	U/$\overline{S}$ signal valid	after R.E., PHI1 T4		48				35	ns
t_{USh}	4-31	U/$\overline{S}$ signal hold	after R.E., PHI1 T4	10				6		ns
t_{NSPF}	4-26	Nonsequential fetch to next $\overline{PFS}$ clock cycle	after R.E., PHI1 T1	4				3		t_{Cp}
t_{PFNS}	4-25	$\overline{PFS}$ clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4				3		t_{Cp}
t_{STSf}	4-14, 4-15	$\overline{STS}$ floating (HOLD)	after R.E., PHI1 Ti		55				40	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32332-10, NS32332-12, NS32332-15* (Continued)

***15 MHz Timing is Advance Information**

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-12		NS32332-15*		Units
				Min	Max	Min	Max	Min	Max	
t_{STSr}	4-16	$\overline{STS}$ not floating (HOLD)	after R.E., PHI1 Ti, T4		55				40	ns
t_{BOUTa}	4-4, 4-6, 4-9	$\overline{BOUT}$ output active	after R.E., PHI2 Tmmu		100				66	ns
t_{BOUTia}	4-4, 4-6, 4-9	$\overline{BOUT}$ output inactive	after R.E., PHI2 T3 or T4		75				50	ns
t_{ILOa}	4-13	$\overline{ILO}$ signal active	after R.E., PHI1 T4		50				38	ns
t_{ILOia}	4-13	$\overline{ILO}$ signal inactive	after R.E., PHI1 Ti		50				38	ns

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

4.4.2.2 Input Signal Requirements: NS32332-10, NS32332-12, NS32332-15*

***15 MHz Timing is Advance Information**

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-12		NS32332-15*		Units
				Min	Max	Min	Max	Min	Max	
t_{PWR}	4-29	Power stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50				33		μs
t_{DIs}	4-4	Data in setup (read cycle)	before F.E., PHI2 T3	10				7		ns
t_{DIh}	4-4	Data in hold (read cycle)	after R.E., PHI1 T4	10				6		ns
t_{HLDa}	4-14, 4-15	$\overline{HOLD}$ active (low) setup time (see note)	before F.E., PHI2 T2/Tmmu or T3 or Ti	25				17		ns
t_{HLDia}	4-16	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25				17		ns
t_{HLDh}	4-14, 4-15, 4-16	$\overline{HOLD}$ hold time	after R.E., PHI1 Ti or T3	0				0		ns
t_{FLTa}	4-17	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25				17		ns
t_{FLTia}	4-18	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T3	25				17		ns
t_{RDYs}	4-4, 4-5, 4-6	RDY setup time	before F.E., PHI1 T3	25				17		ns
t_{RDYh}	4-4, 4-5, 4-6	RDY hold time	after F.E., PHI2 T3	0				0		ns
t_{ABTs}	4-27	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 T2/Tmmu	20				13		ns
t_{ABTs}	4-28	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 Tf	20				13		ns
t_{ABTh}	4-27, 4-28	$\overline{ABT}$ hold time	after R.E., PHI1 T3	0				0		ns
t_{RSTs}	4-29, 4-30	$\overline{RST}$ setup time	before F.E., PHI1	20				13		ns
t_{RSTw}	4-29, 4-30	$\overline{RST}$ pulse width	at 0.8V (both edges)	64				64		t_{Cp}
t_{INTs}	4-32	$\overline{INT}$ setup time	before F.E., PHI1	20				13		ns
t_{NMIw}	4-33	$\overline{NMI}$ pulse width	at 0.8V (both edges)	40				27		ns
t_{DIs}	4-22	Data setup (slave read cycle)	before F.E., PHI2 T1	10				7		ns
t_{DIh}	4-22	Data hold (slave read cycle)	after R.E., PHI1 T4	10				7		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements: NS32332-10, NS32332-12, NS32332-15* (Continued)

***15 MHz Timing is Advance Information**

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-12		NS32332-15*		Units
				Min	Max	Min	Max	Min	Max	
t_{DTs}	4-29	$\overline{DT}$ setup time	before $\overline{RST}$ R.E.	90				60		ns
t_{DTh}	4-29	$\overline{DT}$ hold time	after $\overline{RST}$ R.E.	50				33		ns
t_{SPCd}	4-22	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	10				8		ns
t_{SPCs}	4-22	$\overline{SPC}$ setup time	before F.E., PHI1	25				15		ns
t_{SPCw}	4-22	$\overline{SPC}$ pulse width	at 0.8V (both edges)	20				13		ns
t_{SDNd}	4-21	$\overline{SDONE}$ pulse delay from slave	after R.E., PHI2 T4	10				8		ns
t_{SDNs}	4-21	$\overline{SDONE}$ setup time	before F.E., PHI1	25				15		ns
t_{SDNw}	4-21	$\overline{SDONE}$ pulse width	at 0.8V (both edges)	20				13		ns
t_{SDNSTw}	4-21	$\overline{SDONE}$ pulse width (to force CPU to read slave status)	at 0.8V (both edges)	175				115		ns
t_{BWs}	4-4, 4-5 4-6	$\overline{BW}$ 0–1 setup time	before F.E., PHI1 T3	25				17		ns
t_{BWh}	4-4, 4-6	$\overline{BW}$0-1 hold time	after R.E., PHI1 T4	0				0		ns
t_{BINs}	4-6	$\overline{BIN}$ setup time (for each cycle of the burst)	before F.E., PHI1 T3	25				17		ns
t_{BINh}	4-6	$\overline{BIN}$ hold time	after R.E., PHI1 T4	0				0		ns
t_{BERs}	4-11, 4-12	$\overline{BER}$ setup time	before F.E., PHI1 T4	25				17		ns
t_{BERh}	4-11, 4-12	$\overline{BER}$ hold time	after R.E., PHI1 Ti	0				0		ns
t_{BRTs}	4-7, 4-8, 4-9, 4-10	$\overline{BRT}$ setup time	before F.E., PHI1 T3 and T4	25				17		ns
t_{BRTh}	4-7, 4-9	$\overline{BRT}$ Hold Time	after R.E., PHI1 Ti	0				0		ns

Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

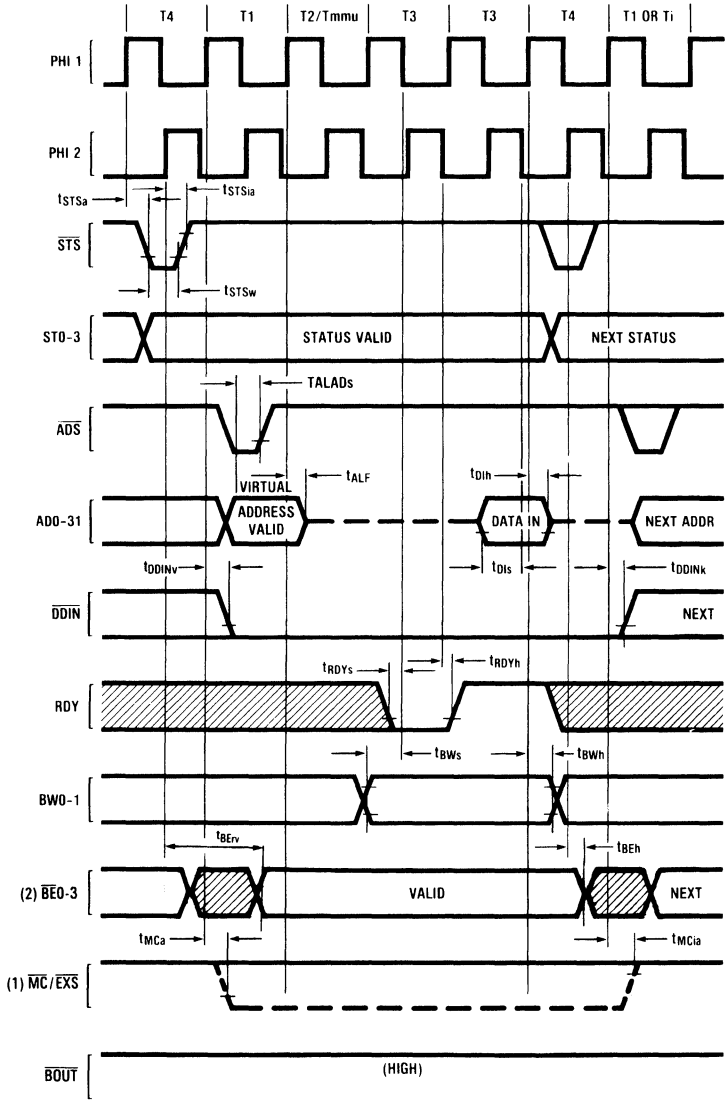
4.4.2.3 Clocking Requirements: NS32332-10, NS32332-12, NS32332-15*

***15 MHz Timing is Advance Information**

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-12		NS32332-15*		Units
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4-23	PHI1, PHI2 rise time	0.8V to V_{CC}–0.9V On R.E., PHI1, PHI2		7				4	ns
t_{CLf}	4-23	PHI1, PHI2 fall time	V_{CC}–0.9V to 0.8V On F.E., PHI1, PHI2		7				4	ns
t_{Cp}	4-23	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	100	5000			66	1000	ns
$t_{CLw(1,2)}$	4-23	PHI1, PHI2 Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	0.5 t_{cp} − 6 ns				0.5 t_{cp} − 10 ns		
$t_{CLh(1,2)}$	4-23	PHI1, PHI2 high time	At V_{CC}–0.9V on PHI1, PHI2 (Both Edges)	0.5 t_{cp} − 15 ns				0.5 t_{cp} − 10 ns		
$t_{nOVL(1,2)}$	4-23	Non-overlap time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7			0	6	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4			−3	3	ns
t_{CLhas}		PHI1, PHI2 asymmetry ($t_{CLh(1)} - t_{CLh(2)}$)	At V_{CC}–0.9V on PHI1, PHI2	−5	5			−3	3	ns

4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams

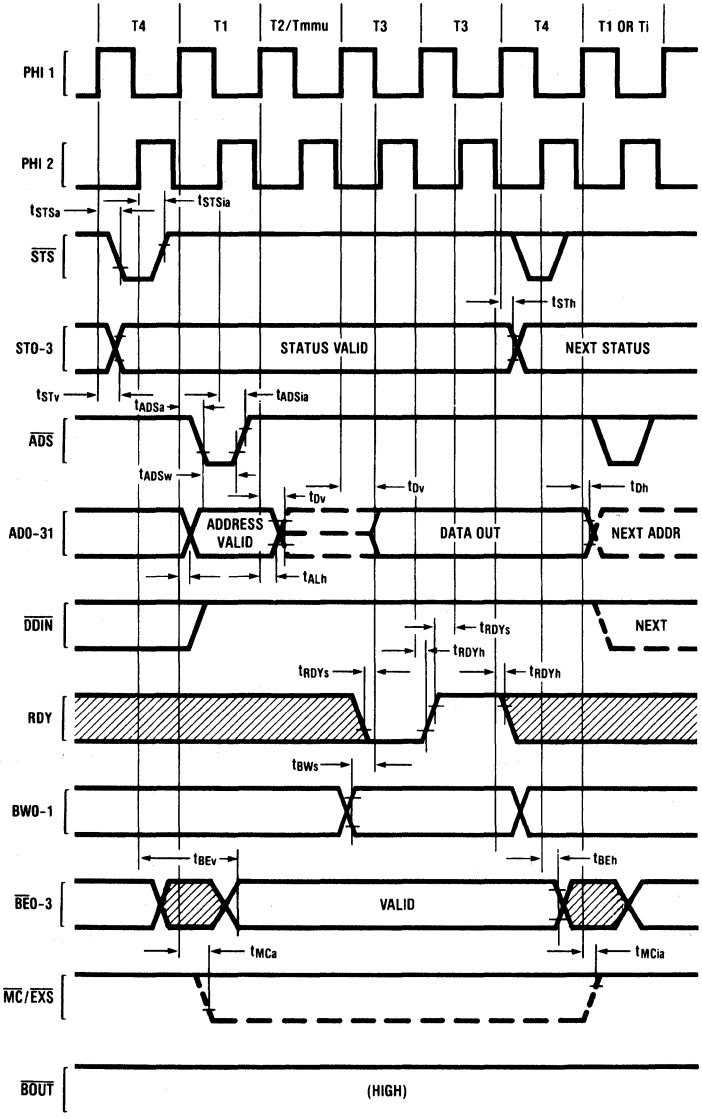


TL/EE/8673-48

Note 1: Asserted (low) when the bus transaction crosses a double-word boundary (address bits A0–1 wrap around during the transaction).
Note 2: $\overline{BE0}$–$\overline{BE3}$ are all active during instruction fetch cycles.

FIGURE 4-4. NS32332 Read Cycle Timing

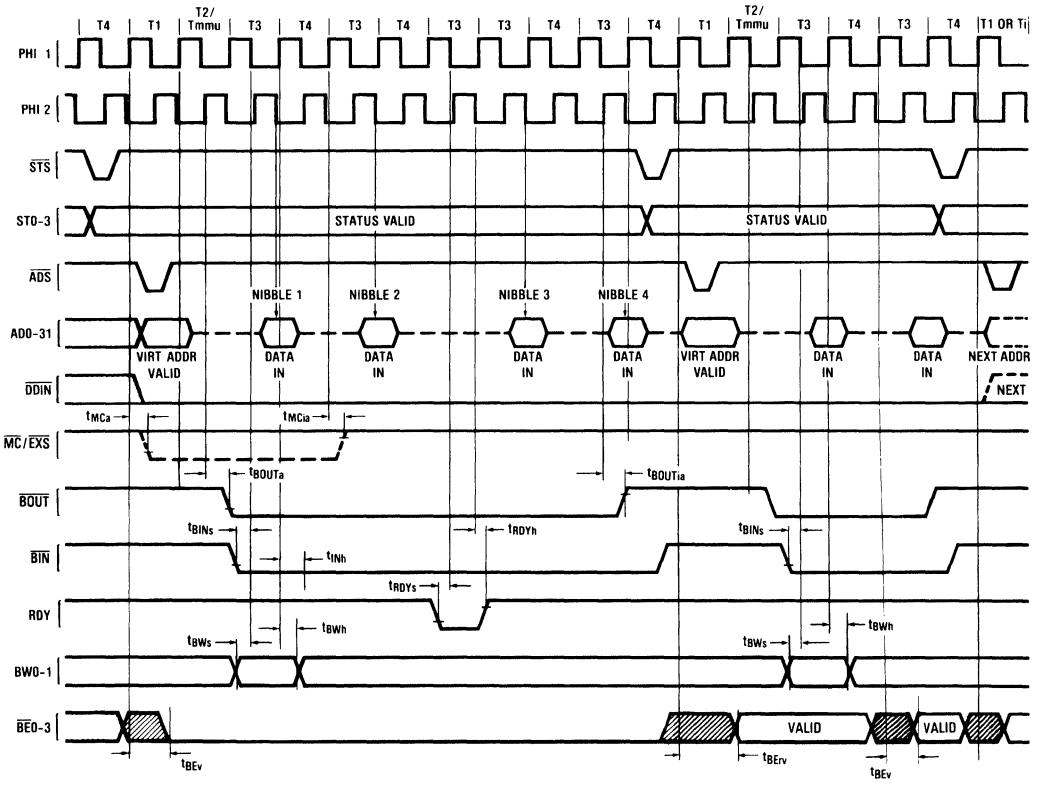
4.0 Device Specifications (Continued)



TL/EE/8673-49

FIGURE 4-5. NS32332 Write Cycle Timing

4.0 Device Specifications (Continued)

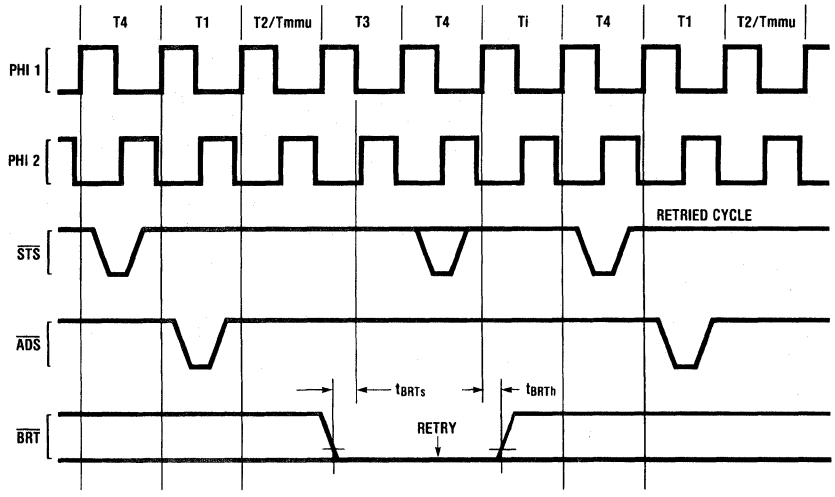


TL/EE/8673-50

FIGURE 4-6. NS32332 Burst Cycle Timing
(Instruction fetches followed by Operand Reads)

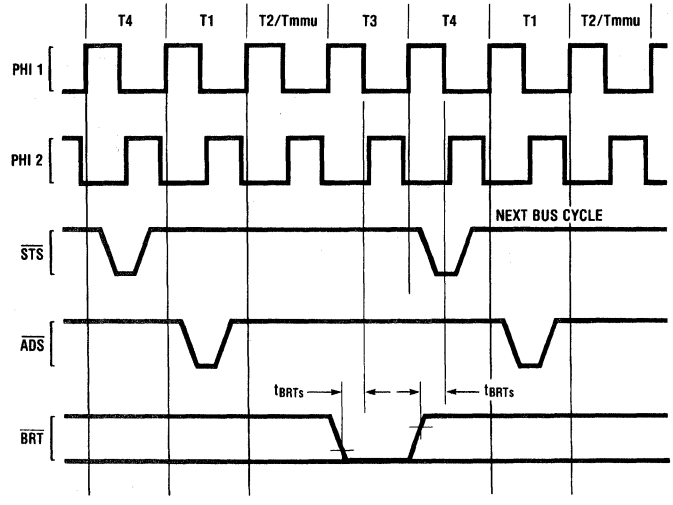
2

4.0 Device Specifications (Continued)



TL/EE/8673-51

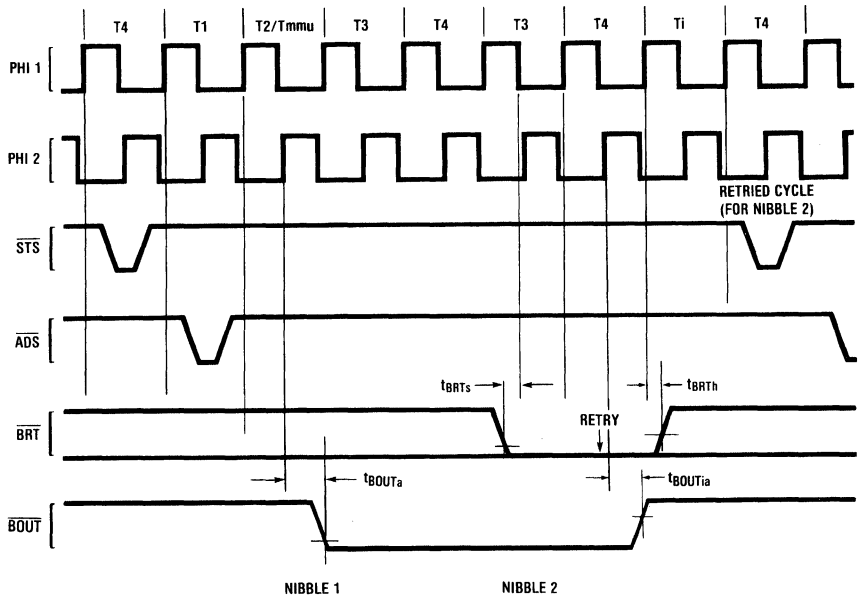
FIGURE 4-7. Bus Retry During Normal Bus Cycle



TL/EE/8673-52

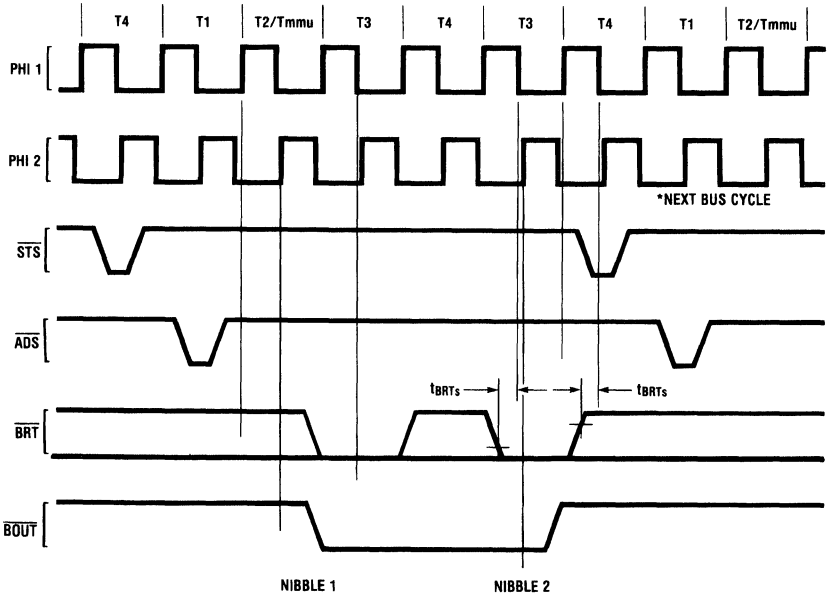
FIGURE 4-8. $\overline{BRT}$ Activated, but no Bus Retry

4.0 Device Specifications (Continued)



TL/EE/8673-53

FIGURE 4-9. Bus Retry During Burst Bus Cycle

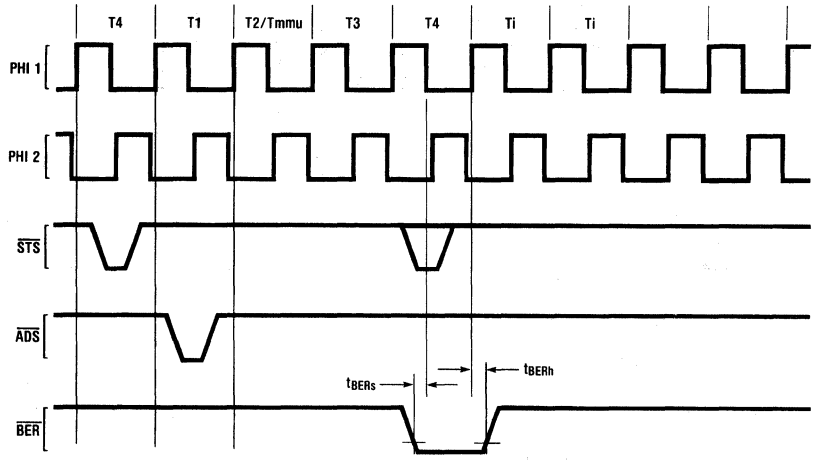


TL/EE/8673-54

*The next bus cycle is a normal bus cycle used to read data otherwise read as nibble 3 of the burst cycle (if $\overline{BRT}$ not activated) or other data.

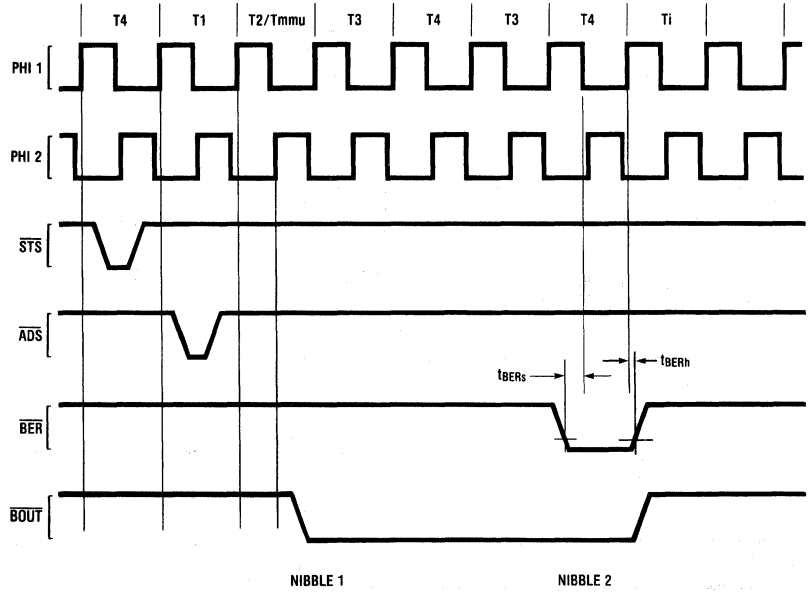
FIGURE 4-10. $\overline{BRT}$ Activated During Burst Bus Cycle, but no Bus Retry

4.0 Device Specifications (Continued)



TL/EE/8673-55

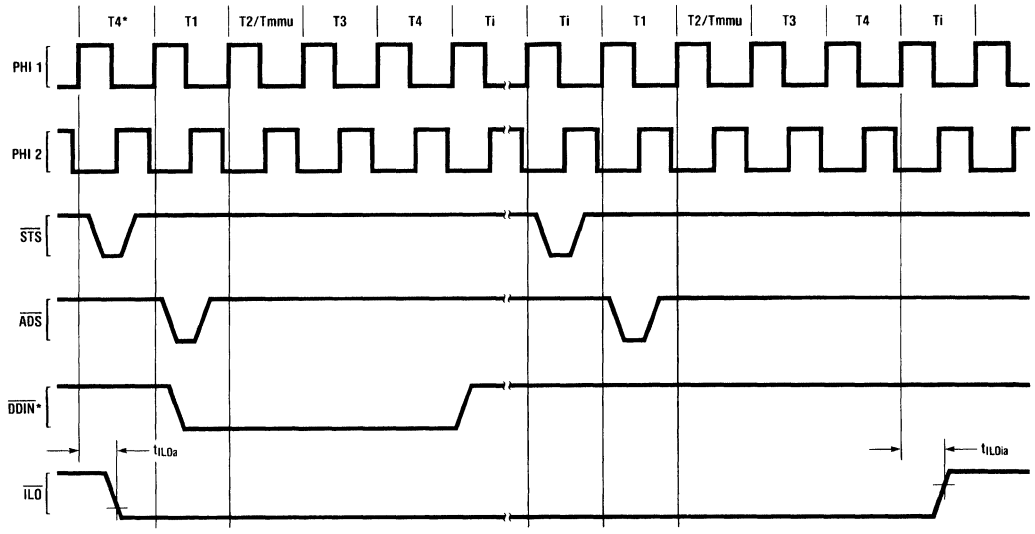
FIGURE 4-11. Bus Error During Normal Bus Cycle



TL/EE/8673-56

FIGURE 4-12. Bus Error During Burst Bus Cycle

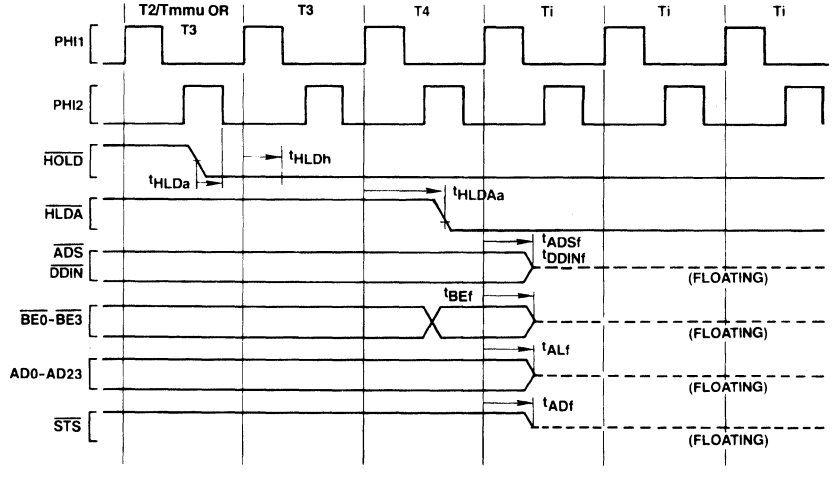
4.0 Device Specifications (Continued)



*End of Dummy Read cycle with the address of the interlocked operand.

TL/EE/8673-57

FIGURE 4-13. Timing of Interlocked Bus Transactions

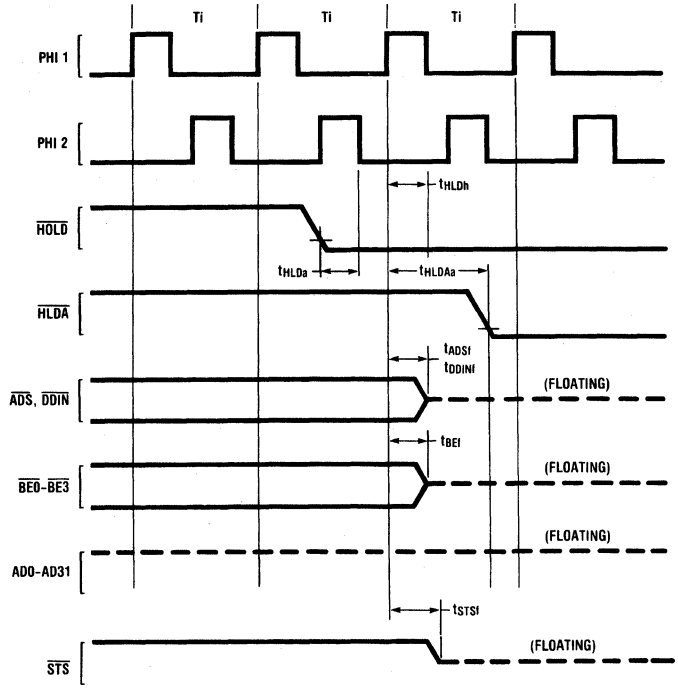


TL/EE/8673-58

FIGURE 4-14. Floating by HOLD Timing (CPU Not Idle Initially)

Note that whenever the CPU is not idling (not in Ti), the HOLD request (HOLD low) must be active tHLDa before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until tHLDh after the rising edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.

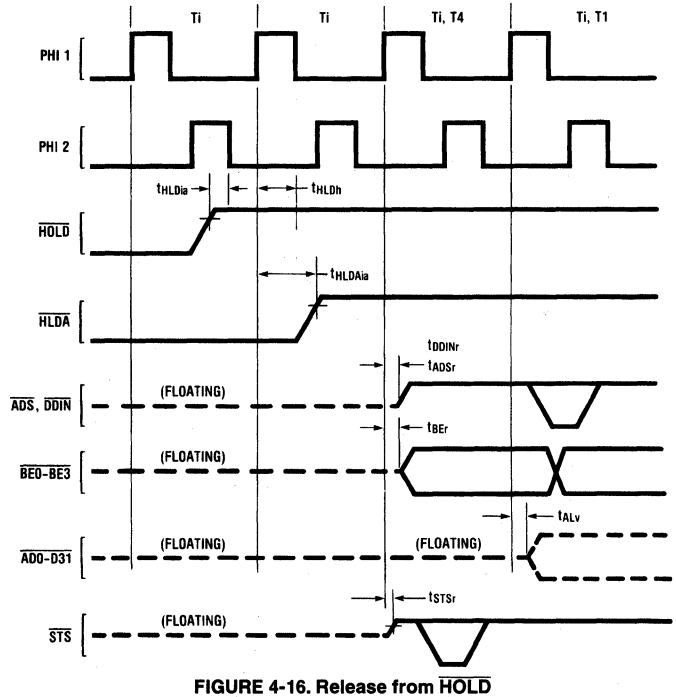
4.0 Device Specifications (Continued)



TL/EE/8673-59

FIGURE 4-15. Floating by $\overline{\text{HOLD}}$ Timing (CPU initially idle)

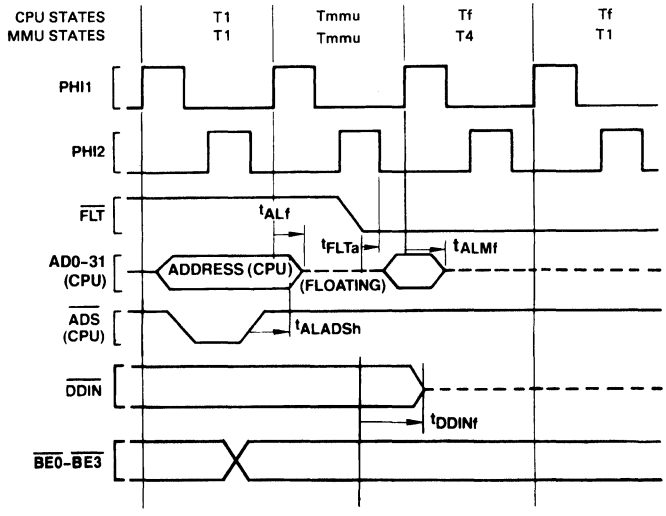
Note that during Ti1 the CPU is already idling.



TL/EE/8673-60

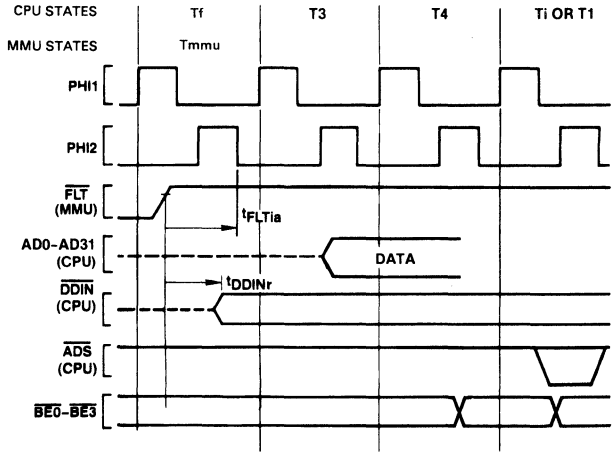
FIGURE 4-16. Release from $\overline{\text{HOLD}}$

4.0 Device Specifications (Continued)



TL/EE/8673-61

FIGURE 4-17. FLT Initiated Cycle Timing

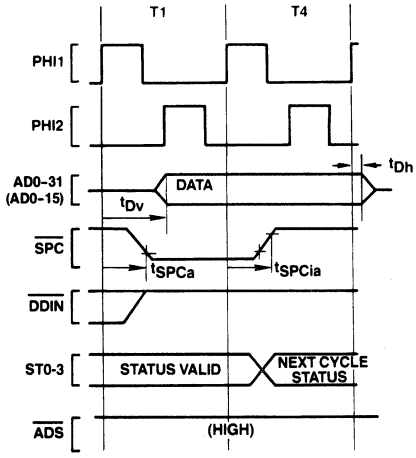


TL/EE/8673-62

FIGURE 4-18. Release from FLT Timing

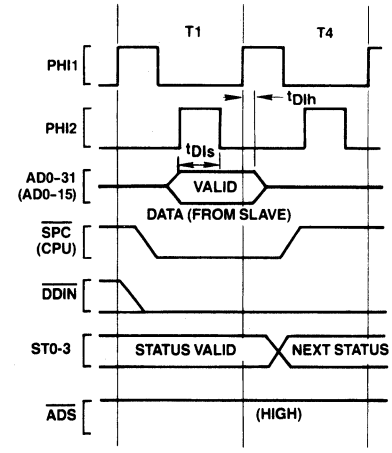
Note that when FLT is deasserted the CPU restarts driving DDIN before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force DDIN to the same logic level.

4.0 Device Specifications (Continued)



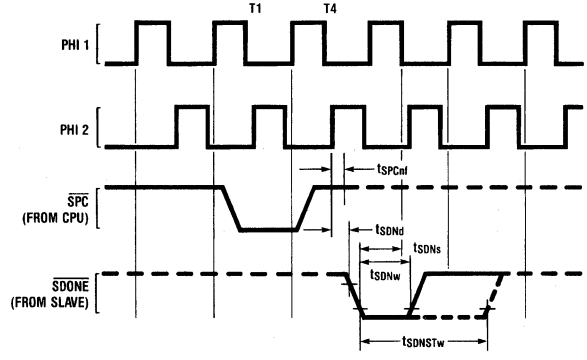
TL/EE/8673-64

FIGURE 4-19. Slave Processor Write Timing



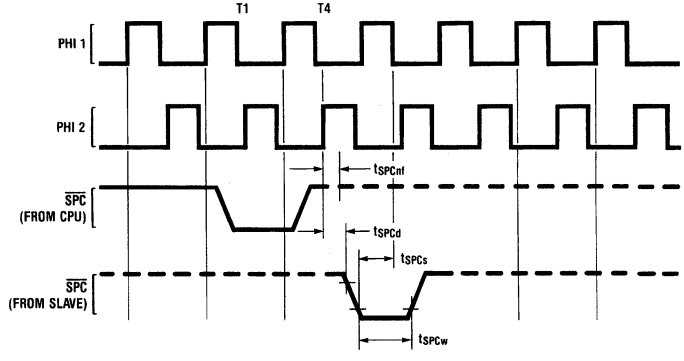
TL/EE/8673-65

FIGURE 4-20. Slave Processor Read Timing



TL/EE/8673-63

FIGURE 4-21. $\overline{DT}/\overline{SDONE}$ Timing (32-Bit Slave Protocol)

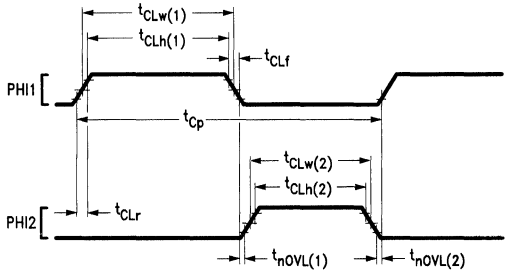


TL/EE/8673-66

FIGURE 4-22. $\overline{SPC}$ Timing (16-Bit Slave Protocol)

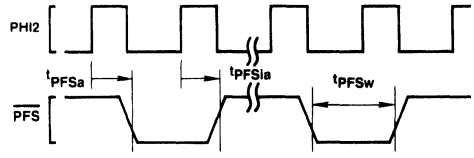
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{SPC}$ high with internal 5 kΩ pullup.

4.0 Device Specifications (Continued)



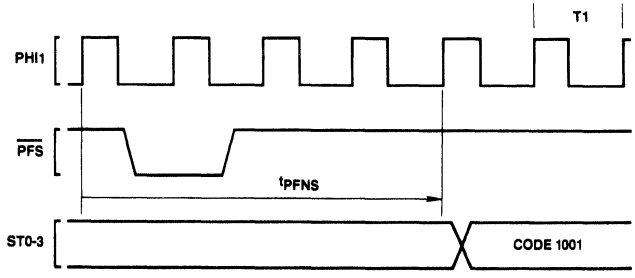
TL/EE/8673-67

FIGURE 4-23. Clock Waveforms



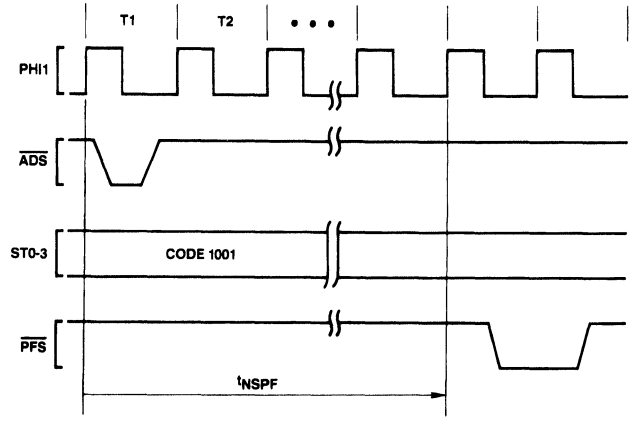
TL/EE/8673-68

FIGURE 4-24. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/8673-69

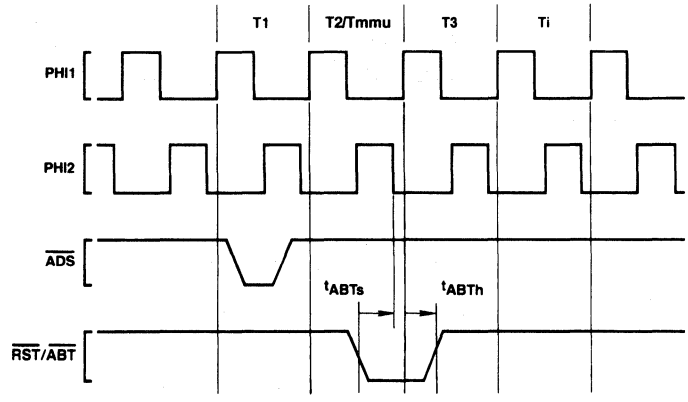
FIGURE 4-25. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/8673-70

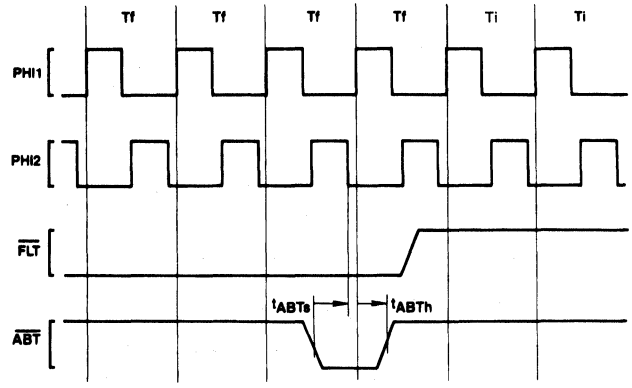
FIGURE 4-26. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



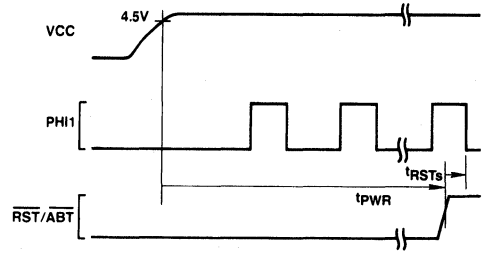
TL/EE/8673-71

FIGURE 4-27. Abort Timing, $\overline{FLT}$ Not Applied



TL/EE/8673-72

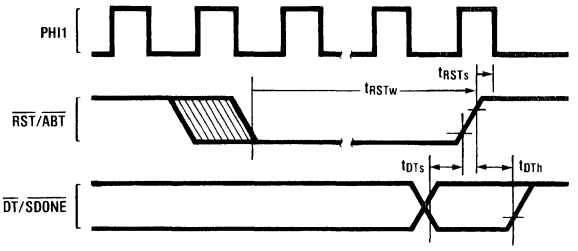
FIGURE 4-28. Abort Timing, $\overline{FLT}$ Applied



TL/EE/8673-73

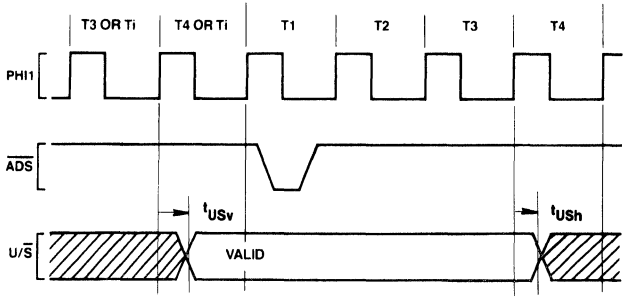
FIGURE 4-29. Power-On Reset

4.0 Device Specifications (Continued)



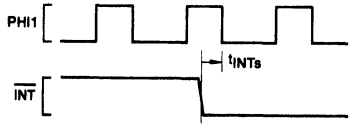
TL/EE/8673-74

FIGURE 4-30. Non-Power-On Reset



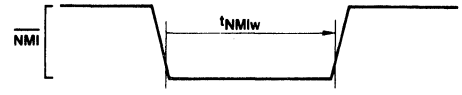
TL/EE/8673-75

FIGURE 4-31. U/S Relationship to Any Bus Cycle — Guaranteed Valid Interval



TL/EE/8673-76

FIGURE 4-32. INT Interrupt Signal Detection



TL/EE/8673-77

FIGURE 4-33. NMI Interrupt Signal Timing

2

Appendix A: Instruction Formats

NOTATIONS

i= Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f= Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c= Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op= Operation Code
Valid encodings shown with each format.

gen, gen 1, gen 2= General Addressing Mode Field
See Sec. 2.2 for encodings.

reg= General Purpose Register Number

cond= Condition Code Field
- 0000 = EQual: Z = 1
- 0001 = Not Equal: Z = 0
- 0010 = Carry Set: C = 1
- 0011 = Carry Clear: C = 0
- 0100 = HIgher: L = 1
- 0101 = Lower or Same: L = 0
- 0110 = Greater Than: N = 1
- 0111 = Less or Equal: N = 0
- 1000 = Flag Set: F = 1
- 1001 = Flag Clear: F = 0
- 1010 = LOwer: L = 0 and Z = 0
- 1011 = Higher or Same: L = 1 or Z = 1
- 1100 = Less Than: N = 0 and Z = 0
- 1101 = Greater or Equal: N = 1 or Z = 1
- 1110 = (Unconditionally True)
- 1111 = (Unconditionally False)

short= Short Immediate value. May contain
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.
  - 0000 = US
  - 0001 – 0111 = (Reserved)
  - 1000 = FP
  - 1001 = SP
  - 1010 = SB
  - 1011 = (Reserved)
  - 1100 = (Reserved)
  - 1101 = PSR
  - 1110 = INTBASE
  - 1111 = MOD
- Options: in String Instructions

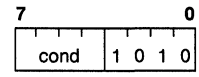
U/W	B	T

  - T = Translated
  - B = Backward
  - U/W = 00: None
    - 01: While Match
    - 11: Until Match

Configuration bits, in SETCFG:

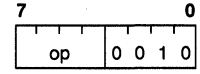
P	FC	FM	FF	C	M	F	I

mreg: MMU Register number, in LMR, SMR.
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = PF0
- 0101 = PF1
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = SC
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA



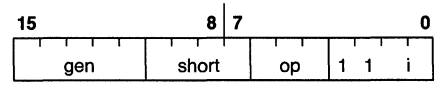
Format 0

Bcond (BR)



Format 1

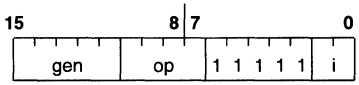
BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111



Format 2

ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Scond	-011		

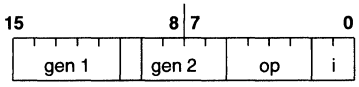
Appendix A: Instruction Formats (Continued)



Format 3

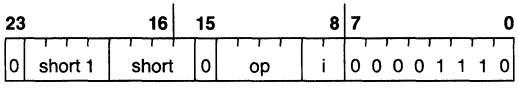
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

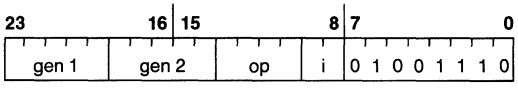
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

MOVS	-0000	SETCFG*	-0010
CMPS	-0001	SKPS	-0011

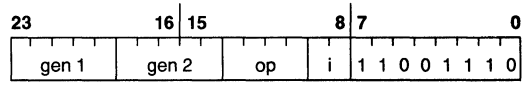
Trap (UND) on 1XXX, 01XX



Format 6

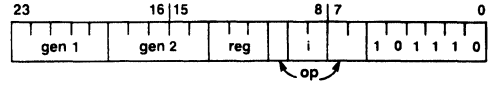
ROT	-0000	NEG	-1000
ASH	-0001	NOT	-1001
CBIT	-0010	Trap (UND)	-1010
CBITI	-0011	SUBP	-1011
Trap (UND)	-0100	ABS	-1100
LSH	-0101	COM	-1101
SBIT	-0110	IBIT	-1110
SBITI	-0111	ADDP	-1111

*Short 1 in format 5 applies only for SETCFG instruction. In other instructions this field is 0.



Format 7

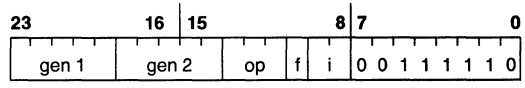
MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZiD	-0110	MOD	-1110
MOVXiD	-0111	DIV	-1111



TL/EE/8673-78

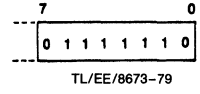
Format 8

EXT	-0 00	INDEX	-1 00
CVTP	-0 01	FFS	-1 01
INS	-0 10		
CHECK	-0 11		
MOVSU	-110, reg = 001		
MOVUS	-110, reg = 011		



Format 9

MOVif	-000	ROUND	-100
LFSR	-001	TRUNC	-101
MOVLF	-010	SFSR	-110
MOVFL	-011	FLOOR	-111

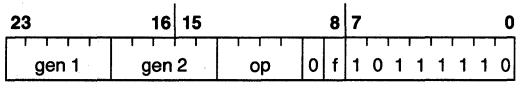


TL/EE/8673-79

Format 10

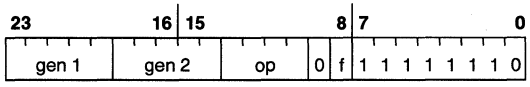
Trap (UND) Always

Appendix A: Instruction Formats (Continued)



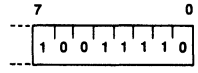
Format 11

ADDf	-0000	DIVf	-1000
MOVf	-0001	Trap (SLAVE)	-1001
CMPf	-0010	Trap (UND)	-1010
Trap (SLAVE)	-0011	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



Format 12

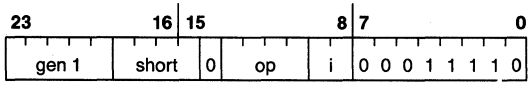
REMf	-0000	Trap (SLAVE)	-1000
SQRTf	-0001	Trap (SLAVE)	-1001
POLYf	-0010	Trap (UND)	-1010
DOTf	-0011	Trap (UND)	-1011
SCALBf	-0100	ATAN2f	-1100
LOGBf	-0101	SICOSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



TL/EE/8673-81

Format 13

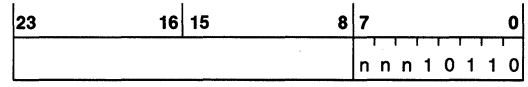
Trap (UND) Always



Format 14

RDVAL	-0000	LMR	-0010
WRVAL	-0001	SMR	-0011

Trap (UND) on 01XX, 1XXX

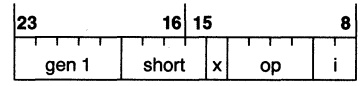


Operation Word ID Byte

Format 15

(Custom Slave)

nnn **Operation Word Format**

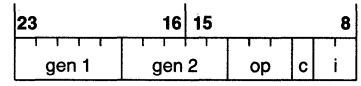


000

Format 15.0

CATST0	-0000	LCR	-1010
CATST1	-0001	SCR	-1011

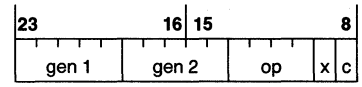
Trap (UND) on all others



001

Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111

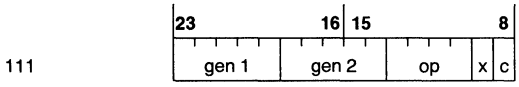


101

Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	CMOV3	-1001
CCMP0	-0010	Trap (UND)	-1010
CCMP1	-0011	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

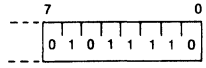
Appendix A: Instruction Formats (Continued)



Format 15.7

CCAL4	-0000	CCAL7	-1000
CMOV4	-0001	CMOV7	-1001
CCAL8	-0010	Trap (UND)	-1010
CCAL9	-0011	Trap (UND)	-1011
CCAL5	-0100	CCAL6	-1100
CMOV6	-0101	CMOV5	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

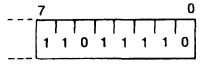
If nnn = 010, 011, 100, 110 then Trap (UND) Always.



TL/EE/8673-82

Format 16

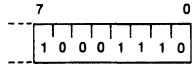
Trap (UND) Always



TL/EE/8673-83

Format 17

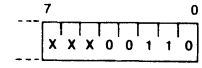
Trap (UND) Always



TL/EE/8673-84

Format 18

Trap (UND) Always

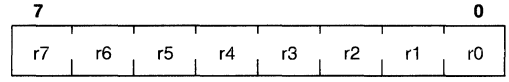


TL/EE/8673-85

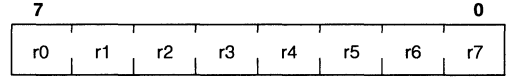
Format 19

Trap (UND) Always

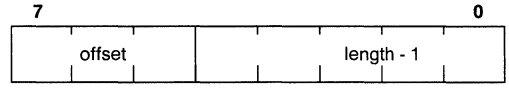
Implied Immediate Encodings:



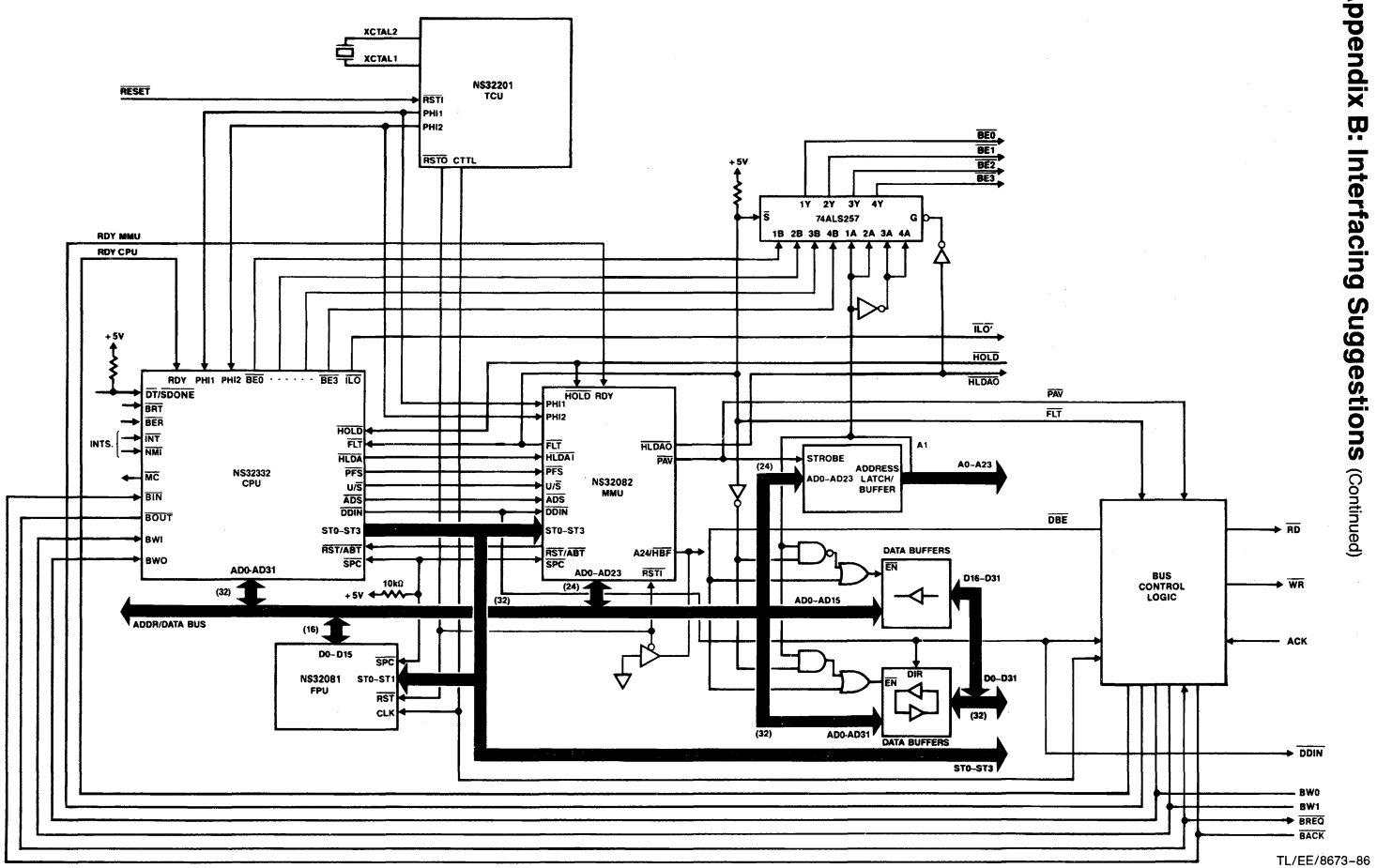
Register Mark, appended to SAVE, ENTER



Register Mark, appended to RESTORE, EXIT



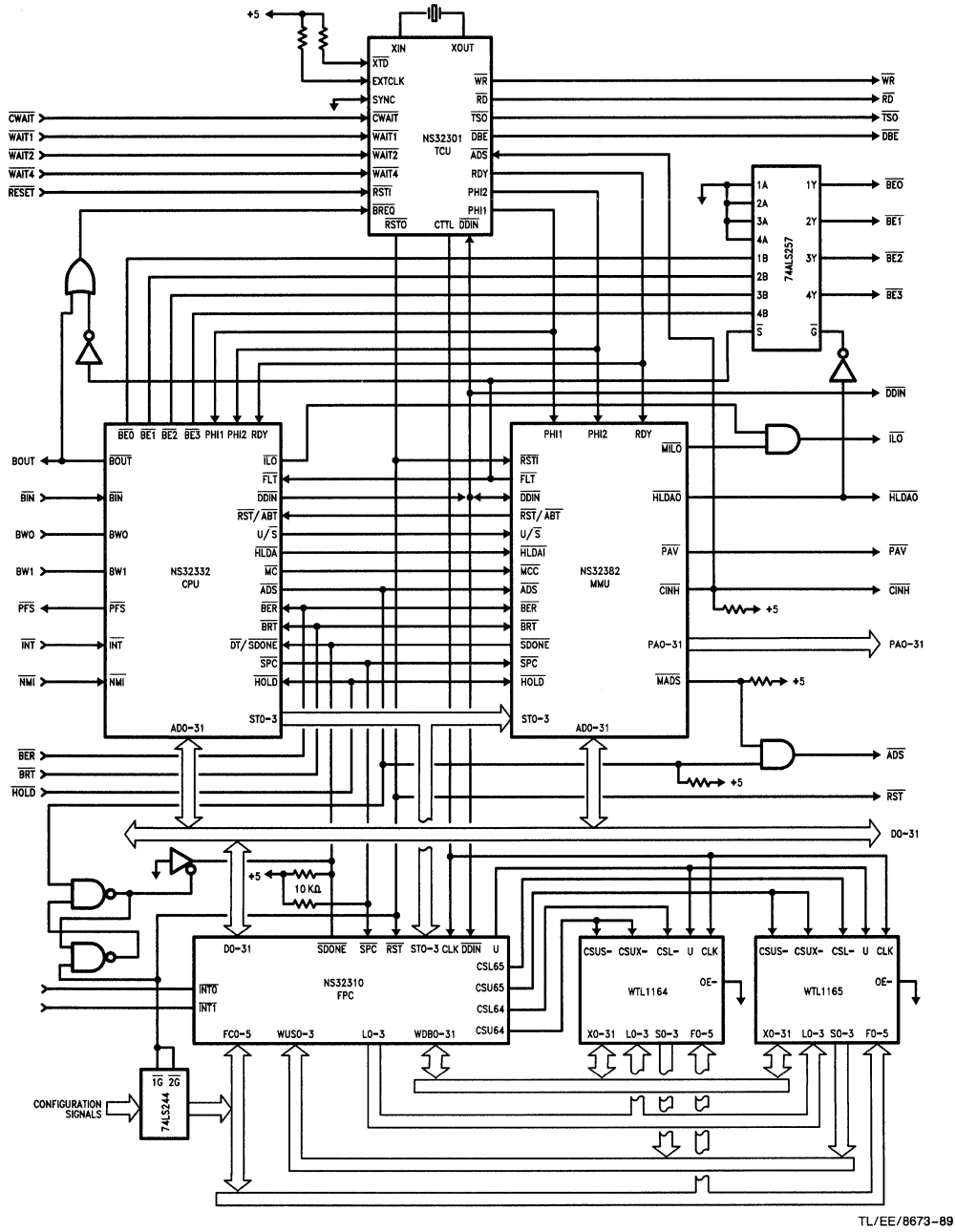
Offset/Length Modifier appended to INSS, EXTS



TL/EE/8673-86

FIGURE B-1. System Connection Diagram (32332, 32081 & 32082)

Appendix B: Interfacing Suggestions (Continued)



TL/EE/8673-89

FIGURE B-2. System Connection Diagram (32332, 32310 & 32382)

2

National Semiconductor Corporation

PRELIMINARY

NS32132-6/NS32132-8/NS32132-10 High-Performance Microprocessors

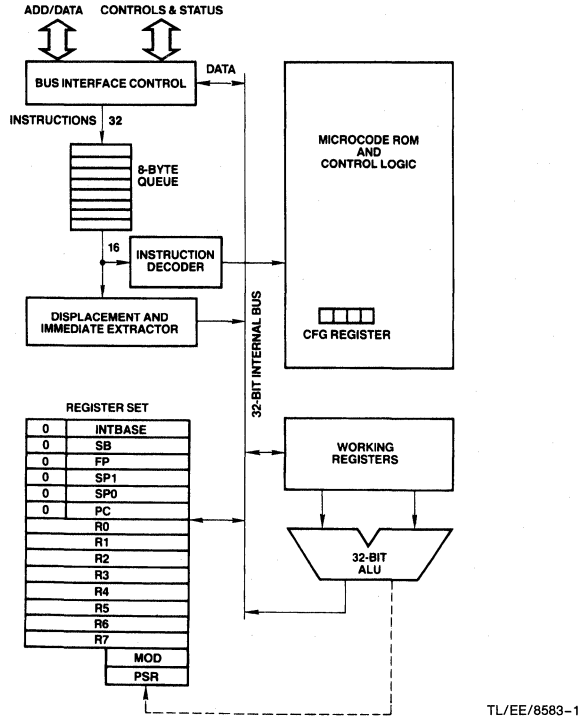
General Description

The NS32132 is a 32-bit, virtual memory microprocessor with a 16-MByte linear address space and a 32-bit external data bus. It has a 32-bit ALU, eight 32-bit general purpose registers, an eight-byte prefetch queue, and a slave processor interface. A dual processing capability is also provided that allows two NS32123 to share the same bus and memory with a resulting performance enhancement of up to 80%. The NS32132 is fabricated with National Semiconductor's advanced XMOS process, and is fully object code compatible with other Series 32000 processors. The Series 32000 instructions set is optimized for modular high-level languages (HLL). The set is very symmetric, it has a two address format, and it incorporates HLL oriented addressing modes. The capabilities of the NS32132 can be expanded with the use of the NS32081 floating point unit (FPU), and the NS32032 demand-paged virtual memory management unit (MMU). Both devices interface to the NS32132 as slave processors. The NS32132 is a general purpose microprocessor that is ideal for a wide range of computational intensive applications.

Features

- 32-bit Architecture and Implementation
- Virtual Memory Support
- 16-MByte Linear Addressing Space
- 32-bit Data Bus
- Tightly Coupled Dual Processing Support
- Powerful Instruction Set
  - — General 2-Address Capability
  - — High Degree of Symmetry
  - — Addressing Modes Optimized for High Level Language
- Series 32000® Slave Processor Support
- High-Speed XMOS™ Technology
- 68-pin Leadless Chip Carrier

Block Diagram



TL/EE/8583-1

Table of Contents

1.0 PRODUCT INTRODUCTION

2.0 ARCHITECTURAL DESCRIPTION

2.1 Programming Model
- 2.1.1 General Purpose Registers
- 2.1.2 Dedicated Registers
- 2.1.3 The Configuration Register (CFG)
- 2.1.4 Memory Organization
- 2.1.5 Dedicated Tables

2.2 Instruction Set
- 2.2.1 General Instruction Format
- 2.2.2 Addressing Modes
- 2.2.3 Instruction Set Summary

3.0 FUNCTIONAL DESCRIPTION

3.1 Power and Grounding

3.2 Clocking

3.3 Resetting

3.4 Bus Cycles
- 3.4.1 Cycle Extension
- 3.4.2 Bus Status
- 3.4.3 Data Access Sequences
  - 3.4.3.1 Bit Accesses
  - 3.4.3.2 Bit Field Accesses
  - 3.4.3.3 Extending Multiply Accesses
- 3.4.4 Instruction Fetches
- 3.4.5 Interrupt Control Cycles
- 3.4.6 Slave Processor Communication
  - 3.4.6.1 Slave Processor Bus Cycles
  - 3.4.6.2 Slave Operand Transfer Sequences

3.5 Memory Management Option
- 3.5.1 Address Translation Strap
- 3.5.2 Translated Bus Timing
- 3.5.3 The $\overline{FLT}$ (Float) Pin
- 3.5.4 Aborting Bus Cycles
  - 3.5.4.1 The Abort Interrupt
  - 3.5.4.2 Hardware Considerations

3.6 Bus Access Control

3.7 Dual Processing
- 3.7.1 Bus Arbitration
- 3.7.2 Processor Assignment

3.0 FUNCTIONAL DESCRIPTION (Continued)

3.8 Instruction Status

3.9 NS32132 Interrupt Structure
- 3.9.1 General Interrupt/Trap Sequence
- 3.9.2 Interrupt/Trap Return
- 3.9.3 Maskable Interrupts (The $\overline{INT}$ Pin)
  - 3.9.3.1 Non-Vectored Mode
  - 3.9.3.2 Vectored-Mode: Non-Cascaded Case
  - 3.9.3.3 Vectored Mode: Cascaded Case
- 3.9.4 Non-Maskable Interrupt (The $\overline{NMI}$ Pin)
- 3.9.5 Traps
- 3.9.6 Prioritization
- 3.9.7 Interrupt/Trap Sequences: Detailed Flow
  - 3.9.7.1 Maskable/Non-Maskable Interrupt Sequences
  - 3.9.7.2 Trap Sequence: Traps Other than Trace
  - 3.9.7.3 Trace Trap Sequence
  - 3.9.7.4 Abort Sequence

3.10 Slave Processor Instructions
- 3.10.1 Slave Processor Protocol
- 3.10.2 Floating Point Instructions
- 3.10.3 Memory Management Instructions
- 3.10.4 Custom Slave Instructions

4.0 DEVICE SPECIFICATIONS

4.1 Pin Descriptions
- 4.1.1 Supplies
- 4.1.2 Input Signals
- 4.1.3 Output Signals
- 4.1.4 Input/Output Signals

4.2 Absolute Maximum Ratings

4.3 Electrical Characteristics

4.4 Switching Characteristics
- 4.4.1 Definitions
- 4.4.2 Timing Tables
  - 4.4.2.1 Output Signals: Internal Propagation Delays
  - 4.4.2.2 Input Signals Requirements
  - 4.4.2.3 Clocking Requirements
- 4.4.3 Timing Diagrams

Appendix A: Instruction Formats

B: Interfacing Suggestions

2

List of Illustrations

The General and Dedicated Registers 2-1
Processor Status Register 2-2
CFG Register 2-3
Module Descriptor Format 2-4
A Sample Link Table 2-5
General Instruction Format 2-6
Index Byte Format 2-7
Displacement Encodings 2-8
Recommended Supply Connections 3-1
Clock Timing Relationships 3-2
Power-on Reset Requirements 3-3
General Reset Timing 3-4
Recommended Reset Connections, Non-Memory-Managed System 3-5a
Recommended Reset Connections, Memory-Managed System 3-5b
Bus Connections 3-6
Read Cycle Timing 3-7
Write Cycle Timing 3-8
RDY Pin Timing 3-9
Extended Cycle Example 3-10
Memory Interface 3-11
Slave Processor Connections 3-12
CPU Read from Slave Processor 3-13
CPU Write to Slave Processor 3-14
Read Cycle With Address Translation (CPU Action) 3-15
Write Cycle With Address Translation (CPU Action) 3-16
Memory-Managed Read Cycle 3-17
Memory-Managed Write Cycle 3-18
$\overline{\text{FLT}}$ Timing 3-19
$\overline{\text{HOLD}}$ Timing, Bus Initially Idle 3-20
$\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle 3-21
Dual Processor System Basic Interconnections 3-22
Dual Processor Bus Arbitration Timing 3-23
Instruction Sequence To Assign CPUs to Different Tasks 3-24
Interrupt Dispatch and Cascade Tables 3-25
Interrupt/Trap Service Routine Calling Sequence 3-26
Return from Trap ($\overline{\text{RETT}}$ n) Instruction Flow 3-27
Return from Interrupt ($\overline{\text{RET}}$) Instruction Flow 3-28
Interrupt Control Connections (16 levels) 3-29
Cascaded Interrupt Control Unit Connections 3-30
Service Sequence 3-31
Slave Processor Protocol 3-32
Slave Processor Status Word Format 3-33
NS32132 Connection Diagram 4-1
Timing Specification Standard (Signal Valid After Clock Edge) 4-2
Timing Specification Standard (Signal Valid Before Clock Edge) 4-3
Write Cycle 4-4
Read Cycle 4-5
Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially) 4-6
Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle) 4-7
Release from $\overline{\text{HOLD}}$ 4-8

List of Illustrations (Continued)

$\overline{FLT}$ Initiated Float Cycle Timing 4-9
Release from $\overline{FLT}$ Timing 4-10
Ready Sampling (CPU Initially READY) 4-11
Ready Sampling (CPU Initially NOT READY) 4-12
Slave Processor Write Timing 4-13
Slave Processor Read Timing 4-14
$\overline{SPC}$ Timing 4-15
Reset Configuration Timing 4-16
Clock Waveforms 4-17
Relationship of $\overline{PFS}$ to Clock Cycles 4-18
Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch 4-19a
Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$ 4-19b
Relationship of $\overline{ILO}$ to First Operand of an Interlocked Instruction 4-20a
Relationship of $\overline{ILO}$ to Last Operand of an Interlocked Instruction 4-20b
Relationship of $\overline{ILO}$ to Any Clock Cycle 4-21
U/$\overline{S}$ Relationship to any Bus Cycle - Guaranteed Valid Interval 4-22
Abort Timing, $\overline{FLT}$ Not Applied 4-23
Abort Timing, $\overline{FLT}$ Applied 4-24
Power-On Reset 4-25
Non-Power-On Reset 4-26
$\overline{INT}$ Interrupt Signal Detection 4-27
$\overline{NMI}$ Interrupt Signal Timing 4-28
Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction 4-29
Dual Processor Bus Arbitration Timing 4-30

Single Processor System Connection Diagram B-1
Dual Processor System Connection Diagram B-2

List of Tables

NS32132 Addressing Modes 2-1
NS32132 Instruction Set Summary 2-2
Bus Access Types 3-1
Access Sequences 3-2
Interrupt Sequences 3-3
Floating Point Instruction Protocols 3-4
Memory Management Instruction Protocols 3-5
Custom Slave Instruction Protocols 3-6

1.0 Product Introduction

The Series 32000 Microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operation. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes. Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types. The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set. While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations. The Series 32000 CPUs represent two-operand machines with each operand addressable by all addressing modes. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management. Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing. The NS32132 has 24-bit address pointers that can address up to 16 megabytes without requiring any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support. Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept. The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32132 CPU.

2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

2.1.2 Dedicated Registers

The eight dedicated registers of the NS32132 are assigned specific functions.

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32132 the upper eight bits of this register are always zero.)

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 the SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32132 the upper eight bits of these registers are always zero).

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.

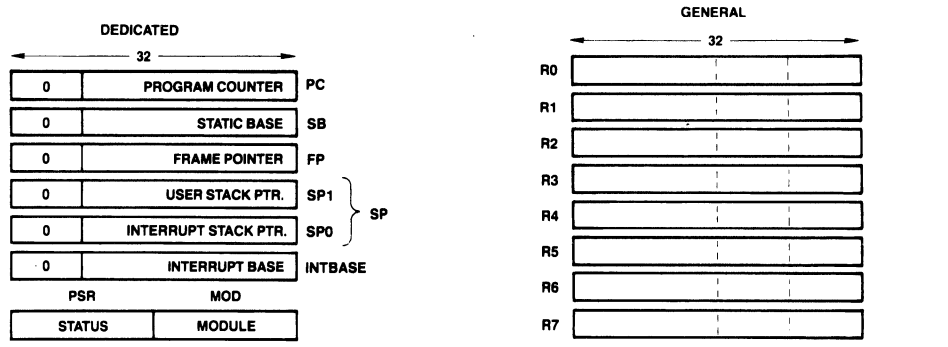
FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32132 the upper eight bits of this register are always zero.)

SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32132 the upper eight bits of this register are always zero.)

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32132 the upper eight bits of this register are always zero.)

2.0 Architectural Description (Continued)



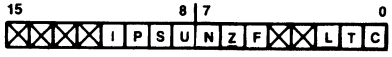
TL/EE/8583-3

FIGURE 2-1. The General and Dedicated Registers

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

PSR: The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the NS32132 microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/8583-4

FIGURE 2-2. Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Sec. 3.8.5).

L: The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the NS32132 is said to be in Supervisor Mode; when U = 1 the NS32132 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Sec. 3.8.5.). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I = 1, then all interrupts will be accepted (Sec. 3.8.). If I = 0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32132 CPU is the four-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3*.

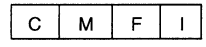


FIGURE 2-3. CFG Register

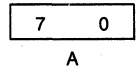
The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{INT}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

2.0 Architectural Description (Continued)

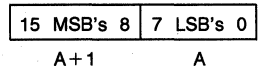
2.1.4 Memory Organization

The main memory of the NS32132 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at $2^{24} - 1$. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.



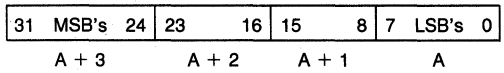
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Sec. 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.



Word at Address A

Two contiguous words are called a double word. Except where noted (Sec. 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



Double Word at Address A

Although memory is addressed as bytes, it is actually organized as double-words. Note that access time to a word or a double-word depends upon its address, e.g. double-words that are aligned to start at addresses that are multiples of four will be accessed more quickly than those not so aligned. This also applies to words that cross a double-word boundary.

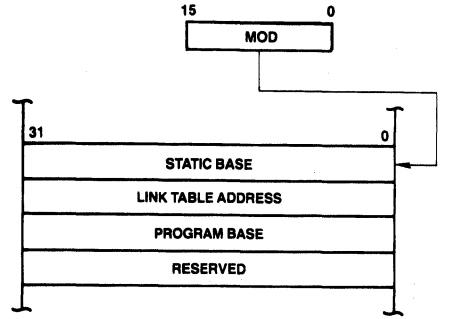
2.1.5 Dedicated Tables

Two of the NS32132 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Sec. 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by NS32132. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically up-dated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in *Figure 2-4*. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.



TL/EE/8583–5

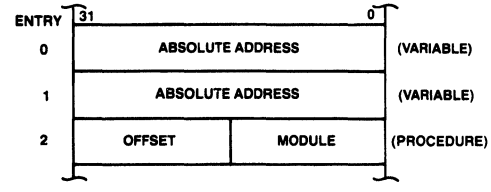
FIGURE 2-4. Module Descriptor Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-5*. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/8583–6

FIGURE 2-5. A Sample Link Table

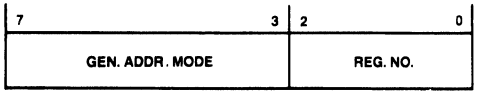
2.0 Architectural Description (Continued)

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-7*.



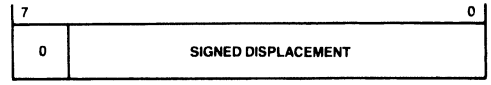
TL/EE/8583-8

FIGURE 2-7. Index Byte Format

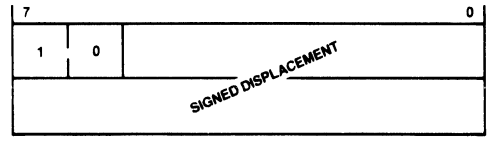
Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected address modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded with the top bits of that field, as shown in *Figure 2-8*, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first. Note that this is different from the memory representation of data (Sec. 2.1.4).

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Sec. 2.2.3).

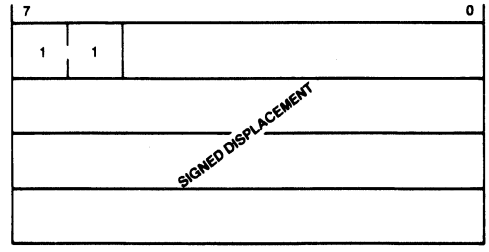
Byte Displacement: Range −64 TO +63



Word Displacement: Range −8192 TO +8191



Double Word Displacement: Range (Entire Addressing Space)

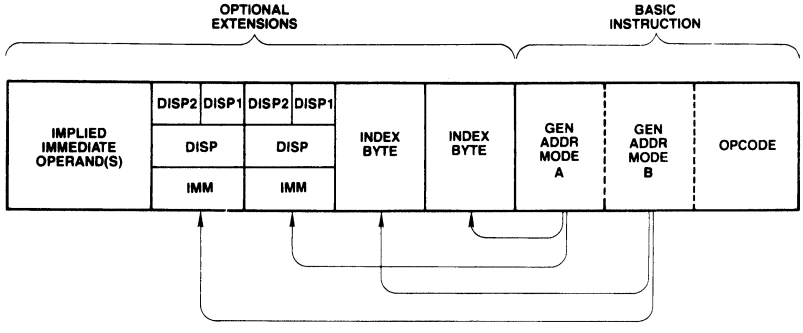


TL/EE/8583-9

FIGURE 2-8. Displacement Encodings

2.2.2 Addressing Modes

The NS32132 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."



TL/EE/8583-7

FIGURE 2-6. General Instruction Format

2.0 Architectural Description (Continued)

Addressing modes in the NS32132 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32132 Addressing Modes fall into nine basic types:

Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

Memory Space. Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode. Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32132 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating-Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0–R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

2.0 Architectural Description (Continued)

TABLE 2-1
NS32132 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	*+disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2× Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4× Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn. 'Mode' and 'n' are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

TABLE 2-2

NS32132 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move Effective Address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDI	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to Extended Integer.
7	DEIi	gen,gen	Divide Extended Integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add Packed.
6	SUBPi	gen,gen	Subtract Packed.

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare Multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical Exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

2.0 Architectural Description (Continued)

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical Shift, left or right.
6	ASHi	gen,gen	Arithmetic Shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to Bit Field Pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 - Comparison Value

R3 - Translation Table Pointer

R2 - String 2 Pointer

R1 - String 1 Pointer

R0 - Limit Count

Options on all string instructions are:

B (Backward): Decrement string pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Descriptions
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare translating, String 1 bytes.
5	SKPSi	options	Skip over String 1 entries
	SKPST	options	Skip, translating bytes for Until/While.

2.0 Architectural Description (Continued)

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor Call.
1	FLAG		Flag Trap.
1	BPT		Breakpoint Trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save General Purpose Registers.
1	RESTORE	[reg list]	Restore General Purpose Registers.
2	LPRi	areg,gen	Load Dedicated Register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store Dedicated Register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust Stack Pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set Configuration Register. (Privileged)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a Floating Point value.
9	MOVLF	gen,gen	Move and shorten a Long value to Standard.
9	MOVFL	gen,gen	Move and lengthen a Standard value to Long.
9	MOVif	gen,gen	Convert any integer to Standard or Long Floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load Memory Management Register. (Privileged)
14	SMR	mreg,gen	Store Memory Management Register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUi	gen,gen	Move a value from Supervisor Space to User Space. (Privileged)
8	MOVUSi	gen,gen	Move a value from User Space to Supervisor Space. (Privileged)

2.0 Architectural Description (Continued)

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No Operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom Calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom Move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
	CMOV3c	gen,gen	
15.5	CCMP0c	gen,gen	Custom Compare.
	CCMP1c	gen,gen	
15.1	CCV0ci	gen,gen	Custom Convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load Custom Status Register.
15.1	SCSR	gen	Store Custom Status Register.
15.0	CATST0	gen	Custom Address/Test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load Custom Register. (Privileged)
15.0	SCR	creg,gen	Store Custom Register. (Privileged)

3.0 Functional Description

3.1 POWER AND GROUNDING

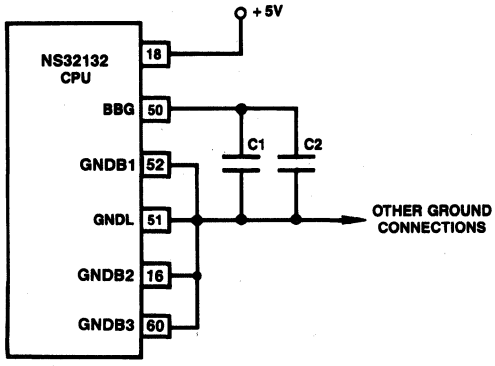
The NS32132 requires a single 5-volt power supply, applied on pin 18 (V_{CC}).

Grounding connections are made on four pins. Logic Ground (GNDL, pin 51) is the common pin for on-chip logic, and Buffer Grounds (GNDB1, pin 52, GNDB2, pin 16 and GNDB3, Pin 60) are the common pins for the output drivers. For optimal noise immunity it is recommended that GNDB1 and GNDB2 be connected together through a single conductor, and GNDL be directly connected to the middle point of this conductor. All other ground connections should be made to the common line as shown in *Figure 3-1*.

In addition to V_{CC} and Ground, the NS32132 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Fig. 3-1*) from the BBG pin to ground. Recommended values for these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.



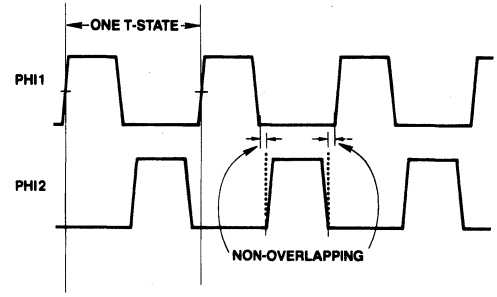
TL/EE/8583–10

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32132 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in *Figure 3-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See Sec. 4 for complete specifications of PHI1 and PHI2.



TL/EE/8583–11

FIGURE 3-2. Clock Timing Relationships

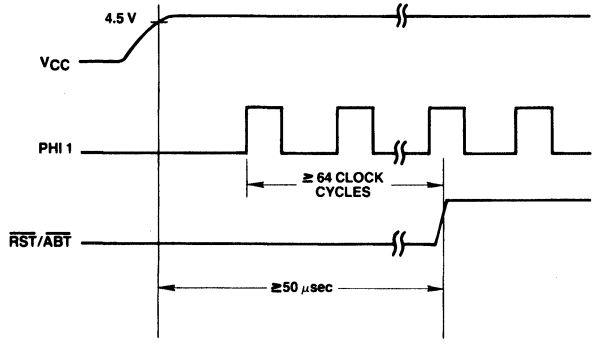
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 RESETTING

The $\overline{RST}/\overline{ABT}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.4.

The CPU may be reset at any time by pulling the $\overline{RST}/\overline{ABT}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, $\overline{RST}/\overline{ABT}$ must be held low for at least 50 μsec after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain



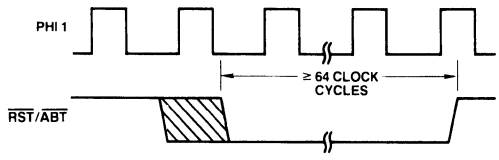
TL/EE/8583–12

FIGURE 3-3. Power-on Reset Requirements

3.0 Functional Description (Continued)

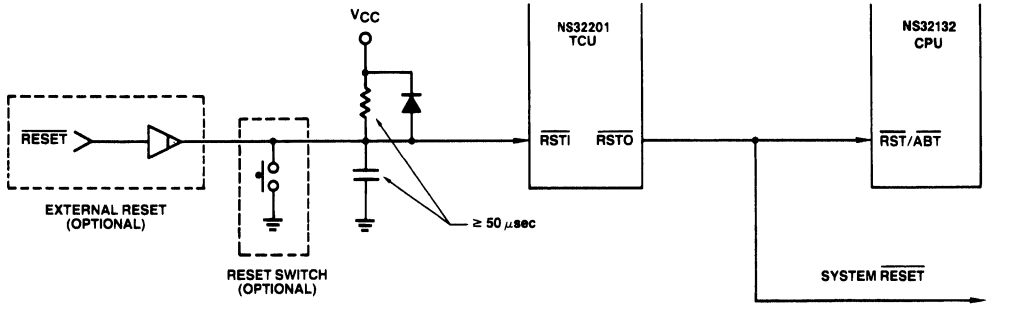
active for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4*.

The NS32201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32132 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



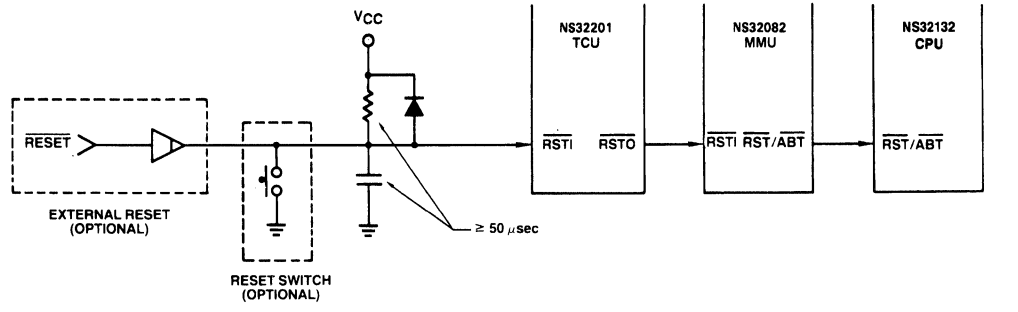
TL/EE/8583–13

FIGURE 3-4. General Reset Timing



TL/EE/8583–14

FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System



TL/EE/8583–15

FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

3.4 BUS CYCLES

The NS32132 CPU has a strap option which defines the Bus Timing Mode as either With or Without Address Translation. This section describes only bus cycles under the No Address Translation option. For details of the use of the strap and of bus cycles with address translation, see Sec. 3.5.

The CPU will perform a bus cycle for one of the following reasons:

1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
2) To fetch instructions into the eight-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4) To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Sec. 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Sec. 3.4.6).

The sequence of events in a non-Slave bus cycle is shown below in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).

3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

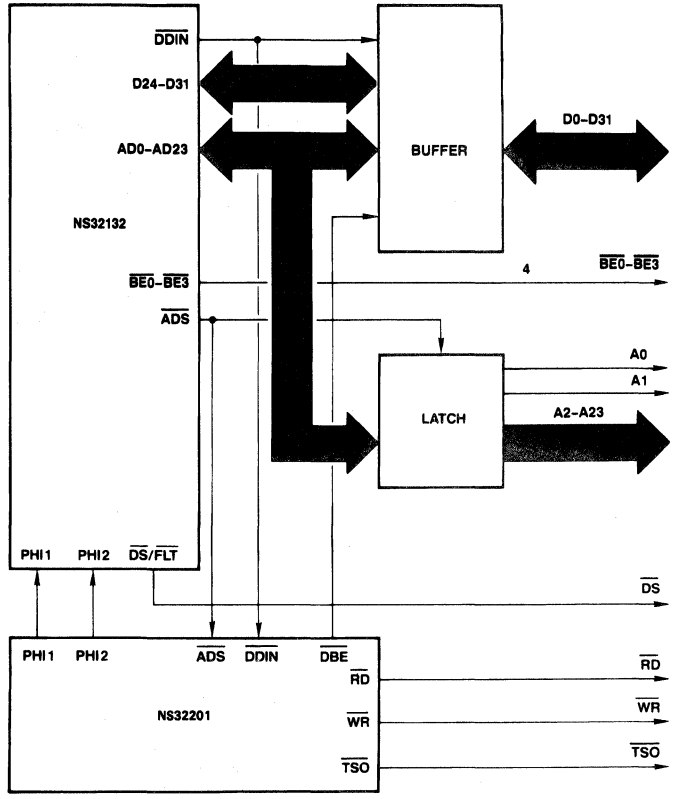
During T1, the CPU applies an address on pins AD0–AD23. It also provides a low-going pulse on the ADS pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–23 from the AD0–AD23 pins. See *Figure 3-6*. During this time also the status signals DDIN, indicating the direction of the transfer, and $\overline{BE0}$–$\overline{BE3}$, indicating which of the four bus bytes are to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD31 to either accept or present data. It also starts the data strobe ($\overline{DS}$), signalling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time: $\overline{RD}$ (Read Strobe) or $\overline{WR}$ (Write Strobe), $\overline{TSO}$ (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the end of T2, on the falling edge of the PHI2 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Sec. 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD31) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Sec. 4. Data must, however, be held at least until the beginning of T4. $\overline{DS}$ and $\overline{RD}$ are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

The T4 state finishes the bus cycle. At the beginning of T4, the $\overline{DS}$, $\overline{RD}$ or $\overline{WR}$, and $\overline{TSO}$ signals go inactive, and at the rising edge of PHI2, $\overline{DBE}$ goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).



TL/EE/8583–16

FIGURE 3-6. Bus Connections

3.0 Functional Description (Continued)

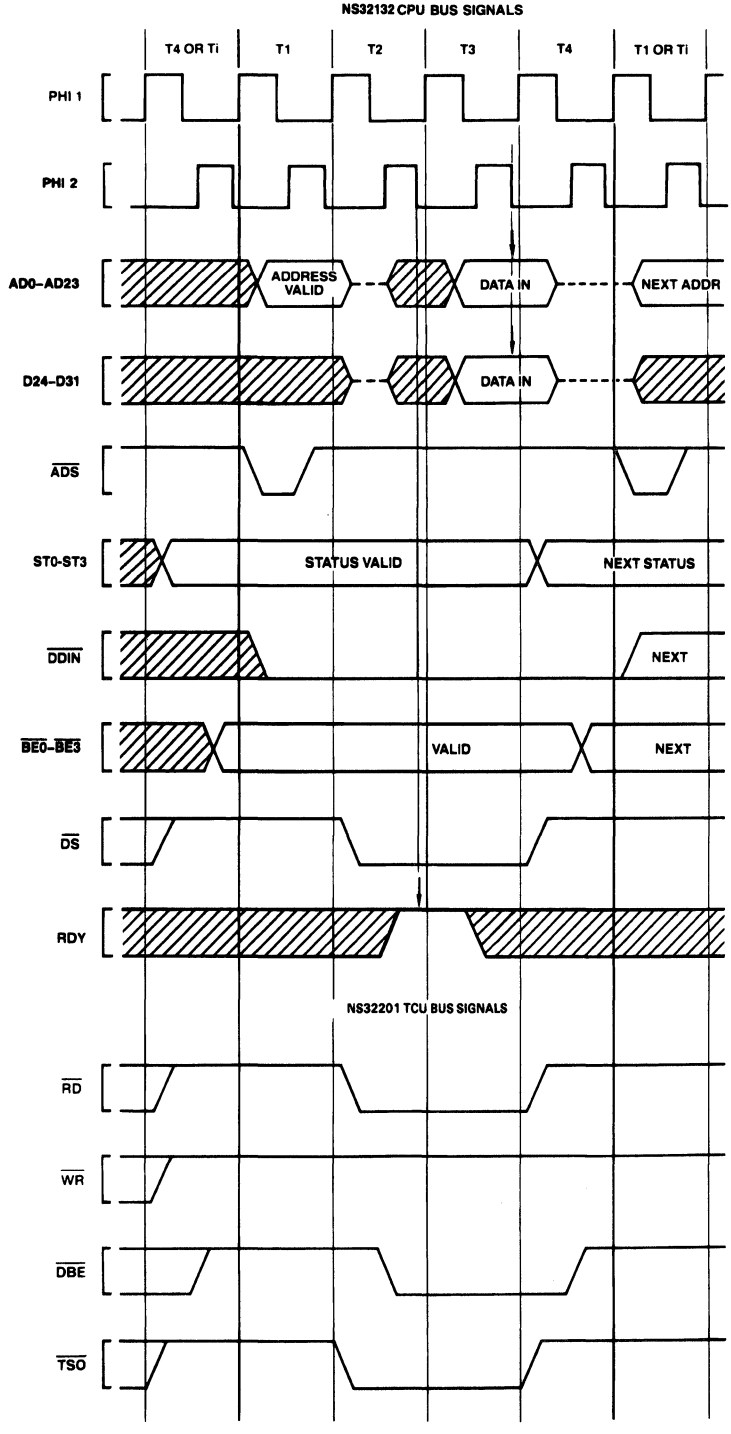
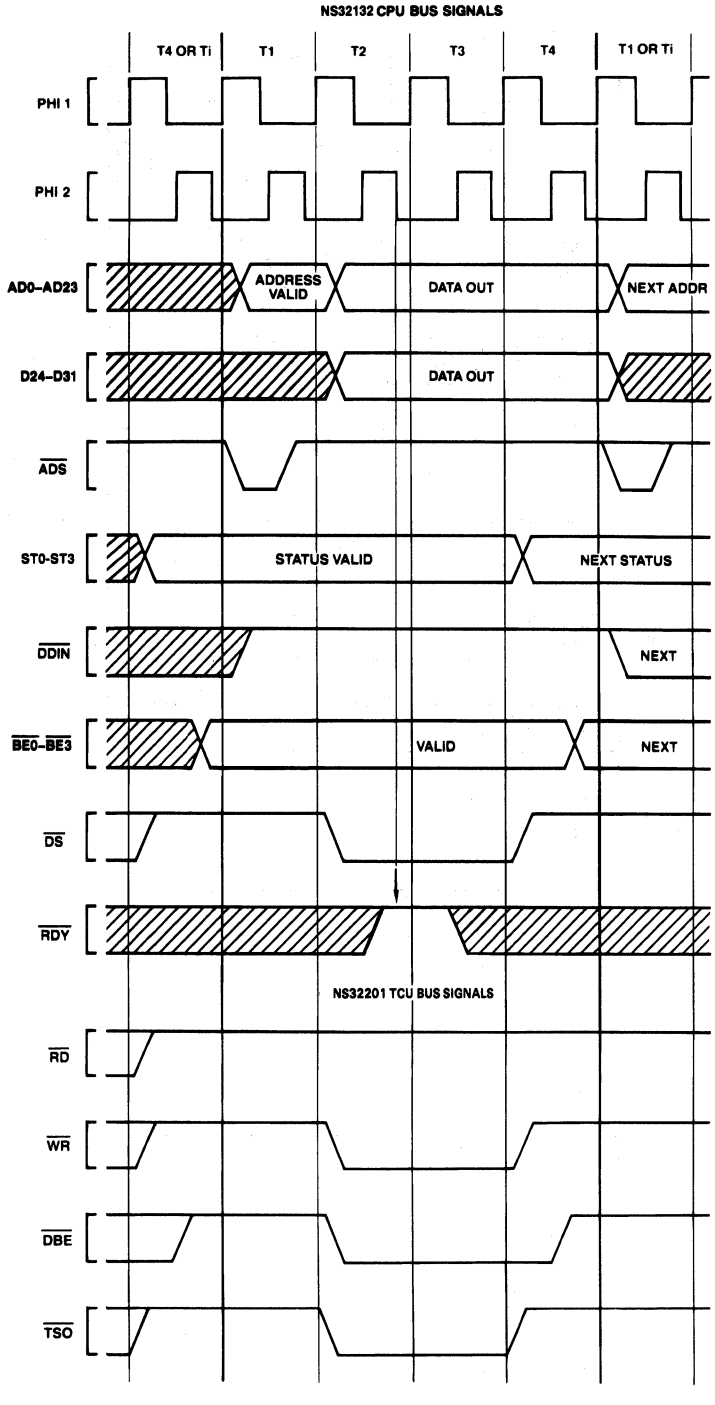


FIGURE 3-7. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/8583–18

FIGURE 3-8. Write Cycle Timing

3.0 Functional Description (Continued)

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32016 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

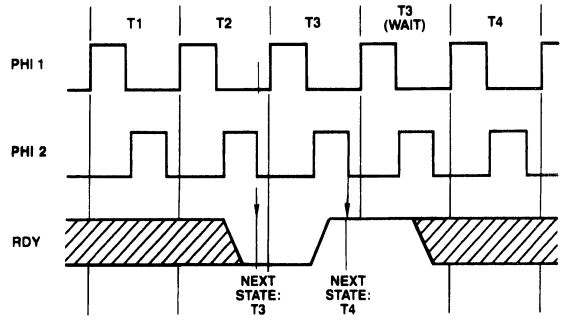
At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "WAIT STATE". See *Figure 3-9*.

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pin:

1) $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT states until removed.
2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
3) $\overline{\text{PER}}$ (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU $\overline{\text{WAITn}}$ pins.



TL/EE/8583-19

FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS32132 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why is it idle.

Referring to *Figures 3-7* and *3-8*, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before $\overline{\text{ADS}}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 – The bus is idle because the CPU does not need to perform a bus access.

0001 – The bus is idle because the CPU is executing the WAIT instruction.

0010 – (Reserved for future use.)

0011 – The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 – Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Sec. 3.4.5.

0110 – End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Sec. 3.4.5.

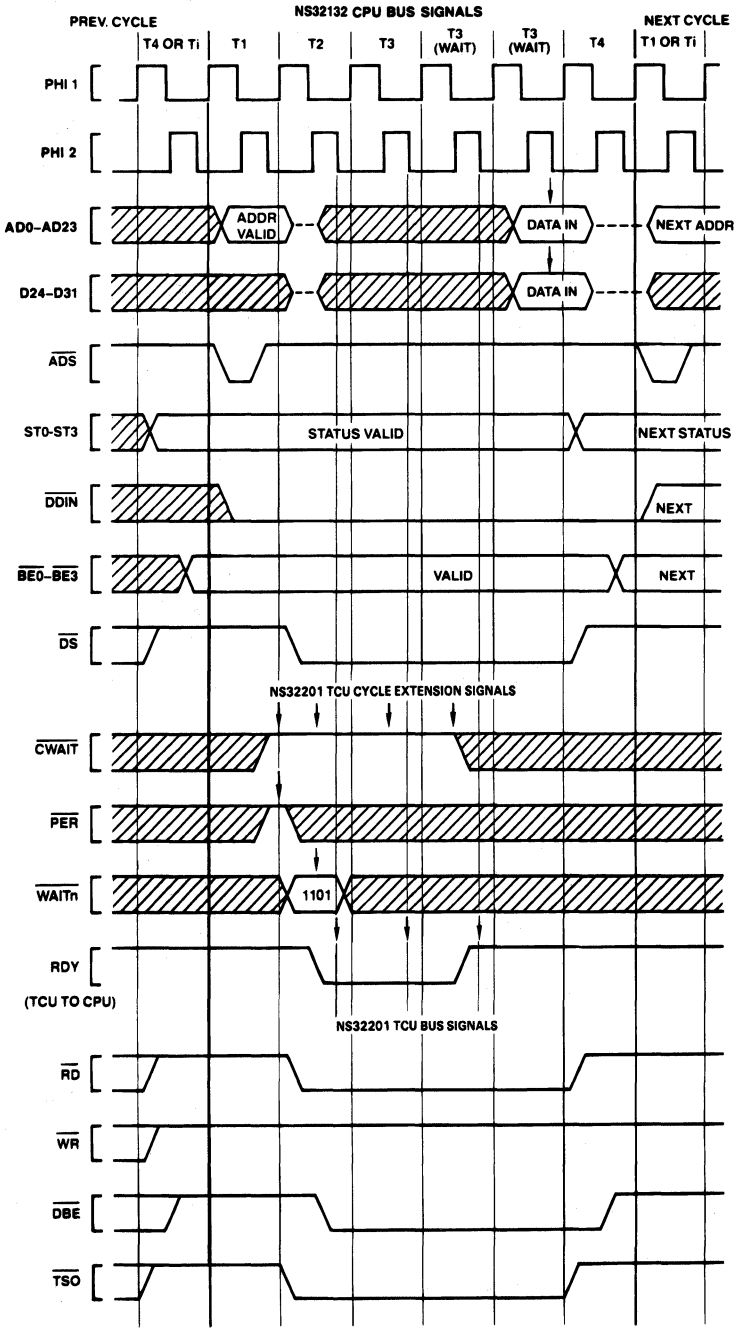
0111 – End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Sec. 3.4.5.

1000 – Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction

3.0 Functional Description (Continued)



TL/EE/8583–20

FIGURE 3-10. Extended Cycle Example

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAIT}}$n indicate points at which the TCU samples. Arrows on AD0–AD15 and RDY indicate points at which the CPU samples.

3.0 Functional Description (Continued)

Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.

The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

1110 – Read Slave Processor Status.

The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Sec. 3.9.1.

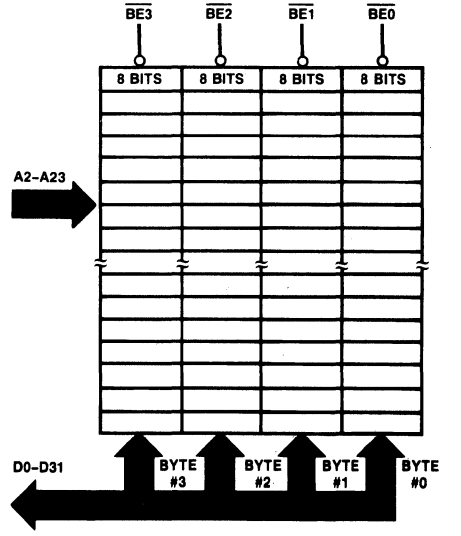
1111 – Broadcast Slave ID.

The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS32132 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32132 is that the presence of a 32-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32132 provides special control signals. Byte Enable ($\overline{BE0}$–$\overline{BE3}$) which facilitate individual byte accessing on a 32-bit bus.

Memory is organized as four eight-bit banks, each bank receiving the double-word address (A2–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7 is enabled when $\overline{BE0}$ is low. The second bank, connected to data bus pins AD8–AD15 is enabled when $\overline{BE1}$ is low. The third and fourth banks are enabled by $\overline{BE2}$ and $\overline{BE3}$, respectively. See *Figure 3-11*.



TL/EE/8583–21

FIGURE 3-11. Memory Interface

Since operands do not need to be aligned with respect to the double-word bus accessed performed by the CPU, a given double-word access can contain one, two, three, or four bytes of the operand being addressed, and these bytes can begin at various positions, as determined by A1, A0. Table 3-1 lists the 10 resulting access types.

TABLE 3-1

Bus Access Types

Type	Bytes Accessed	A1,A0	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$
1	1	00	1	1	1	0
2	1	01	1	1	0	1
3	1	10	1	0	1	1
4	1	11	0	1	1	1
5	2	00	1	1	0	0
6	2	01	1	0	0	1
7	2	10	0	0	1	1
8	3	00	1	0	0	0
9	3	01	0	0	0	1
10	4	00	0	0	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment. Table 3-2 lists the bus cycles performed for each situation.

3.0 Functional Description (Continued)

TABLE 3-2
Access Sequences

Cycle	Type	Address	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	Data Bus Byte 3	Byte 2	Byte 1	Byte 0
A. ***Word at address ending with 11***									BYTE 1 | BYTE 0	← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	1	A + 1	1	1	1	0	X	X	X	Byte 1
B. ***Double word at address ending with 01***							BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
C. ***Double word at address ending with 10***							BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
D. ***Double word at address ending with 11***							BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
E. ***Quad word at address ending with 00***							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	10	A	0	0	0	0	Byte 3	Byte 2	Byte 1	Byte 0
Other bus cycles (instruction prefetch or slave) can occur here.										
2.	10	A + 4	0	0	0	0	Byte 7	Byte 6	Byte 5	Byte 4
F. ***Quad word at address ending with 01***							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	9	A + 4	0	0	0	1	Byte 6	Byte 5	Byte 4	X
4.	1	A + 7	1	1	1	0	X	X	X	Byte 7
G. ***Quad word at address ending with 10***							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	7	A + 4	0	0	1	1	Byte 5	Byte 4	X	X
4.	5	A + 6	1	1	0	0	X	X	Byte 7	Byte 6
H. ***Quad word at address ending with 11***							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
Other bus cycles (instruction prefetch or slave) can occur here.										
1.	4	A + 4	0	1	1	1	Byte 4	X	X	X
2.	8	A + 5	1	0	0	0	X	Byte 7	Byte 6	Byte 5

X = Don't Care

3.0 Functional Description (Continued)

3.4.3.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.3.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

3.4.4 Instruction Fetches

Instructions for the NS32132 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the eight-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Sec. 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always type 10 Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle depends on the destination address.

Note: During non-sequential fetches $\overline{BE0}$–$\overline{BE3}$ are all active regardless of the alignment.

3.4.5 Interrupt Control Cycles

Activating the $\overline{INT}$ or $\overline{NMI}$ pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status pesented on pins ST0–ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32132 interrupt structure, see Sec. 3.8.

3.0 Functional Description (Continued)

TABLE 3-3
Interrupt Sequences

								Data Bus			
Cycle	**Status**	**Address**	**$\overline{DDIN}$**	**$\overline{BE3}$**	**$\overline{BE2}$**	**$\overline{BE1}$**	**$\overline{BE0}$**	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**
				A. Non-Maskable Interrupt Control Sequences							
Interrupt Acknowledge											
1	0100	$FFFF00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
None: Performed through Return from Trap (RETT) instruction.											
				B. Non-Vectored Interrupt Control Sequences							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	X
				C. Vectored Interrupt Sequences: Non-Cascaded.							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Range: 0-127
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Same as in Previous Int. Ack. Cycle
				D. Vectored Interrupt Sequences: Cascaded							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0101	Cascade Address	0			See Note		Vector, range 9–255; on appropriate byte of data bus.			
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: Same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address)											
2	0111	Cascade Address	0			See Note		X	X	X	X

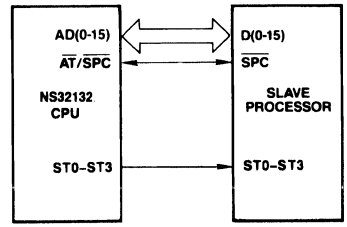
X = Don't Care

Note: $\overline{BE0}$-$\overline{BE3}$ signals will be activated according to the cascaded ICU address. The cycle type can be 1, 2, 3 or 4, when reading the interrupt vector. The vector value can be in the range 0–255.

3.0 Functional Description (Continued)

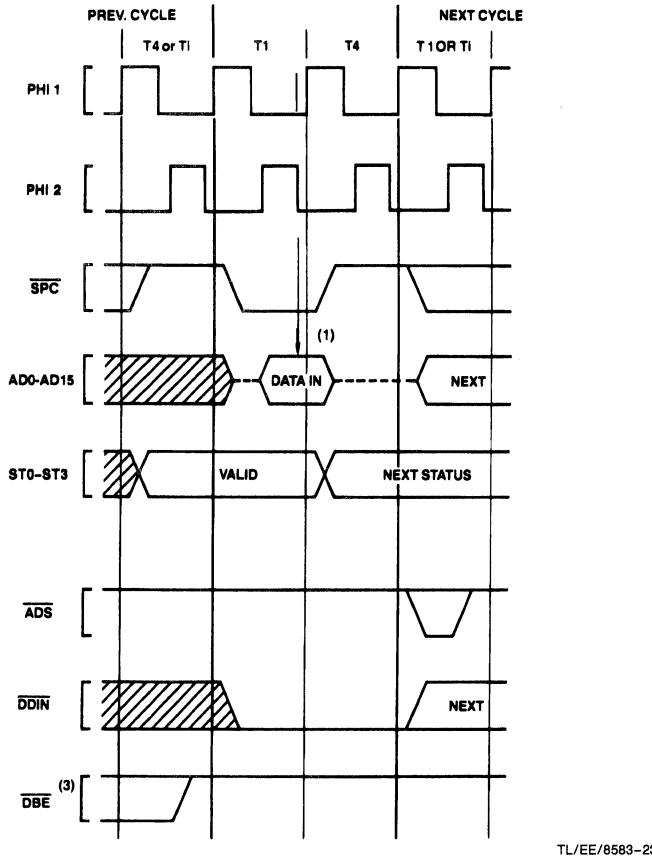
3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Sec. 3.5.1), the $\overline{AT}/\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{SPC}$). In a Slave Processor bus cycle, data is transferred on the Data Bus (AD0–AD15), and the status lines ST0–ST3 are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.



TL/EE/8583–22

FIGURE 3-12. Slave Processor Connections



TL/EE/8583–23

Note:

(1) CPU samples Data Bus here.

(2) $\overline{DBE}$ and all other NS32201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor

3.0 Functional Description (Continued)

3.4.6.1 Slave Processor Bus Cycles

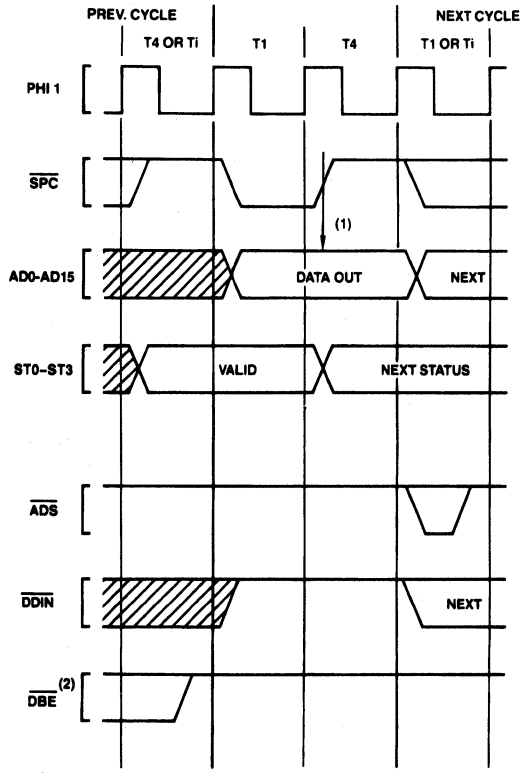
A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-13* and *3-14*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ($\overline{ADS}$), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.6.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0–AD7), and a Word operand is transferred on bits AD0–AD15. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.

Note that the NS32132 uses only the two least significant bytes of the data bus for slave cycles. This is to maintain compatibility with existing slave processors.



TL/EE/8583–24

Note:

(1) Slave Processor samples Data Bus here.

(2) $\overline{DBE}$, being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32132 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

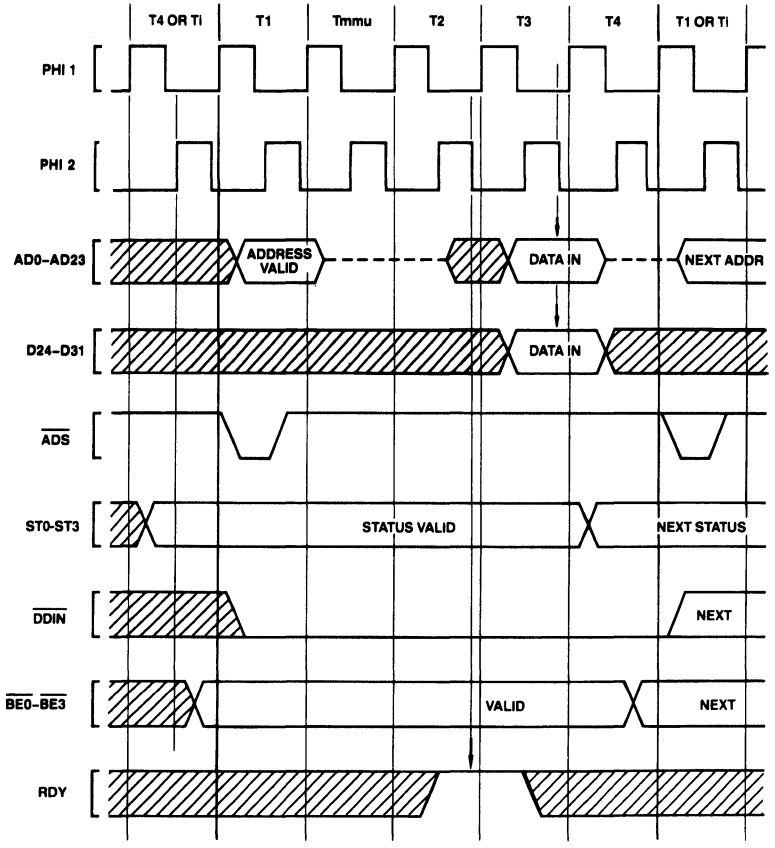
3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32132 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the $\overline{AT}/\overline{SPC}$ (Address Translation/Slave Processor Control) pin on the rising edge of the $\overline{RST}$ (Reset) pulse. If $\overline{AT}/\overline{SPC}$ is sampled as high, the bus timing is as previously described in Sec. 3.4. If it is sampled as low, two changes occur:

1) An extra clock cycle, Tmmu, is inserted into all bus cycles except Slave Processor transfers.
2) The $\overline{DS}/\overline{FLT}$ pin changes in function from a Data Strobe output ($\overline{DS}$) to a Float Command input ($\overline{FLT}$).

The NS32082 MMU will itself pull the CPU $\overline{AT}/\overline{SPC}$ pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the



TL/EE/8583-25

FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

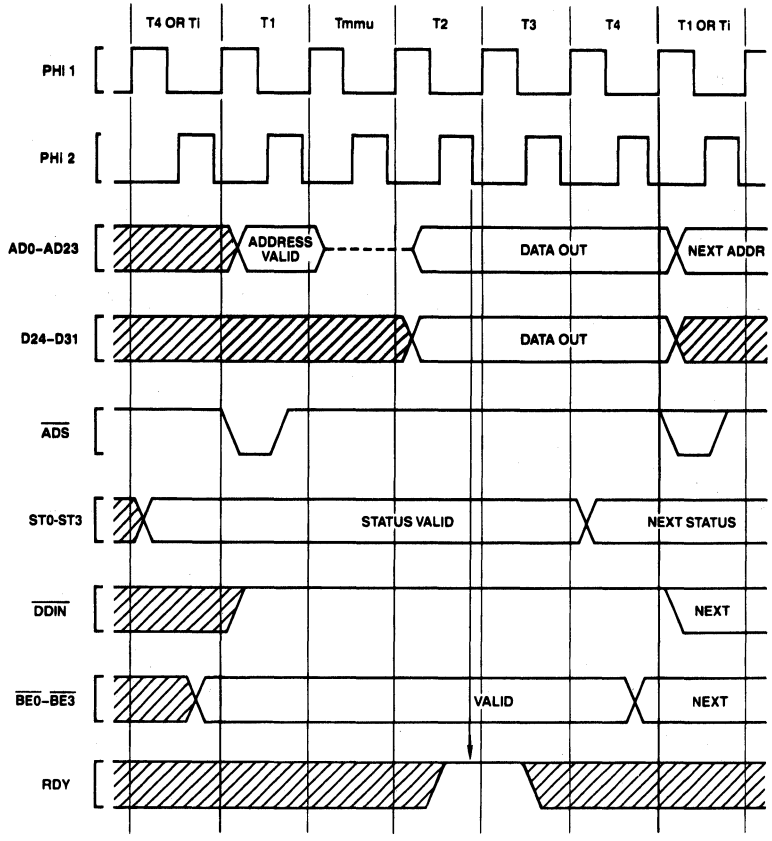
3.0 Functional Description (Continued)

presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Sec. 2.1.3.

3.5.2 Translated Bus Timing

Figures 3-15 and *3-16* illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0–AD23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{PAV}$. T2 through T4 of the cycle are identical to their counterparts without Address Translation. Note that in order for the NS32082 MMU to operate correctly it must be set to the 32132 mode by forcing A24/$\overline{HBF}$ low during reset. In this mode the bus lines AD16–AD23 are floated after the MMU address has been latched, since they are used by the CPU to transfer data.

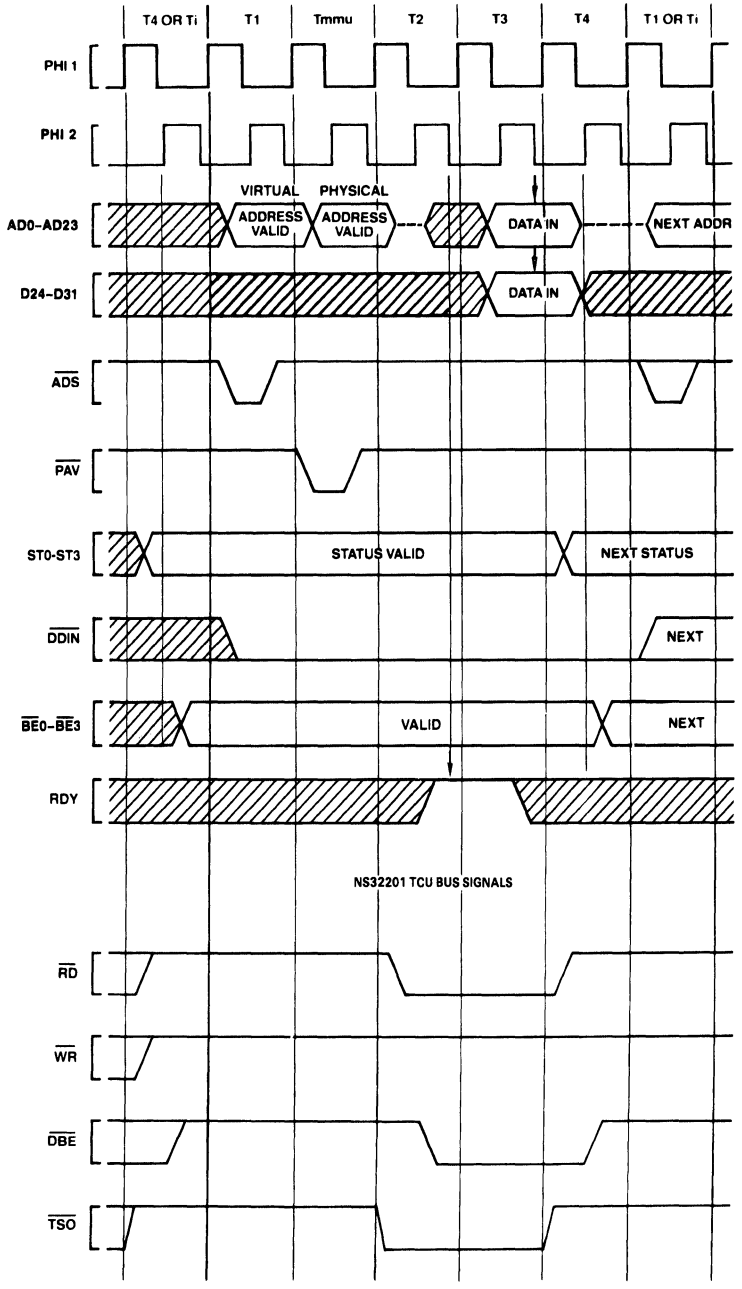
Figures 3-17 and *3-18* show a Read cycle and a Write cycle as generated by the 32132/32082/32201 group. Note that with the CPU $\overline{ADS}$ signal going only to the MMU, and with the MMU $\overline{PAV}$ signal substituting for $\overline{ADS}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.



TL/EE/8583–26

FIGURE 3-16. Write Cycle with Address Translation (CPU Action)

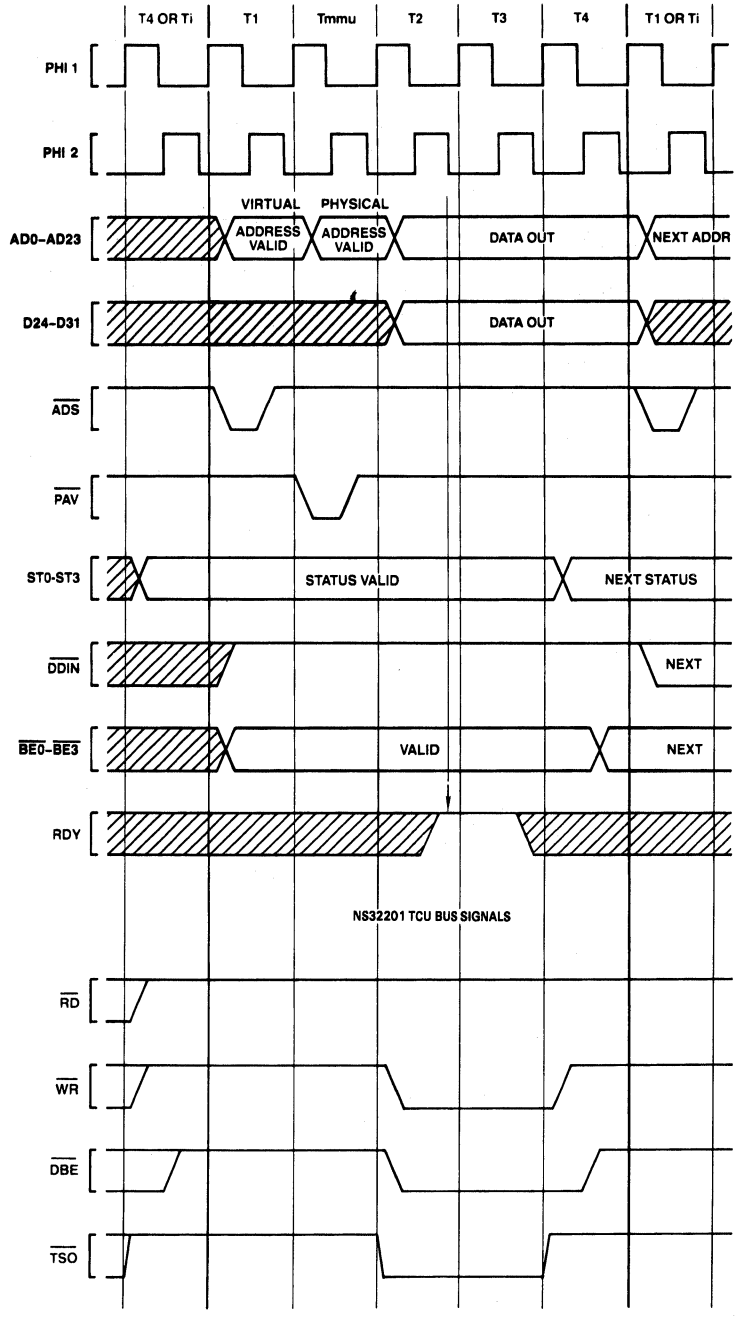
3.0 Functional Description (Continued)



TL/EE/8583-27

FIGURE 3-17. Memory-Managed Read Cycle

3.0 Functional Description (Continued)



TL/EE/8583-28

FIGURE 3-18. Memory-Managed Write Cycle

3.0 Functional Description (Continued)

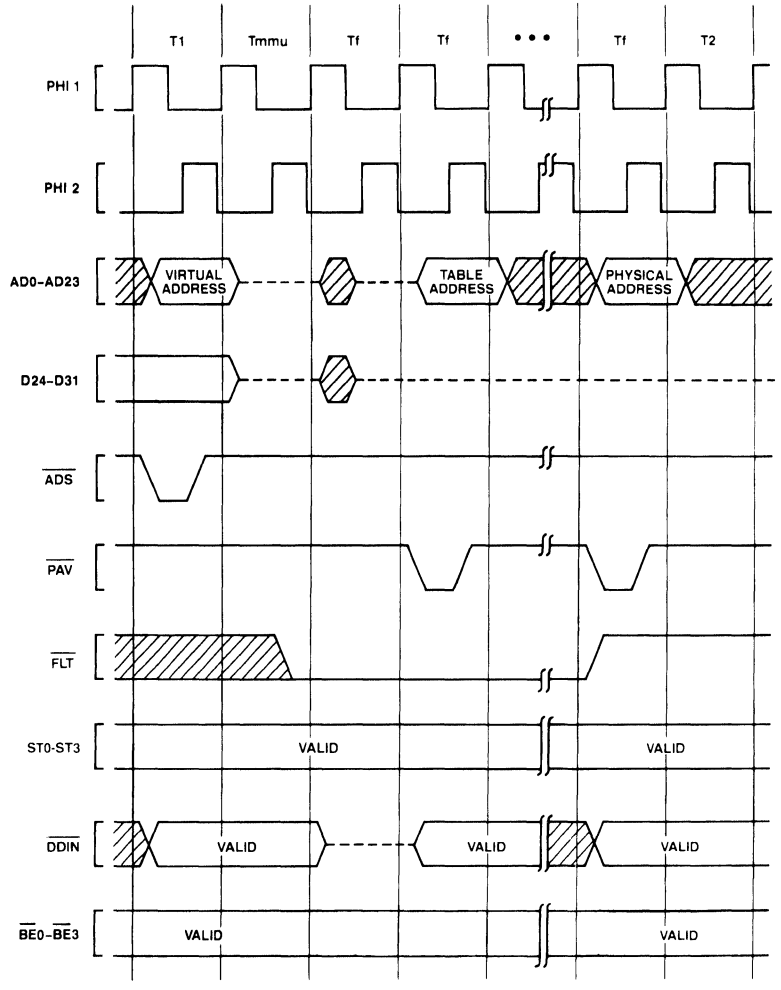
3.5.3 The FLT (Float) Pin

The $\overline{FLT}$ pin is used by the CPU for address translation support. Activating $\overline{FLT}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its internal translation look-aside buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effect of $\overline{FLT}$. Upon sampling $\overline{FLT}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD23, D24–D31 and $\overline{DDIN}$ to the TRI-STATE condition ("floating").
2) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{RST}/\overline{ABT}$ description, Sec. 3.5.4.)

Note that the AD0–AD23 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until $\overline{FLT}$ again goes high. See the Timing Specifications, Sec. 4.



TL/EE/8583-29

FIGURE 3-19. $\overline{FLT}$ Timing

3.0 Functional Description (Continued)

3.5.4 Aborting Bus Cycles

The $\overline{RST}/\overline{ABT}$ pin, apart from its Reset function (Sec. 3.3), also serves as the means to "abort", or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the $\overline{RST}/\overline{ABT}$ pin is held active for only one clock cycle.

If $\overline{RST}/\overline{ABT}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then Ti, thereby terminating the cycle. Since it is the MMU $\overline{PAV}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.
2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

3.5.4.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the NS32082 Memory Management Unit.

1) If $\overline{FLT}$ has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, *Figure 4-22*.
2) If $\overline{FLT}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{FLT}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{FLT}$ is removed. See *Figure 4-23*.
3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If $\overline{RST}/\overline{ABT}$ is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

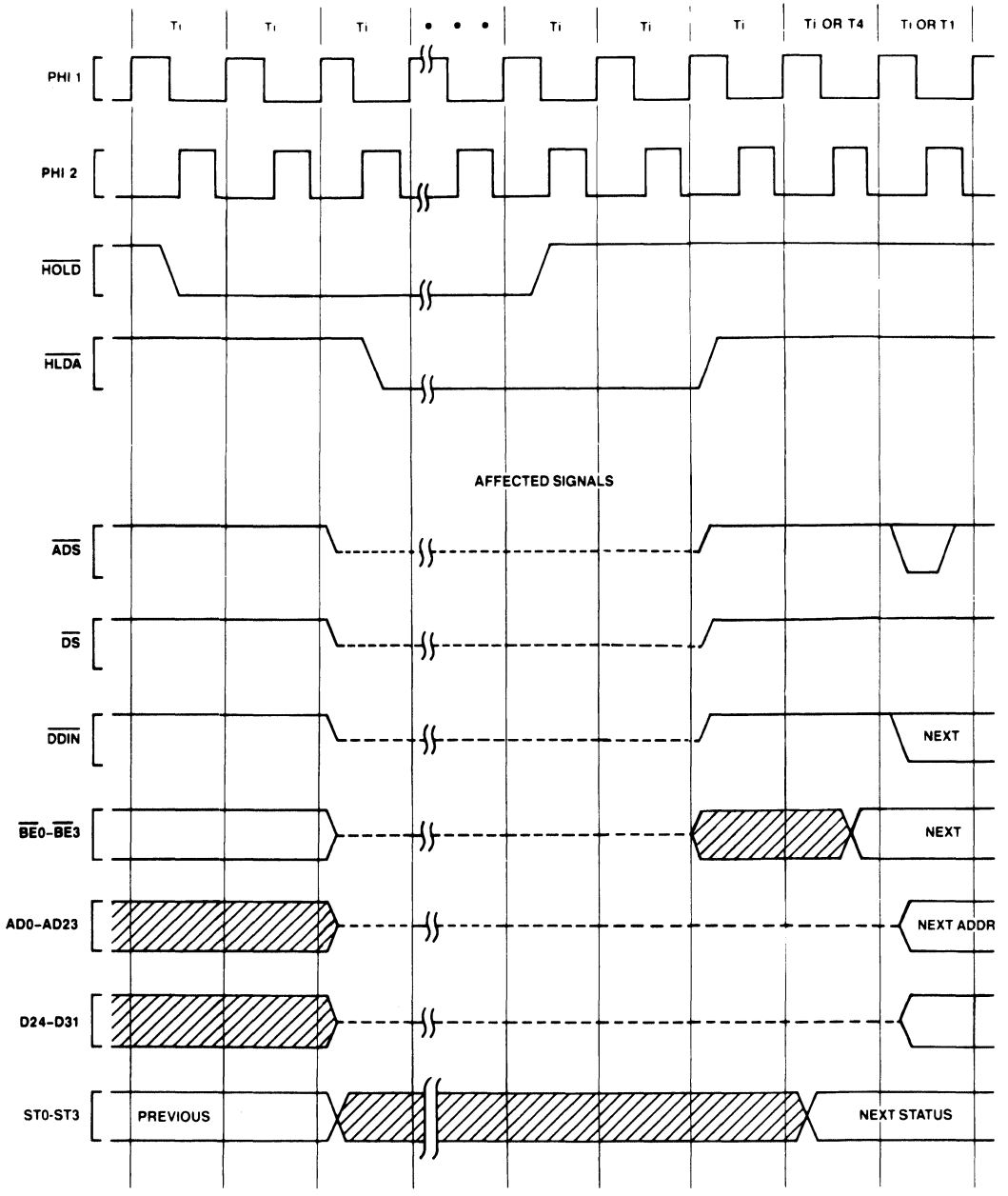
3.6 BUS ACCESS CONTROL

The NS32132 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{HOLD}$ (Hold Request) and $\overline{HLDA}$ (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of $\overline{HLDA}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD23, D24–D31, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{BE0}$–$\overline{BE3}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{HOLD}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{HLDA}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{HOLD}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-20* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-21* shows the sequence if the CPU is using the bus at the time that the $\overline{HOLD}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{HLDA}$ signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.

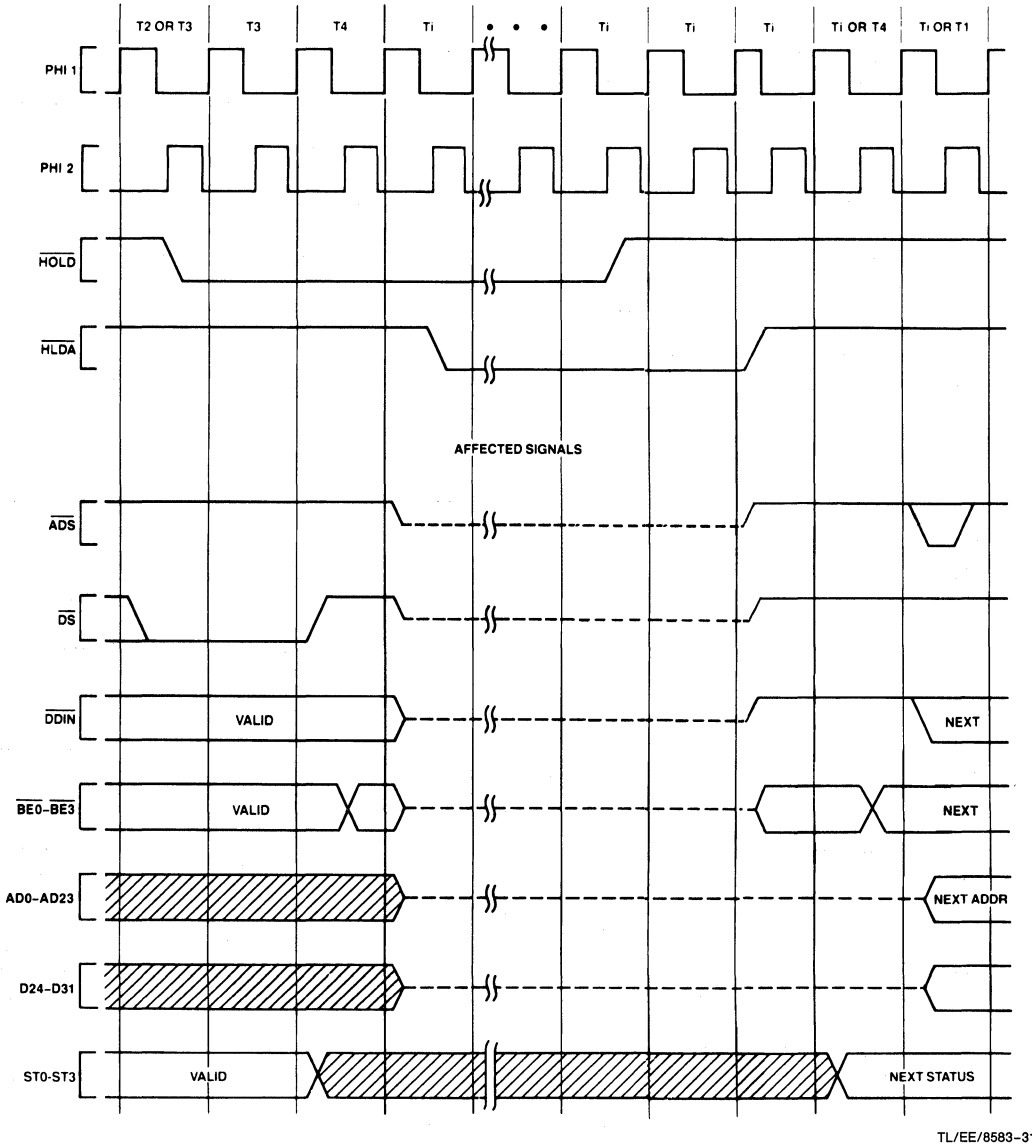
3.0 Functional Description (Continued)



TL/EE/8583-30

FIGURE 3-20. HOLD Timing, Bus Initially Idle

3.0 Functional Description (Continued)



TL/EE/8583-31

FIGURE 3-21. HOLD Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

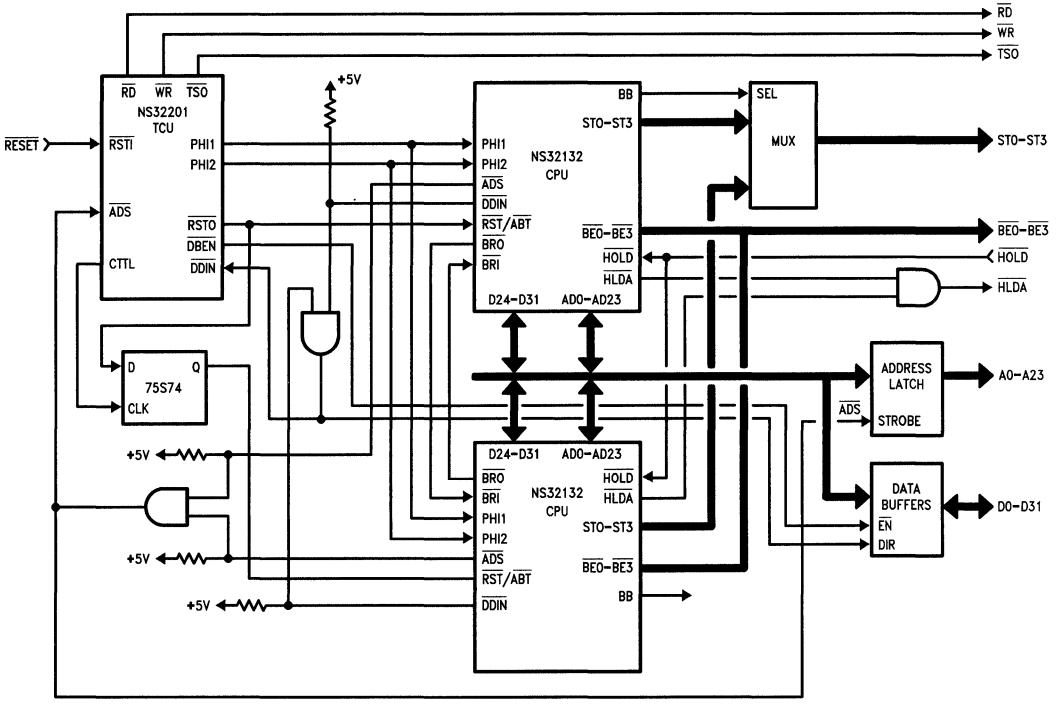
3.7 DUAL PROCESSING

An important feature of the NS32132 is its ability to cooperate with another NS32132, sharing the same bus and memory in a mode known as "Tightly-Coupled Dual Processing."

With the addition of one NS32132 and a minimal amount of standard TTL logic, this feature may enhance performance up to 80%. This is possible because the NS32132 average bus usage is in the range of 45% to 55%.

In a memory-managed system, an extra NS32201 TCU is required since a single TCU cannot drive two CPUs and two MMUs. In this case, a delay line may be needed to deskew the clock outputs from the two TCUs to avoid speed degradations.

Figure 3.22 shows a basic dual processing system configuration. A more complete connection diagram of a memory-managed system is given in Appendix B.



TL/EE/8583-32

FIGURE 3-22. Dual Processor System Basic Interconnections

3.0 Functional Description (Continued)

3.7.1 Bus Arbitration

The NS32132 provides two signals for the arbitration of the bus in a dual processing system:

- bus request out ($\overline{BRO}$)
- bus request in ($\overline{BRI}$)

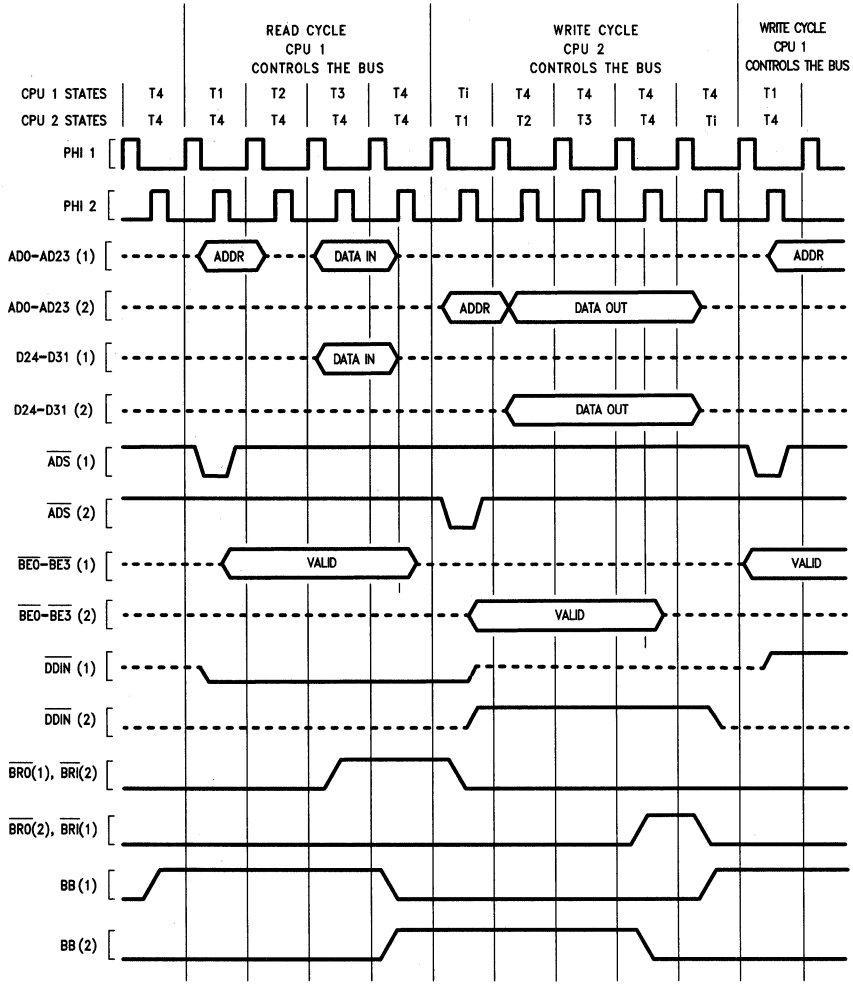
A third signal (BB), activated only when the CPU is in control of the bus, is provided for testing purposes and for controlling external logic.

At any given time only one CPU is in control of the bus, while the other has its address/data lines and other relevant signals floating. Whenever the CPU not in control of the bus needs to perform a bus cycle, it asserts $\overline{BRO}$ requesting the other CPU to release the bus. The CPU in control of the bus, detecting a low signal on its $\overline{BRI}$ pin, will keep the bus for no more than two bus cycles and release it. It then notifies the other CPU by deactivating its $\overline{BRO}$ signal, and bus control switches.

An exception to this rule occurs if the CPU in control of the bus is executing an interlocked instruction. In this case, the bus is released at the end of the instruction rather than at the end of the current bus cycle.

When a bus switch occurs, the address/data lines and the $\overline{BE0}$–$\overline{BE3}$ lines from the CPU in control of the bus are floated a half clock cycle before the other CPU starts driving these lines. This scheme ensures that no contentions occur during the switching when the lines from the two CPUs are hardwired.

If a bus switch follows a read cycle, the T4 cycle of the CPU in control of the BUS is immediately followed by the T1 cycle of the other CPU. If the bus switch follows a write cycle, the CPU in control of the bus adds an extra clock cycle and releases the bus during this cycle. This ensures that the data hold times for single and dual processor configurations are the same. A bus switching timing diagram is shown in *Figure 3-23.*



TL/EE/8583–33

FIGURE 3-23. Dual Processor Bus Arbitration Timing

3.0 Functional Description (Continued)

3.7.2 Processor Assignment

One problem with a multiprocessing system is ensuring that after a common initialization sequence, each processor branches to a different location in memory to execute a certain program. In cases where external hardware makes the system asymmetrical, the solution to this problem is simple. In a perfectly symmetrical system it is still possible to identify each processor uniquely so that tasks can be assigned to different processors. It is impossible, however, to know which processor will be assigned to a certain task.

The NS32132 relies on the fact that two tightly-coupled CPUs will switch bus control after at most two bus cycles, except when an interlocked instruction is executed. *Figure 3.24* shows an instruction sequence that can be used to assign processors to different tasks.

Both CPUs clear the same memory location on subsequent bus cycles. This ensures that both CPUs have cleared the memory location before any one of them executes the interlocked instruction. The CPU that exited the reset condition first executes the Set Bit Interlocked Instruction (SBITI), finding the bit clear. The other CPU will then execute the same instruction, but will find the bit set. Both CPUs will execute Branch on Flag Clear (BFC) instruction. The CPU that found the bit clear executes the branch; the other CPU continues to the next instruction. In this case the time constraint which delays one processor's exit from the reset condition makes it possible to know which processor will be assigned to a certain task.

```
SEM:    BLKB   1              ;Reserve storage
                               ;location for semaphore

        MOVQB  0, SEM         ;Clear semaphore
        NOP

        SBITI  0, SEM         ;Set bit interlocked

        BFC    TSK1           ;Branch on flag clear

        _____
        _____                 } CODE FOR TASK 2
        _____

TSK1:   _____
        _____                 } CODE FOR TASK 1
        _____
```

FIGURE 3-24. Instruction Sequence to Assign CPUs to Different Tasks

3.0 Functional Description (Continued)

3.8 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32132 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/$\overline{S}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection, and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-21*.

$\overline{ILO}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing. As with the U/$\overline{S}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, *Figure 4-19*.

3.9 NS32132 INTERRUPT STRUCTURE

$\overline{INT}$, on which maskable interrupts may be requested,

$\overline{NMI}$, on which non-maskable interrupts may be requested, and

$\overline{RST}$/$\overline{ABT}$, which may be used to abort a bus cycle and any associated instruction. It generates an interrupt request if an instruction was aborted. See Sec. 3.5.4.

In addition there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.9.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

1) Adjustment of Registers.

   Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2) Saving Processor Status.

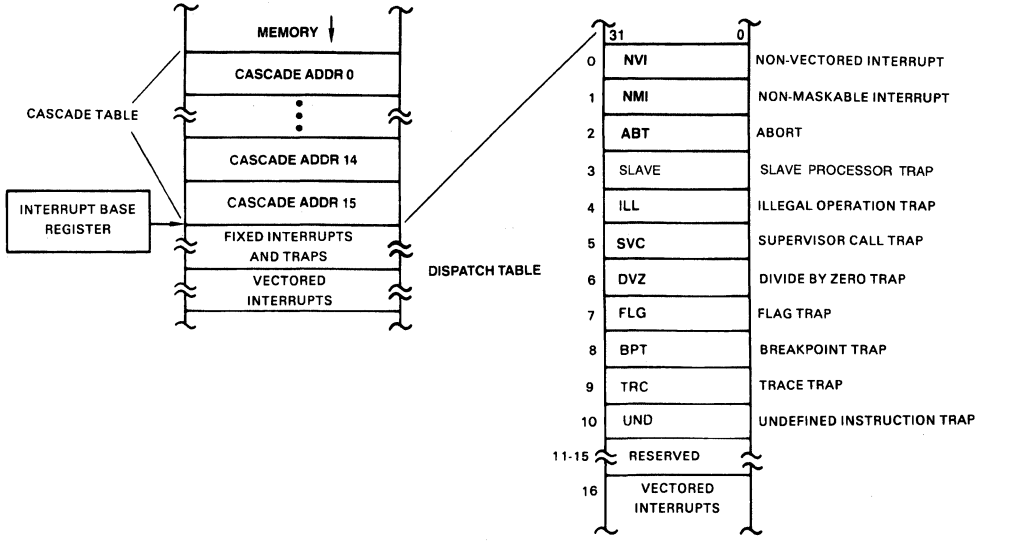
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

3) Vector Acquisition.

   A Vector is either obtained from the Data Bus or is supplied by default.

4) Service Call.

   The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-25*. A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.



TL/EE/8583–34

FIGURE 3-25. Interrupt Dispatch and Cascade Tables

3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-26*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Sec. 3.9.7.1.
Abort Interrupt:	Sec. 3.9.7.4.
Traps (except Trace):	Sec. 3.9.7.2.
Trace Trap:	Sec. 3.9.7.3.

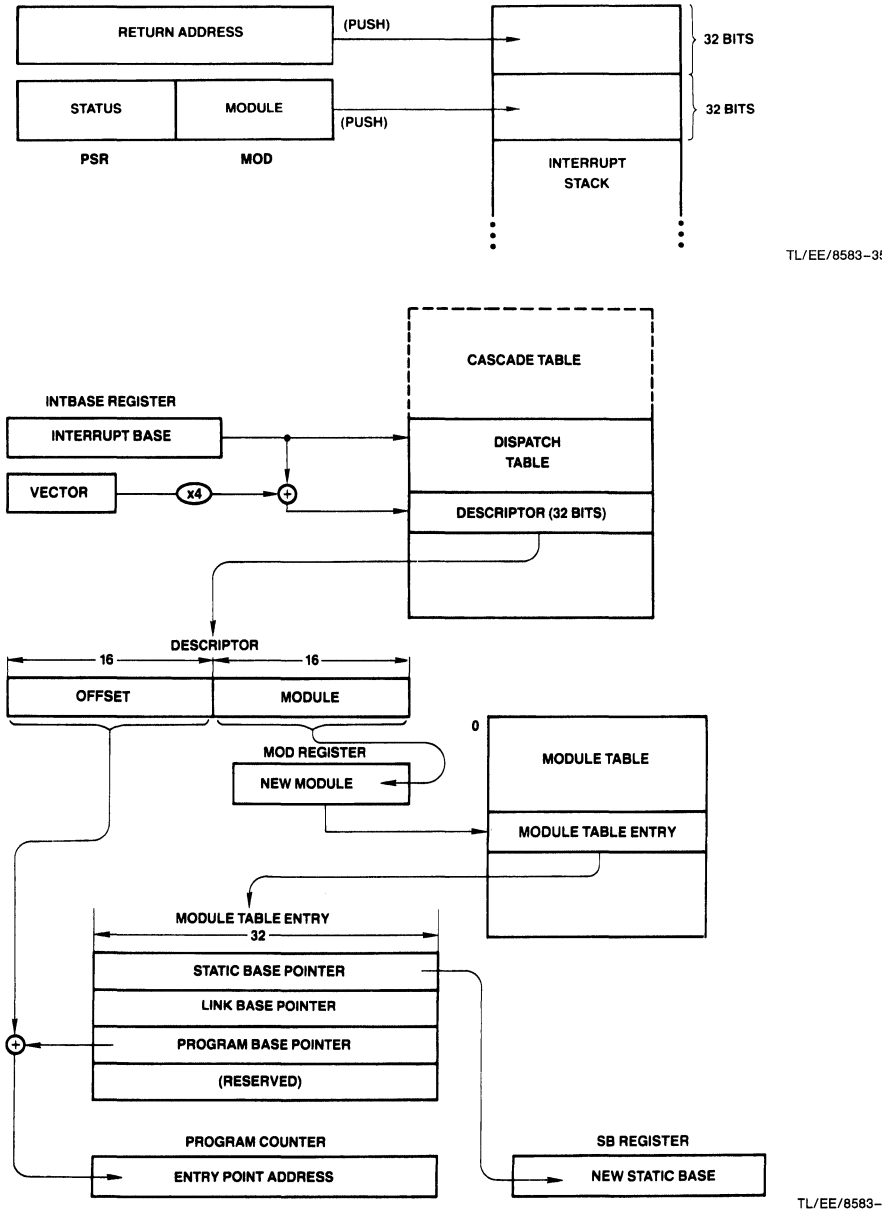


FIGURE 3-26. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.9.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction (*Figure 3-27*) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-28*.

3.9.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$, $\overline{\text{NMI}}$ or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

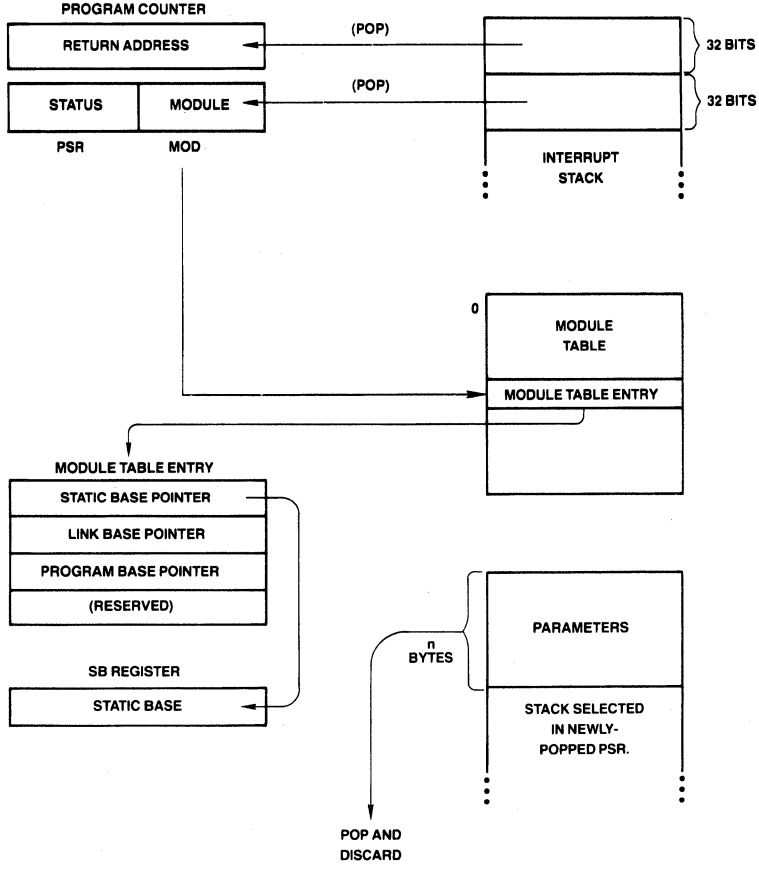
The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = C) or Vectored (bit I = 1).

3.9.3.1 Non-Vectored Mode

In the Non-Vectored mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.

3.9.3.2 Vectored Mode: Non-Cascaded Case

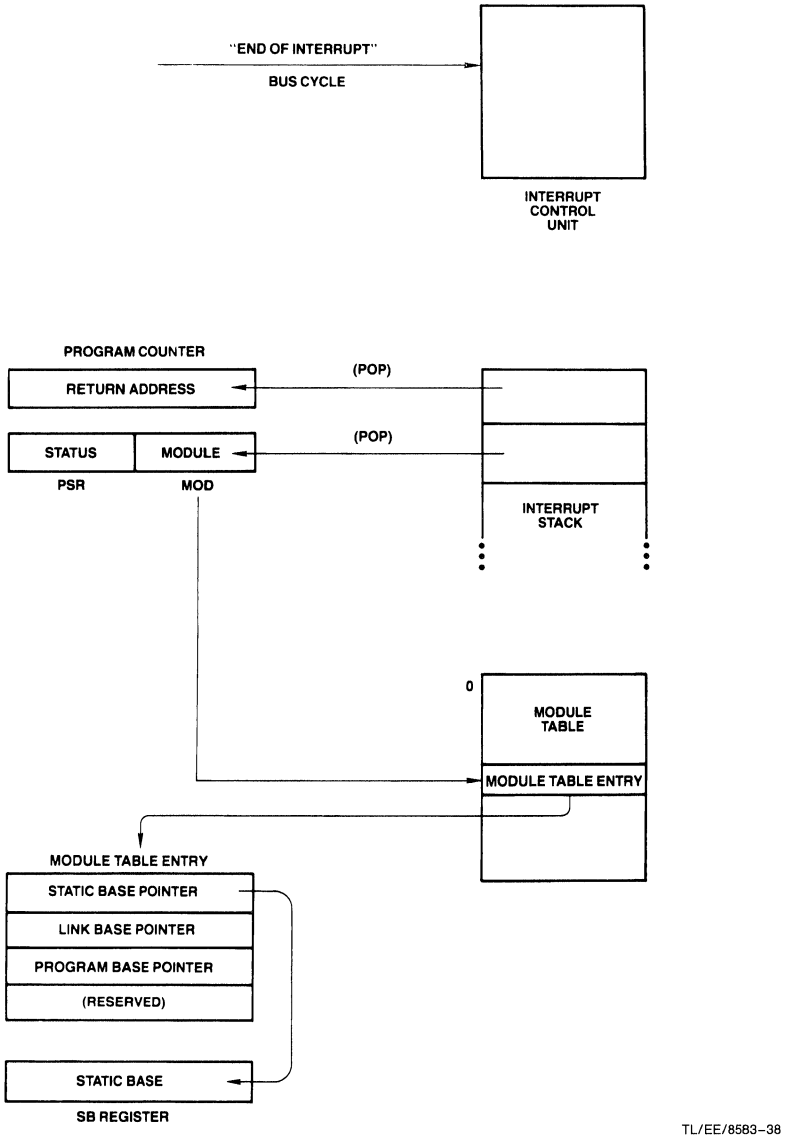
In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle



TL/EE/8583–37

FIGURE 3-27. Return from Trap (RETT n) Instruction Flow

3.0 Functional Description (Continued)



TL/EE/8583–38

FIGURE 3-28. Return from Interrupt (RETI) Instruction Flow

3.0 Functional Description (Continued)

(Sec. 3.4.2) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.9.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-30*, shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{INT}$ pin.

In a system which uses cascading, two tasks must be performed upon initialization:

1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.

2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

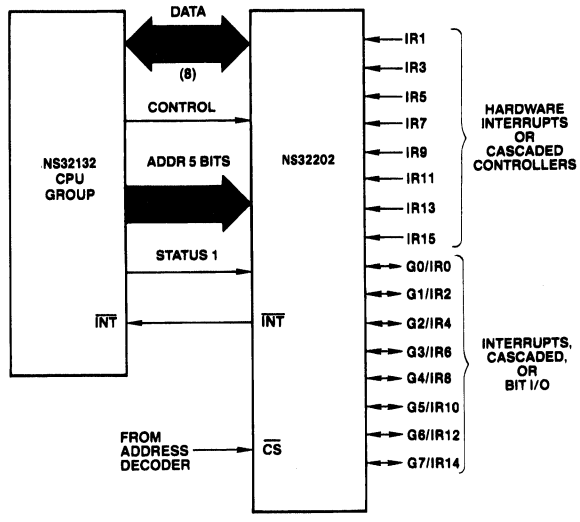
Figure 3-25 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range −16 to −1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Sec. 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Sec. 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

NOTE: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the interrupt mask register of the interrupt controller.

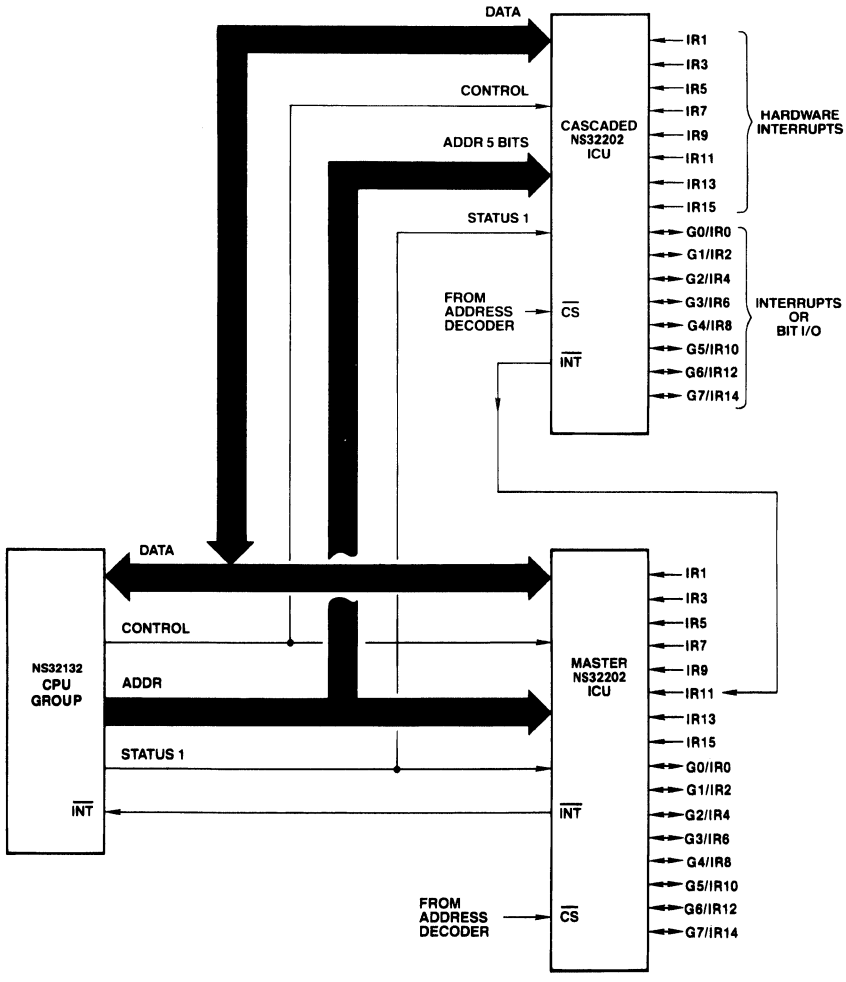
However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.



TL/EE/8583-39

FIGURE 3-29. Interrupt Control Unit Connections (16 Levels)

3.0 Functional Description (Continued)



TL/EE/8583–40

FIGURE 3-30. Cascaded Interrupt Control Unit Connections

3.9.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the $\overline{\text{NMI}}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is FFFF00_{16}. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Sec. 3.9.7.1.

3.9.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32132 CPU are:

Trap (SLAVE): An exceptional condition was detected by the Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.10.1).

3.0 Functional Description (Continued)

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

3.9.6 Prioritization

The NS32016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1) Traps other than Trace (Highest priority)
2) Abort
3) Non-Maskable Interrupt
4) Maskable Interrupts
5) Trace Trap (Lowest priority)

3.9.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-31*. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{INT}$ or $\overline{NMI}$ pins, respectively), see Sec. 3.9.7.1 For Abort Interrupts, see Sec. 3.9.7.4. For the Trace Trap, see Sec. 3.9.7.3, and for all other traps see Sec. 3.9.7.2.

3.9.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{NMI}$ pin receives a falling edge, or the $\overline{INT}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.
   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master, Sec. 3.4.2). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.2). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.2).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE $+4*$ Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Sec. 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-31*.

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector* 4 + INTBASE Register contents.

2) Move the Module field of the Descriptor into the MOD Register.

3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.

4) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.

5) Flush queue: Non-sequentially fetch first instruction of Interrupt routine.

6) Push MOD Register into the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)

7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-31. Service Sequence

Invoked during all interrupt/trap sequences.

3.0 Functional Description (Continued)

3.9.7.2 Trap Sequence: Traps Other Than Trace

1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2) Set "Vector" to the value corresponding to the trap type.
   SLAVE: Vector = 3.
   ILL: Vector = 4.
   SVC: Vector = 5.
   DVZ: Vector = 6.
   FLG: Vector = 7.
   BPT: Vector = 8.
   UND: Vector = 10.
3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Return Address" to the address of the first byte of the trapped instruction.
6) Perform Service (Vector, Return Address), *Figure 3-31.*

3.9.7.3 Trace Trap Sequence

1) In the Processor Status Register (PSR), clear the P bit.
2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4) Set "Vector" to 9.
5) Set "Return Address" to the address of the next instruction.
6) Perform Service (Vector, Return Address), *Figure 3-31.*

3.9.7.4 Abort Sequence

1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
2) Clear the PSR P bit.
3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Vector" to 2.
6) Set "Return Address" to the address of the first byte of the aborted instruction.
7) Perform Service (Vector, Return Address), *Figure 3-31.*

3.10 SLAVE PROCESSOR INSTRUCTIONS

The NS32132 CPU recognizes three groups of instructions being executable by external Slave Processor:

Floating Point Instruction Set

Memory Management Instruction Set

Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

3.10.1 Slave Processor Protocol

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.
2) It specifies which Slave Processor will execute it.
3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-32.* While applying Status Code 1111 (Broadcast ID, Sec. 3.4.2), the CPU transfers the ID Byte on the least-significant byte of the Data Bus (AD0–AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The operation Word is swapped on the Data Bus, that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Address Mode fields within the Operation Word, the CPU starts fetching operand and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible

Status Combinations:

Send ID (ID): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operaton Word.
3	OP	CPY Sends Required Operands
4	—	Slave Starts Execution. CPU Pre-fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-32. Slave Processor Protocol

3.0 Functional Description (Continued)

for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.2).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this, and for the Address Translation strap function, $\overline{AT}/\overline{SPC}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Sec. 3.4.2).

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.2). This word has the format shown in *Figure 3-33*. If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the SLAVE vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.10.2 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

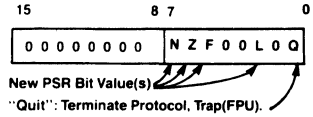
The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-33*).

TABLE 3-4
Floating Point Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note:
D = Double Word
i = Integer size (B,W,D) specified in mnemonic.
f = Floating Point type (F,L) specified in mnemonic.
N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)



TL/EE/8583–41

FIGURE 3-33. Slave Processor Status Word Format

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

3.10.3 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Instruction Set Reference Manual and the NS32082 MMU Data Sheet.

TABLE 3-5

Memory Management Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Instruction Set Reference Manual and the NS32082 Memory Management Unit Data Sheet.

D = Double Word

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)

3.10.4 Custom Slave Instructions

Provided in the NS32132 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

TABLE 3-6
Custom Slave Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op.2	none
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to OP. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

Note:

D = Double Word

i = Integer size (B,W,D) specified in mnemonic.

c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.

* = Privileged instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 NS32132 PIN DESCRIPTIONS

The following is a brief description of all NS32132 pins. The descriptions reference portions of the Functional Description. Sec. 3.

Unless otherwise indicated (see pin 34) reserved pins should be left open.

4.1.1 Supplies

Power (V_{CC}): +5V Positive Supply. Sec. 3.1.

Logic Ground (GNDL): Ground reference for on-chip logic. Sec. 3.1.

Buffer Grounds (GNDB1, GNDB2, GNDB3): Ground references for the on-chip output drivers.

Back-Bias Generator (BBG): Cutput of on-chip substrate voltage generator. Sec. 3.1.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Sec. 3.2.

Ready (RDY): Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Sec. 3.4.1.

Hold Request ($\overline{HOLD}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes.

Note 1. $\overline{HOLD}$ must not be asserted until $\overline{HLDA}$ from a previous $\overline{HOLD}/\overline{HLDA}$ sequence is deasserted.

Note 2. If the $\overline{HOLD}$ signal is generated asynchronously, its set up and hold times may be violated.

In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{HLDA}$ latency. This is to avoid speed degradations in cases of heavy $\overline{HOLD}$ activity (i.e. DMA controller cycles interleaved with CPU cycles.)

Interrupt ($\overline{INT}$): Active low. Maskable Interrupt request. Sec. 3.9.

Non-Maskable Interrupt ($\overline{NMI}$): Active low. Non-Maskable Interrupt request. Sec. 3.9.

Reset/Abort ($\overline{RST}/\overline{ABT}$): Active low. If held active for one clock cycle and released, this pin causes an instruction Abort, Sec. 3.5.4. If held longer, it initiates a Reset. Sec. 3.3.

Bus Request IN ($\overline{BRI}$): Active Low. Used in Dual-Processing systems to signal one NS32132 that the other NS32132 in the system is requesting the bus. Sec. 3.7.

This pin should be connected to V_{CC} through a 4.7 kΩ resistor if dual processing is not used.

4.1.3 Output Signals

Address Strobe ($\overline{ADS}$): Active low. Controls address latches: indicates start of a bus cycle. Sec. 3.4.

Data Direction in ($\overline{DDIN}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Sec. 3.4.

Byte Enable ($\overline{BE0}$–$\overline{BE3}$): Active low. Four control signals enabling data transfers on individual bus bytes. Sec. 3.4.3.

Status (ST0–ST3): Bus cycle status code, ST0 least significant. Sec. 3.4.2. Encodings are:

0000 — Idle: CPU Inactive on Bus.
0001 — Idle: WAIT Instruction.
0010 — (Reserved).
0011 — Idle: Waiting for Slave.
0100 — Interrupt Acknowledge, Master.
0101 — Interrupt Acknowledge, Cascaded.
0110 — End of Interrupt, Master.
0111 — End of Interrupt, Cascaded.
1000 — Sequential Instruction Fetch.
1001 — Non-Sequential Instruction Fetch.
1010 — Data Transfer.
1011 — Read Read-Modify-Write Operand.
1100 — Read for Effective Address.
1101 — Transfer Slave Operand.
1110 — Read Slave Status Word.
1111 — Broadcast Slave ID.

Hold Acknowledge ($\overline{HLDA}$): Active low. Applied by the CPU in response to $\overline{HOLD}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Sec. 3.6.

User/Supervisor (U/$\overline{S}$): User or Supervisor Mode status. Sec. 3.7. High state indicates User Mode, low indicates Supervisor Mode. Sec. 3.8.

Interlocked Operation ($\overline{ILO}$): Active low. Indicates that an interlocked instruction is being executed. Sec. 3.8.

Bus Request OUT ($\overline{BRO}$): Active Low. Used in Dual-Processing Systems by the NS32132 to signal the other CPU in the system that the bus is needed. Sec. 3.7.

Bus Busy (BB): This signal is used in Dual-Processing Systems and is activated by the CPU in control of the bus. When BB is inactive the Address-data-bus, $\overline{BE0}$–$\overline{BE3}$ and $\overline{DDIN}$ are floated. Sec. 3.7.

Program Flow Status ($\overline{PFS}$): Active low. Pulse indicates beginning of an instruction execution. Sec. 3.8.

4.1.4 Input/Output Signals

Address/Data 0–23 (AD0–AD23): Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Sec. 3.4.

Data Bits 24–31 (D24–D31): The high order 8 bits of the data bus.

Address Translation/Slave Processor Control ($\overline{AT}$/$\overline{SPC}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of a SLAVE instruction. Sec. 3.4.6; Sec. 3.9. Sampled on the rising edge of Reset as Address Translation Strap. Sec. 3.5.1.

In Non-Memory-Managed Systems this PIN should be pulled-up to V_{CC} through a 10 kΩ resistor.

Data Strobe/Float ($\overline{DS}/\overline{FLT}$): Active low. Data Strobe output, Sec. 3.4, or Float Command input, Sec. 3.5.3. Pin function is selected on $\overline{AT}/\overline{SPC}$ pin, Sec. 3.5.1.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages With Respect to GND	−0.5V to +7V
Power Dissipation	1.5 Watt

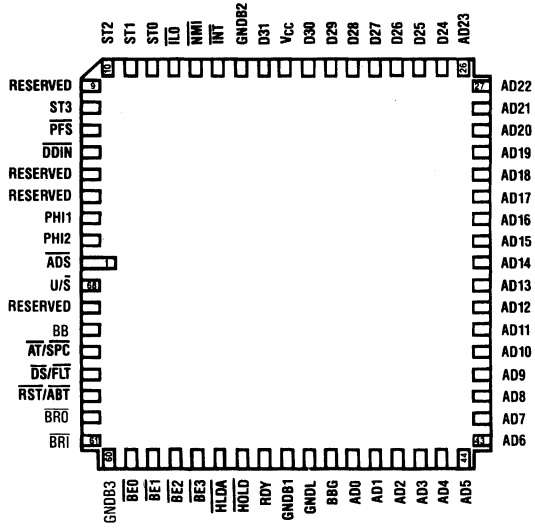
Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS $T_A = 0°$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC}+0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC}-0.35$		$V_{CC}+0.5$	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage. Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\ \mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2$ mA			0.45	V
I_{ILS}	$\overline{AT}/\overline{SPC}$ Input Current (low)	$V_{IN} = 0.4V$, $\overline{AT}/\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current (Output Pins in TRI-STATE Condition)	$0.4 \le V_{OUT} \le V_{CC}$	−20		30	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		200	300	mA

Connection Diagram

68 Pin LCC Package



TL/EE/8583-2

FIGURE 4-1

Bottom View

Order Number NS32132E-6, NS32132E-8, NS32132E-10, NS32132V-6, NS32132V-8 or NS32132V-10
See NS Package E68B or V68A

4.0 Device Specifications (Continued)

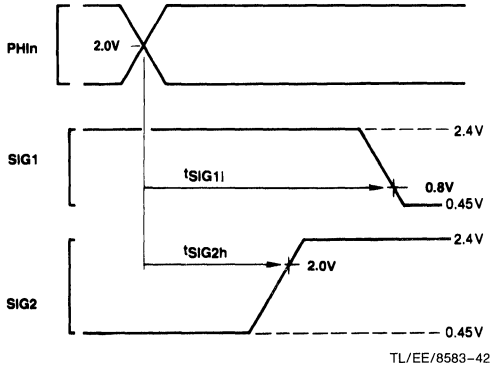
4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on rising or falling edges of the clock phases PHI1 and PHI2, and 0.8V or 2.0V on all other signals as illustrated in *Figures 4-2* and *4-3*, unless specifically stated otherwise.

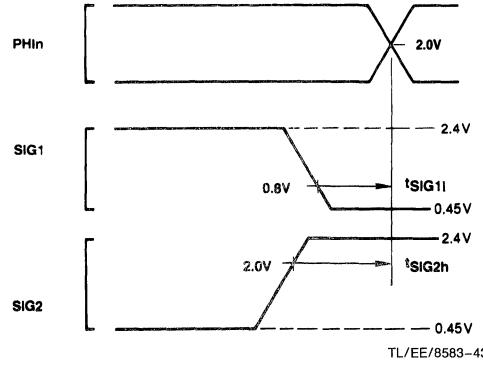
ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge
T.E. — trailing edge F.E. — falling edge



TL/EE/8583–42

FIGURE 4-2. Timing Specification Standard (Signal Valid After Clock Edge)



TL/EE/8583–43

FIGURE 4-3. Timing Specification Standard (Signal Valid Before Clock Edge)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays NS32132-6, NS32132-8, NS32132-10

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/Conditions	NS32132-6		NS32132-8		NS32132-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4-4	Address bits 0–23 valid	after R.E., PHI1 T1		80		65		50	ns
t_{ALh}	4-4	Address bits 0–23 hold	after R.E., PHI1 Tmmu or T2	5		5		5		ns
t_{Dv}	4-4	Data valid (write cycle)	after R.E., PHI1 T2		80		65		50	ns
t_{Dh}	4-4	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	4-4	Address bits 0–23 set up	before $\overline{ADS}$ T. E.	25		25		25		ns
t_{ALADSh}	4-10	Address bits 0–23 hold	after $\overline{ADS}$ T. E.	15		15		15		ns
t_{ALf}	4-5	Address bits 0–23 floating (no MMU)	after R.E., PHI1 T2		25		25		25	ns
t_{ADf}	4-5	Data bits D24–D31 floating (no MMU)	after R.E., PHI1 T2		25		25		25	ns
t_{ALMf}	4-9	Address bits 0–23 floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
t_{ADMf}	4-9	Data bits 21–31 floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
t_{BEv}	4-4	$\overline{BEn}$ signals valid	after R.E., PHI2 T4		95		70		45	ns
t_{BEh}	4-4	$\overline{BEn}$ signals hold	after R.E., PHI2 T4 or Ti	0		0		0		ns
t_{STv}	4-4	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		90		70		45	ns
t_{STh}	4-4	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	4-4	$\overline{DDIN}$ signal valid	after R.E., PHI1 T1		110		90		65	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays NS32132-6, NS32132-8, NS32132-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32132-6		NS32132-8		NS32132-10		Units
				Min	Max	Min	Max	Min	Max	
t_{DDINh}	4-4	$\overline{DDIN}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	4-4	$\overline{ADS}$ signal active (low)	after R.E., PHI1 T1		55		45		35	ns
t_{ADSia}	4-4	$\overline{ADS}$ signal inactive	after R.E., PHI2 T1		60		55		45	ns
t_{ADSw}	4-4	$\overline{ADS}$ pulse width	at 0.8V (both edges)	50		40		30		ns
t_{DSa}	4-4	$\overline{DS}$ signal active (low)	after R.E., PHI1 T2		70		60		45	ns
t_{DSia}	4-4	$\overline{DS}$ signal inactive	after R.E., PHI1 T4		50		50		40	ns
t_{ALf}	4-6	AD0–AD23 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{ADf}	4-6	D24–D31 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{DSf}	4-6	$\overline{DS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		80		55	ns
t_{ADSf}	4-6	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{BEf}	4-6	$\overline{BEn}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-6	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HLDAa}	4-6	$\overline{HLDA}$ signal active (low)	after R.E., PHI1 Ti		100		90		75	ns
t_{HLDAia}	4-8	$\overline{HLDA}$ signal inactive	after R.E., PHI1 Ti		100		90		75	ns
t_{DSr}	4-8	$\overline{DS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		80		55	ns
t_{ADSr}	4-8	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{BEr}	4-8	$\overline{BEn}$ signals return from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINr}	4-8	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-9	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		80		65		50	ns
t_{DDINr}	4-10	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		75		65		50	ns
t_{SPCa}	4-13	$\overline{SPC}$ output active (low)	after R.E., PHI1 T1		50		45		35	ns
t_{SPCia}	4-13	$\overline{SPC}$ output inactive	after R.E., PHI1 T4		50		45		35	ns
t_{SPCnf}	4-15	$\overline{SPC}$ output nonforcing	after R.E., PHI2 T4		40		25		10	ns
t_{Dv}	4-13	Data valid (slave processor write)	after R.E., PHI1 T1		80		65		50	ns
t_{Dh}	4-13	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	4-18	$\overline{PFS}$ pulse width	at 0.8V (both edges)	70		70		70		ns
t_{PFSa}	4-18	$\overline{PFS}$ pulse active (low)	after R.E., PHI2		70		60		50	ns
t_{PFSia}	4-18	$\overline{PFS}$ pulse inactive	after R.E., PHI2		70		60		50	ns
t_{ILOs}	4-20a	$\overline{ILO}$ signal setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32132-6, NS32132-8, NS32132-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32132-6		NS32132-8		NS32132-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ILOh}	4-20b	$\overline{ILO}$ signal hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t_{ILOa}	4-21	$\overline{ILO}$ signal active (low)	after R.E., PHI1		70		65		55	ns
t_{ILOia}	4-21	$\overline{ILO}$ signal inactive	after R.E., PHI1		70		65		55	ns
t_{USv}	4-22	U/$\overline{S}$ signal valid	after R.E., PHI1 T4		70		60		45	ns
t_{USh}	4-22	U/$\overline{S}$ signal hold	after R.E., PHI1 T4	10		10		10		ns
t_{NSPF}	4-19b	Nonsequential fetch to next $\overline{PFS}$ clock cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	4-19a	$\overline{PFS}$ clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	4-29	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t_{Cp}
t_{BBa}	4-30	BB signal active	after R.E., PHI1 T4		90		75		60	ns
t_{BBia}	4-20	BB signal inactive	after R.E., PHI1 T4		90		75		60	ns
t_{BROa}	4-20	$\overline{BRO}$ signal active	after R.E., PHI2		90		75		60	ns
t_{BROia}	4-20	$\overline{BRO}$ signal inactive	after R.E., PHI2		90		75		60	ns
t_{BEf}	4-30	$\overline{BE}$n signals floating (caused by $\overline{BRI}$)	after R.E., PHI2 T4		100		80		55	ns
t_{BEr}	4-30	$\overline{BE}$n signals return from floating (caused by $\overline{BRI}$)	after R.E., PHI1 T1		100		80		55	ns
t_{DDINf}	4-30	$\overline{DDIN}$ floating (caused by $\overline{BRI}$)	after R.E., PHI1 T1		100		80		55	ns
t_{DDINr}	4-30	$\overline{DDIN}$ return from floating (caused by $\overline{BRI}$)	after R.E., PHI1 T1		100		80		55	ns
t_{ALf}	4-30	AD0–AD23 floating (caused by $\overline{BRI}$)	after R.E., PHI1 T4		45		35		25	ns
t_{ADf}	4-30	D24–D31 Floating (caused by $\overline{BRI}$)	after R.E., PHI1 T4		45		35		25	ns

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

4.4.2.2 Input Signal Requirements: NS32132-6, NS32132-8, NS32132-10

Name	Figure	Description	Reference/Conditions	NS32132-6		NS32132-8		NS32132-10		Units
				Min	Max	Min	Max	Min	Max	
t_{PWR}	4-25	Power stable to $\overline{RST}$ T.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	4-5	Data in setup (read cycle)	before F.E., PHI2 T3	20		15		10		ns
t_{DIh}	4-5	Data in hold (read cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	4-6	$\overline{HOLD}$ active (low) setup time (see note)	before F.E., PHI2 TX1	25		25		25		ns
t_{HLDia}	4-8	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	4-6	$\overline{HOLD}$ hold time	after R.E., PHI1 TX2	0		0		0		ns
t_{FLTa}	4-9	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25		25		25		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements NS32132-6, NS32132-8, NS32132-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32132-6		NS32132-8		NS32132-10		Units
				Min	Max	Min	Max	Min	Max	
t_{FLTia}	4-10	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T2	25		25		25		ns
t_{RDYs}	4-11, 4-12	RDY setup time	before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	4-11, 4-12	RDY hold time	after F.E., PHI1 T3	0		0		0		ns
t_{ABTs}	4-23	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t_{ABTs}	4-24	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 T2	30		25		20		ns
t_{ABTh}	4-23	$\overline{ABT}$ hold time	after R.E., PHI1	0		0		0		ns
t_{RSTs}	4-25, 4-26	$\overline{RST}$ setup time	before F.E., PHI1	20		15		10		ns
t_{RSTw}	4-26	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	4-27	$\overline{INT}$ setup time	before F.E., PHI1	20		20		20		ns
t_{NMIw}	4-28	$\overline{NMI}$ pulse width	at 0.8V (both edges)	70		70		70		ns
t_{DIs}	4-14	Data setup (slave read cycle)	before F.E., PHI2 T1	20		20		10		ns
t_{DIh}	4-14	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	4-15	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	17		13		10		ns
t_{SPCs}	4-15	$\overline{SPC}$ setup time	before F.E., PHI1	42		32		25		ns
t_{SPCw}	4-15	$\overline{SPC}$ pulse width (from slave processor)	at 0.8V (both edges)	30		25		20		ns
t_{ATs}	4-16	$\overline{AT}/\overline{SPC}$ setup for address translation strap	before R.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1		1		1		t_{Cp}
t_{ATh}	4-16	$\overline{AT}/\overline{SPC}$ hold for address translation strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		t_{Cp}
t_{BRis}	4-20	$\overline{BRI}$ Setup Time	before F.E. PHI2	15		15		15		ns
t_{BRin}	4-20	$\overline{BRI}$ Hold Time	after R.E. PHI1	10		10		10		ns

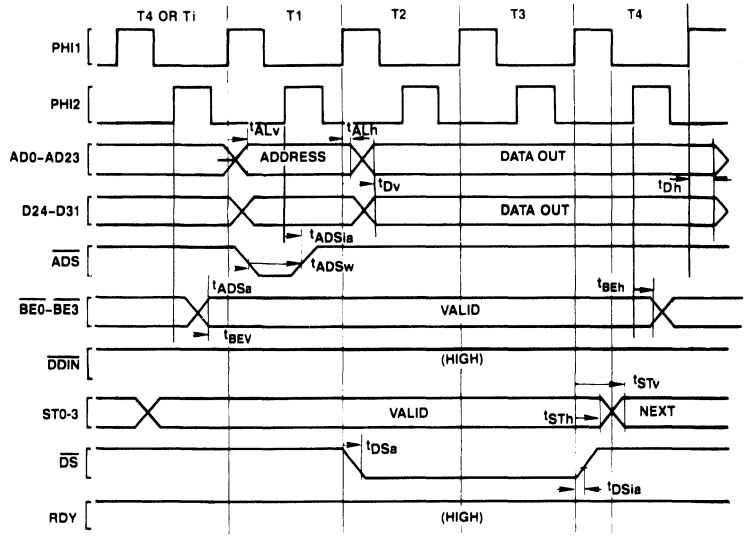
Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

4.4.2.3 Clocking Requirements: NS32132-6, NS32132-8, NS32132-10

Name	Figure	Description	Reference/ Conditions	NS32132-6		NS32132-8		NS32132-10		Units
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4-17	PHI1, PHI2 rise time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	4-17	PHI1, PHI2 fall time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	4-17	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4-17	PHI1, PHI2 pulse width	At 2.0V on PHI1, PHI2 (both edges)	$0.5t_{Cp}-14$		$0.5t_{Cp}-12$		$0.5t_{Cp}-10$		ns
$t_{CLh(1,2)}$	4-17	PHI1, PHI2 high time	At V_{CC} − 0.9V on PHI1, PHI2 (both edges)	$0.5t_{Cp}-18$		$0.5t_{Cp}-17$		$0.5t_{Cp}-15$		ns
$t_{nOVL(1,2)}$	4-17	Non-overlap time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)}-t_{nOVL(2)}$)	at 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}		PHI1, PHI2 asymmetry ($t_{CLw(1)}-t_{CLw(2)}$)	at 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

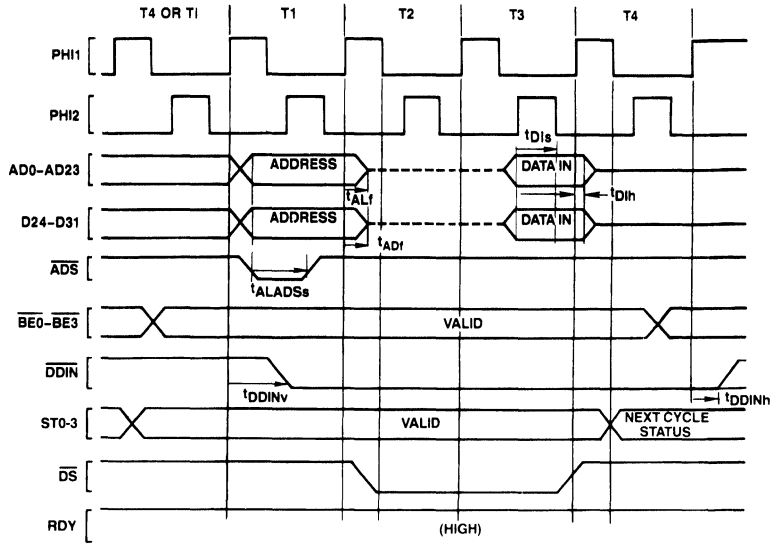
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



TL/EE/8583-44

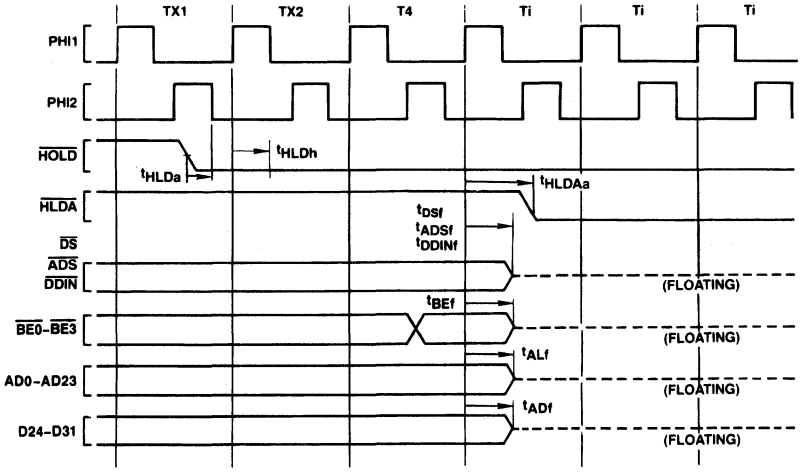
FIGURE 4-4. Write Cycle



TL/EE/8583-45

FIGURE 4-5. Read Cycle

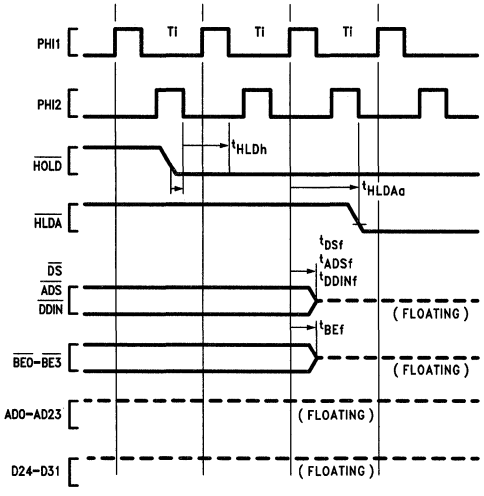
4.0 Device Specifications (Continued)



TL/EE/8583-46

FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially)

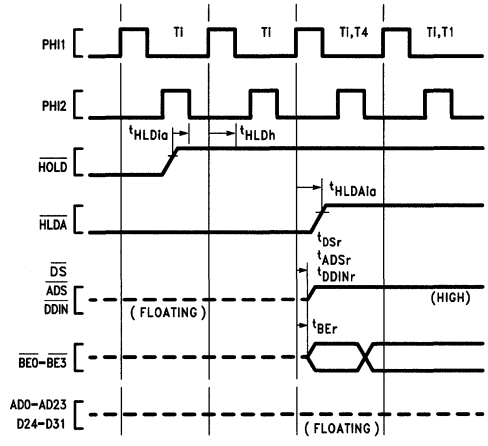
Note that whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ request ($\overline{\text{HOLD}}$ low) must be active tHLDa before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until tHLDh after the rising edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.



TL/EE/8583-47

FIGURE 4-7. Floating by $\overline{\text{HOLD}}$ Timing (CPU initially idle)

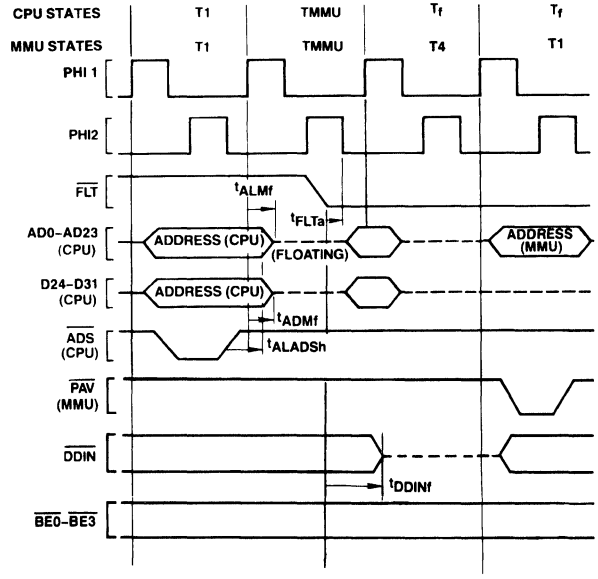
Note that during Ti1 the CPU is already idling.



TL/EE/8583-48

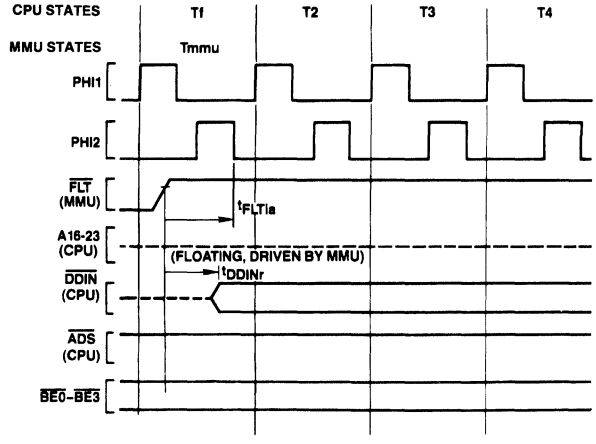
FIGURE 4-8. Release from $\overline{\text{HOLD}}$

4.0 Device Specifications (Continued)



TL/EE/8583-49

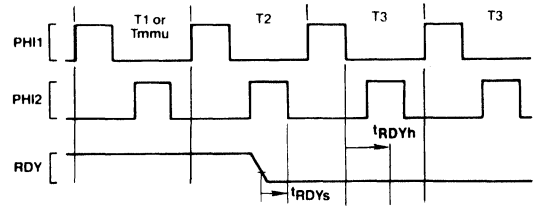
FIGURE 4-9. $\overline{FLT}$ Initiated Float Cycle Timing



TL/EE/8583-50

FIGURE 4-10. Release from $\overline{FLT}$ Timing

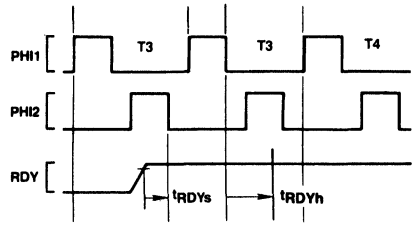
Note that when $\overline{FLT}$ is deasserted the CPU restarts driving $\overline{DDIN}$ before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force $\overline{DDIN}$ to the same logic level.



TL/EE/8583-51

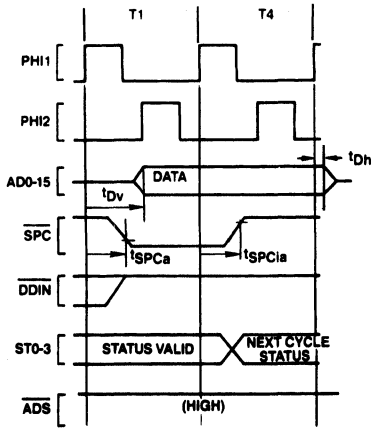
FIGURE 4-11. Ready Sampling (CPU Initially READY)

4.0 Device Specifications (Continued)



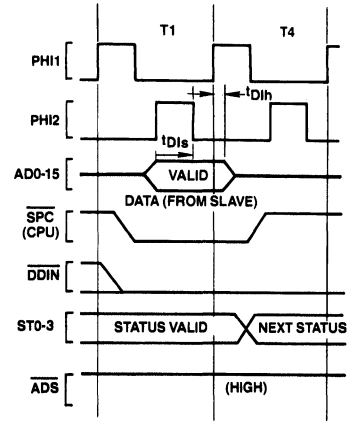
TL/EE/8583-52

FIGURE 4-12. Ready Sampling (CPU Initially NOT READY)



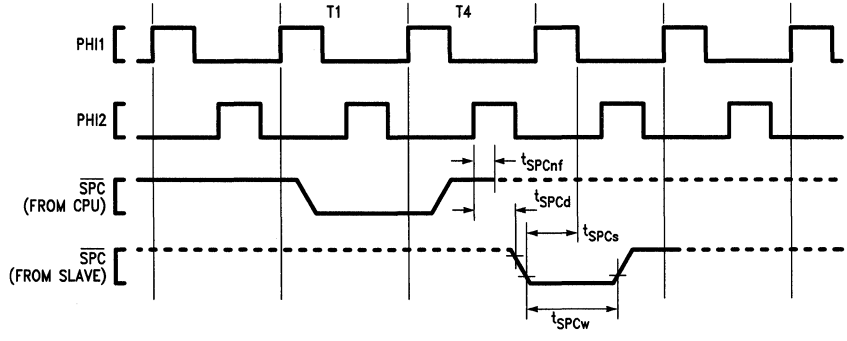
TL/EE/8583-53

FIGURE 4-13. Slave Processor Write Timing



TL/EE/8583-54

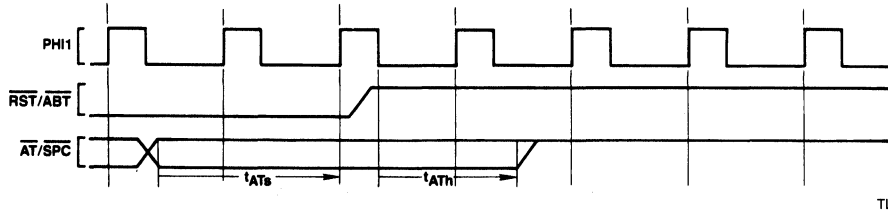
FIGURE 4-14. Slave Processor Read Timing



TL/EE/8583-83

FIGURE 4-15. $\overline{SPC}$ Timing

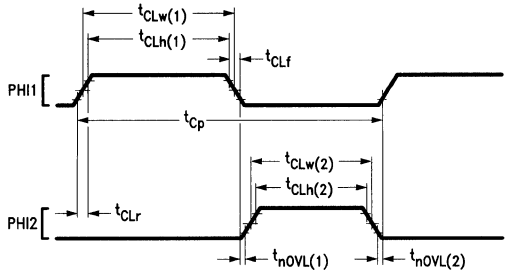
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{SPC}$ high with internal 5 kΩ pullup.



TL/EE/8583-56

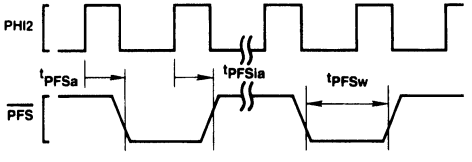
FIGURE 4-16. Reset Configuration Timing

4.0 Device Specifications (Continued)



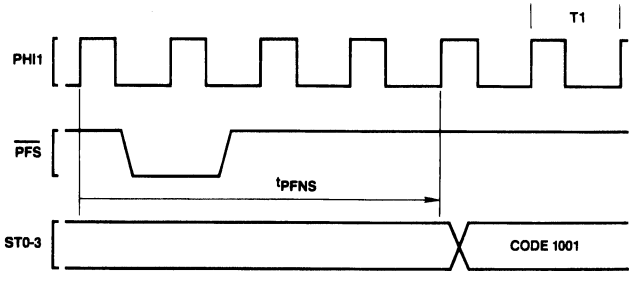
TL/EE/8583–57

FIGURE 4-17. Clock Waveforms



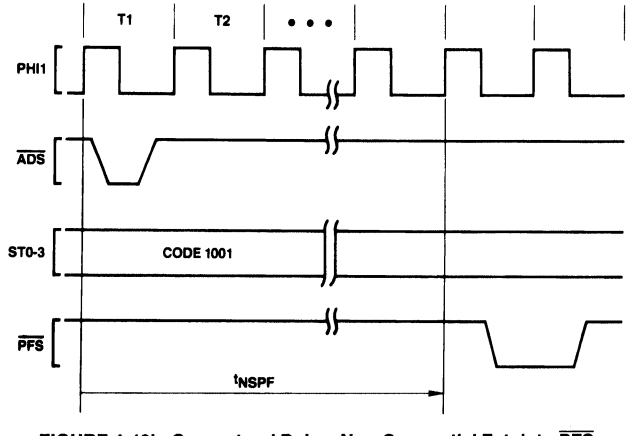
TL/EE/8583–58

FIGURE 4-18. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/8583–59

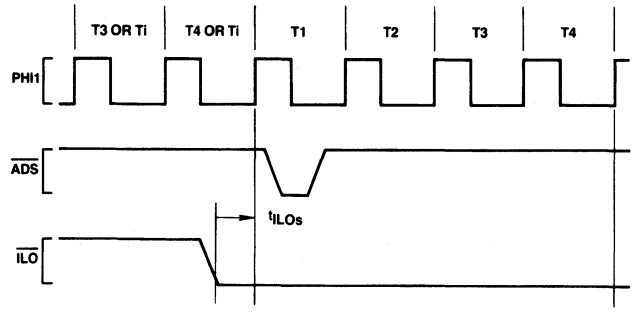
FIGURE 4-19a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/8583–60

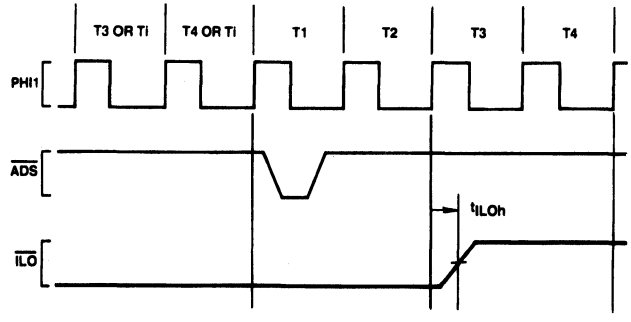
FIGURE 4-19b. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



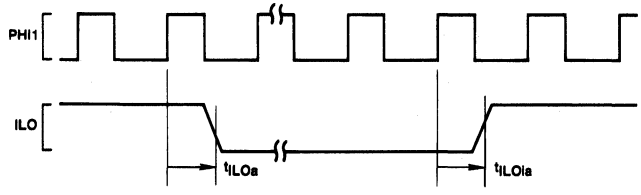
TL/EE/8583-61

FIGURE 4-20a. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction



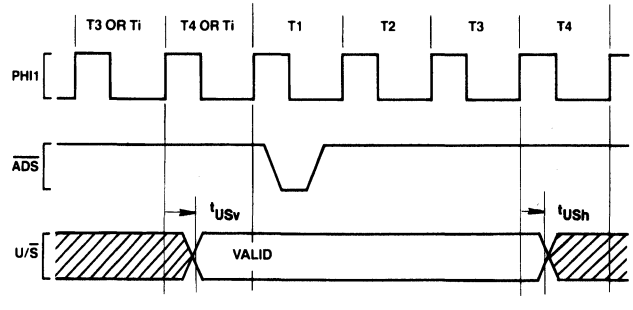
TL/EE/8583-62

FIGURE 4-20b. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction



TL/EE/8583-63

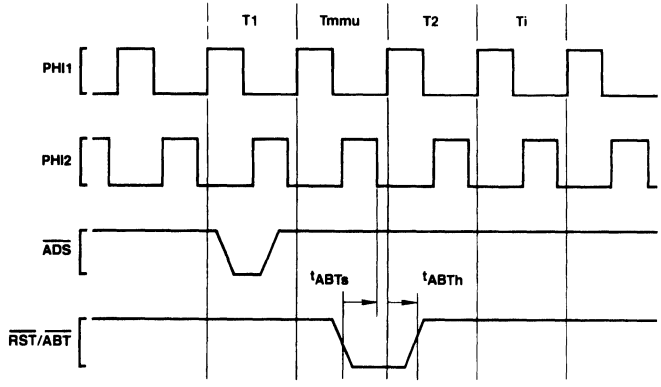
FIGURE 4-21. Relationship of $\overline{ILO}$ to Any Clock Cycle



TL/EE/8583-64

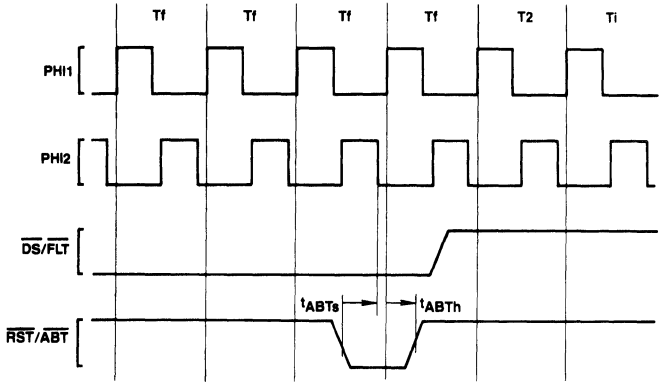
FIGURE 4-22. U/$\overline{S}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval

4.0 Device Specifications (Continued)



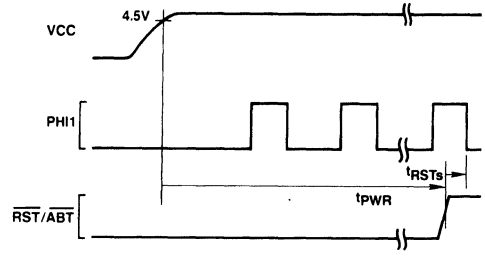
TL/EE/8583-65

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Not Applied



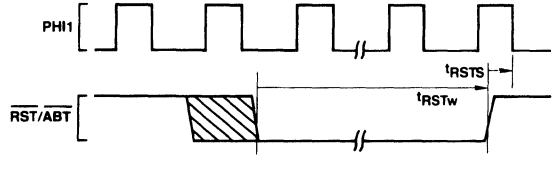
TL/EE/8583-66

FIGURE 4-24. Abort Timing, $\overline{FLT}$ Applied



TL/EE/8583-67

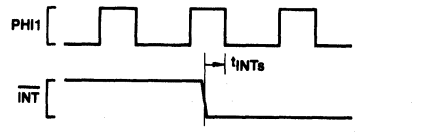
FIGURE 4-25. Power-On Reset



TL/EE/8583-68

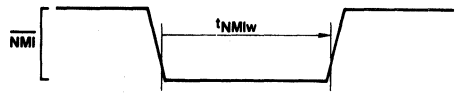
FIGURE 4-26. Non-Power-On Reset

4.0 Device Specifications (Continued)



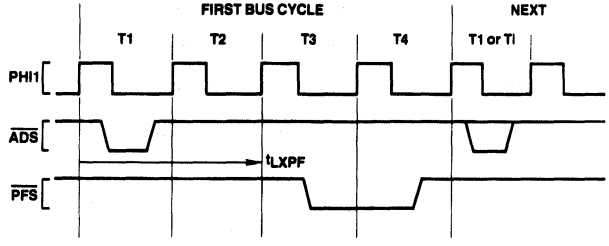
TL/EE/8583–69

FIGURE 4-27. $\overline{INT}$ Interrupt Signal Detection



TL/EE/8583–70

FIGURE 4-28. $\overline{NMI}$ Interrupt Signal Timing

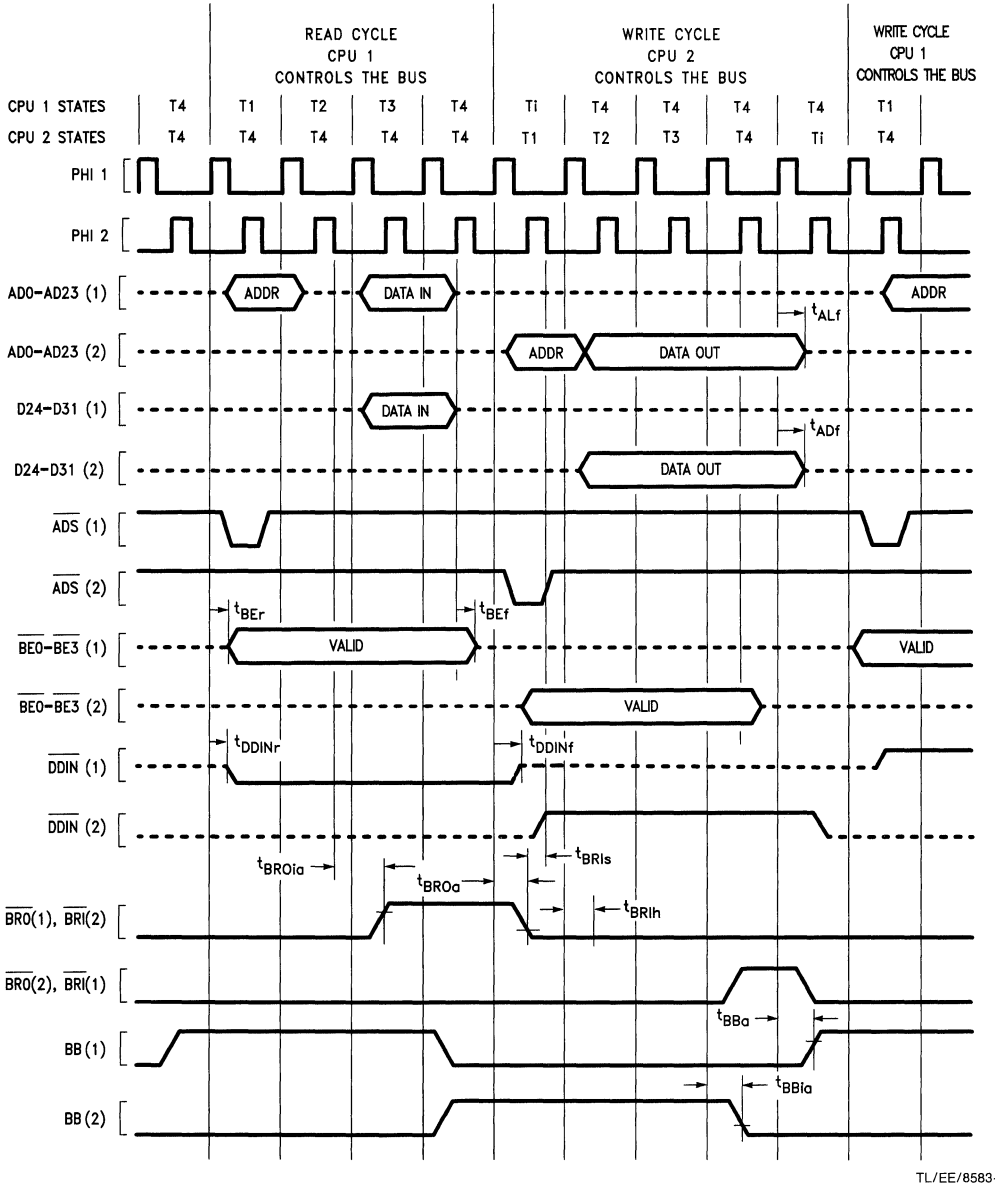


TL/EE/8583–71

FIGURE 4-29. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction

Note: In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

4.0 Device Specifications (Continued)



TL/EE/8583-72

FIGURE 4-30. Dual Processor Bus Arbitration Timing

Appendix A: Instruction Formats

NOTATIONS

i= Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f= Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c= Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op= Operation Code
Valid encodings shown with each format.

gen, gen 1, gen 2= General Addressing Mode Field
See Sec. 2.2 for encodings.

reg= General Purpose Register Number

cond= Condition Code Field
- 0000 = EQual: Z = 1
- 0001 = Not Equal: Z = 0
- 0010 = Carry Set: C = 1
- 0011 = Carry Clear: C = 0
- 0100 = HIgher: L = 1
- 0101 = Lower or Same: L = 0
- 0110 = Greater Than: N = 1
- 0111 = Less or Equal: N = 0
- 1000 = Flag Set: F = 1
- 1001 = Flag Clear: F = 0
- 1010 = LOwer: L = 0 and Z = 0
- 1011 = Higher or Same: L = 1 or Z = 1
- 1100 = Less Than: N = 0 and Z = 0
- 1101 = Greater or Equal: N = 1 or Z = 1
- 1110 = (Unconditionally True)
- 1111 = (Unconditionally False)

short= Short Immediate value. May contain
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.
  - 0000 = US
  - 0001 – 0111 = (Reserved)
  - 1000 = FP
  - 1001 = SP
  - 1010 = SB
  - 1011 = (Reserved)
  - 1100 = (Reserved)
  - 1101 = PSR
  - 1110 = INTBASE
  - 1111 = MOD
- Options: in String Instructions

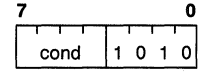
U/W	B	T

  - T = Translated
  - B = Backward
  - U/W = 00: None
    - 01: While Match
    - 11: Until Match

Configuration bits, in SETCFG:

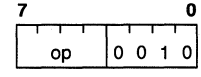
C	M	F	I

mreg: MMU Register number, in LMR, SMR.
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = PF0
- 0101 = PF1
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = SC
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA



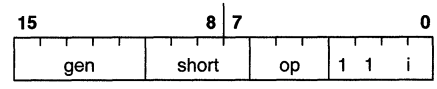
Format 0

Bcond (BR)



Format 1

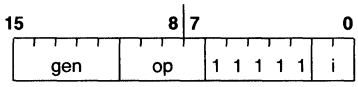
BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111



Format 2

ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Scond	-011		

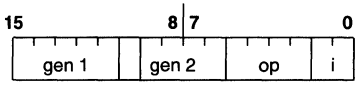
Appendix A: Instruction Formats (Continued)



Format 3

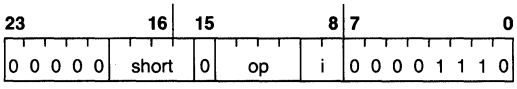
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

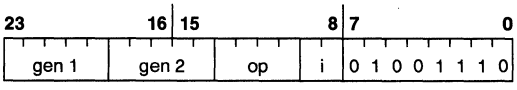
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

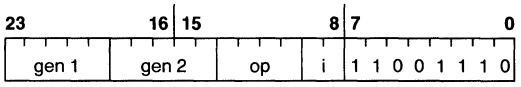
MOVS	-0000	SETCFG	-0010
CMPS	-0001	SKPS	-0011

Trap (UND) on 1XXX, 01XX



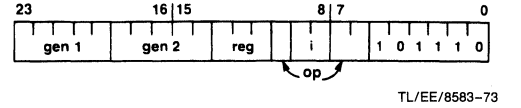
Format 6

ROT	-0000	NEG	-1000
ASH	-0001	NOT	-1001
CBIT	-0010	Trap (UND)	-1010
CBITI	-0011	SUBP	-1011
Trap (UND)	-0100	ABS	-1100
LSH	-0101	COM	-1101
SBIT	-0110	IBIT	-1110
SBITI	-0111	ADDP	-1111



Format 7

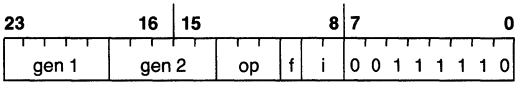
MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZiD	-0110	MOD	-1110
MOVXiD	-0111	DIV	-1111



TL/EE/8583-73

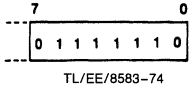
Format 8

EXT	-0 00	INDEX	-1 00
CVTP	-0 01	FFS	-1 01
INS	-0 10		
CHECK	-0 11		
MOVSU	-110, reg = 001		
MOVUS	-110, reg = 011		



Format 9

MOVif	-000	ROUND	-100
LFSR	-001	TRUNC	-101
MOVLF	-010	SFSR	-110
MOVFL	-011	FLOOR	-111

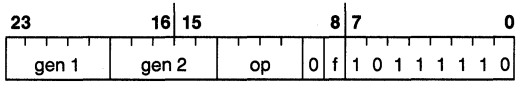


TL/EE/8583-74

Format 10

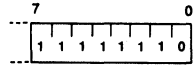
Trap (UND) Always

Appendix A: Instruction Formats (Continued)



Format 11

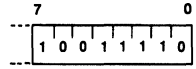
ADDf	-0000	DIVf	-1000
MOVf	-0001	Trap (Slave)	-1001
CMPf	-0010	Trap (UND)	-1010
Trap (Slave)	-00T1	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



TL/EE/8583–75

Format 12

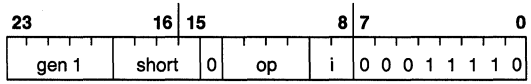
Trap (UND) Always



TL/EE/8583–76

Format 13

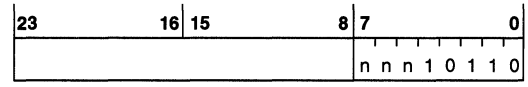
Trap (UND) Always



Format 14

RDVAL	-0000	LMR	-1010
WRVAL	-0001	SMR	-1011

Trap (UND) on 01XX, 1XXX



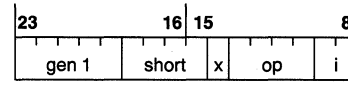
Operation Word ID Byte

Format 15

(Custom Slave)

nnn **Operation Word Format**

000

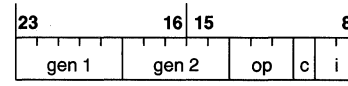


Format 15.0

CATST0	-0000	LCR	-1010
CATST1	-0001	SCR	-1011

Trap (UND) on all others

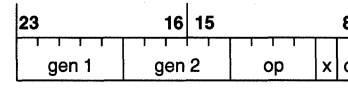
001



Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111

101

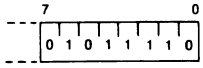


Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	CMOV3	-1001
CCMP	-0010	Trap (UND)	-1010
CCMP1	-0011	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

If nnn = 010, 011, 100, 110, 111
then Trap (UND) Always

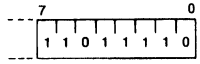
Appendix A: Instruction Formats (Continued)



TL/EE/8583–77

Format 16

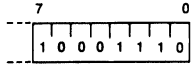
Trap (UND) Always



TL/EE/8583–78

Format 17

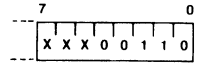
Trap (UND) Always



TL/EE/8583–79

Format 18

Trap (UND) Always

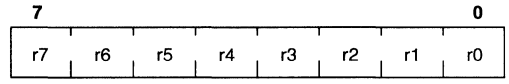


TL/EE/8583–80

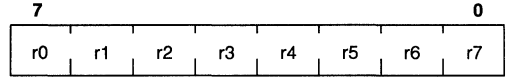
Format 19

Trap (UND) Always

Implied Immediate Encodings:



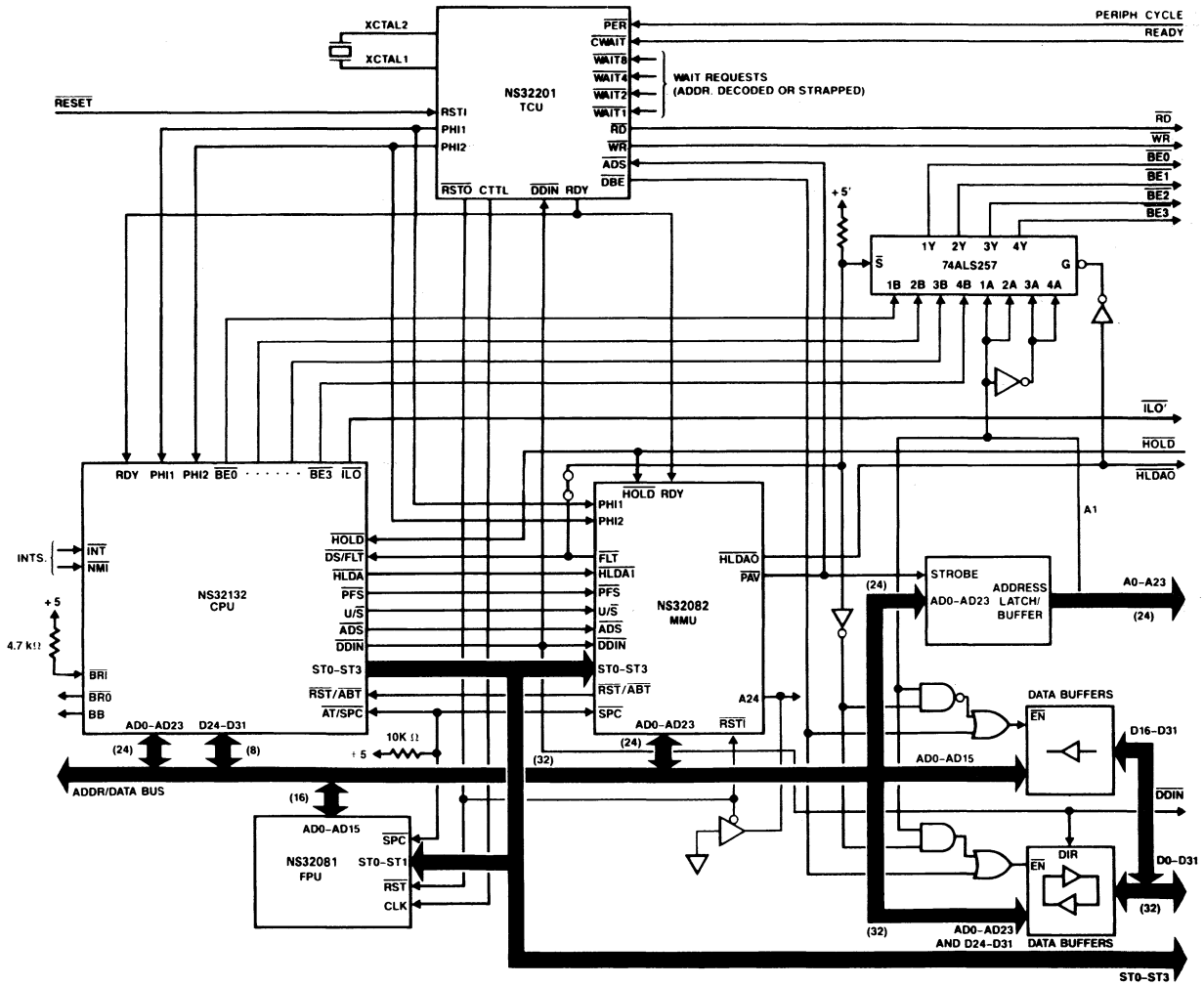
Register Mark, appended to SAVE, ENTER



Register Mark, appended to RESTORE, EXIT

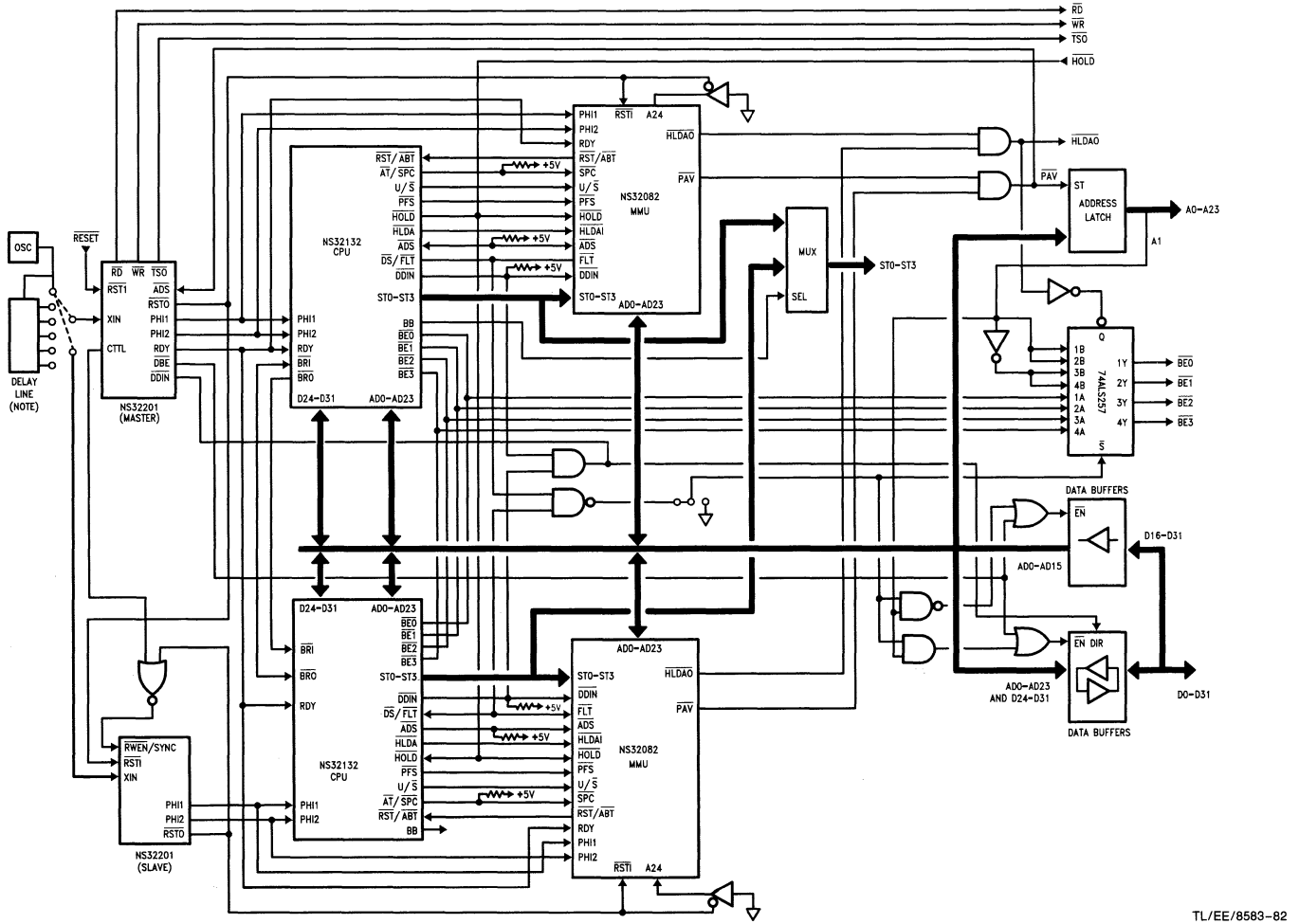
Offset/Length Modifier appended to INSS, EXTS

2



TL/EE/8583-81

FIGURE B-1. Single Processor System Connection Diagram



TL/EE/8583-82

FIGURE B-2. Dual Processor System Connections Diagram

Note: The delay line is used to deskew the clock outputs from the two TCUs in order to avoid speed degradation.

National Semiconductor Corporation

NS32032-6/NS32032-8/NS32032-10 High-Performance Microprocessors

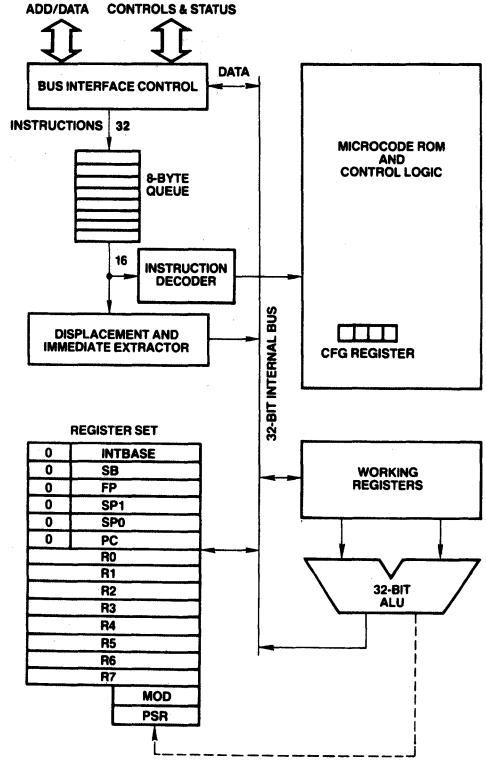
General Description

The NS32032 is a 32-bit, virtual memory microprocessor with a 16-MByte linear address space and a 32-bit external data bus. It has a 32-bit ALU, eight 32-bit general purpose registers, an eight-byte prefetch queue, and a slave processor interface. The NS32032 is fabricated with National Semiconductor's advanced XMOS™ process, and is fully object code compatible with other Series 32000® processors. The Series 32000 instruction set is optimized for modular, high-level languages (HLL). The set is very symmetric, it has a two address format, and it incorporates HLL oriented addressing modes. The capabilities of the NS32032 can be expanded with the use of the NS32081 floating point unit (FPU), and the NS32082 demand-paged virtual memory management unit (MMU). Both devices interface to the NS32032 as slave processors. The NS32032 is a general purpose microprocessor that is ideal for a wide range of computational intensive applications.

Features

- 32-bit architecture and implementation
- Virtual memory support
- 16-MByte linear address space
- 32-bit data bus
- Powerful instruction set
  - — General 2-address capability
  - — Very high degree of symmetry
  - — Addressing modes optimized for high-level languages
- Series 32000 slave processor support
- High-speed XMOS technology
- 68-pin leadless chip carrier

Block Diagram



TL/EE/5491–1

FIGURE 1

Table of Contents

1.0 PRODUCT INTRODUCTION

2.0 ARCHITECTURAL DESCRIPTION

2.1 Programming Model
- 2.1.1 General Purpose Registers
- 2.1.2 Dedicated Registers
- 2.1.3 The Configuration Register (CFG)
- 2.1.4 Memory Organization
- 2.1.5 Dedicated Tables

2.2 Instruction Set
- 2.2.1 General Instruction Format
- 2.2.2 Addressing Modes
- 2.2.3 Instruction Set Summary

3.0 FUNCTIONAL DESCRIPTION

3.1 Power and Grounding
3.2 Clocking
3.3 Resetting
3.4 Bus Cycles
- 3.4.1 Cycle Extension
- 3.4.2 Bus Status
- 3.4.3 Data Access Sequences
  - 3.4.3.1 Bit Accesses
  - 3.4.3.2 Bit Field Accesses
  - 3.4.3.3 Extending Multiply Accesses
- 3.4.4 Instruction Fetches
- 3.4.5 Interrupt Control Cycles
- 3.4.6 Slave Processor Communication
  - 3.4.6.1 Slave Processor Bus Cycles
  - 3.4.6.2 Slave Operand Transfer Sequences

3.5 Memory Management Option
- 3.5.1 Address Translation Strap
- 3.5.2 Translated Bus Timing
- 3.5.3 The $\overline{FLT}$ (Float) Pin
- 3.5.4 Aborting Bus Cycles
  - 3.5.4.1 The Abort Interrupt
  - 3.5.4.2 Hardware Considerations

3.6 Bus Access Control
3.7 Instruction Status

3.0 FUNCTIONAL DESCRIPTION (Continued)

3.8 NS32032 Interrupt Structure
- 3.8.1 General Interrupt/Trap Sequence
- 3.8.2 Interrupt/Trap Return
- 3.8.3 Maskable Interrupts (The $\overline{INT}$ Pin)
  - 3.8.3.1 Non-Vectored Mode
  - 3.8.3.2 Vectored Mode: Non-Cascaded Case
  - 3.8.3.3 Vectored Mode: Cascaded Case
- 3.8.4 Non-Maskable Interrupt (The $\overline{NMI}$ Pin)
- 3.8.5 Traps
- 3.8.6 Prioritization
- 3.8.7 Interrupt/Trap Sequences Detailed Flow
  - 3.8.7.1 Maskable/Non-Maskable Interrupt Sequence
  - 3.8.7.2 Trap Sequence: Traps Other Than Trace
  - 3.8.7.3 Trace Trap Sequence
  - 3.8.7.4 Abort Sequence

3.9 Slave Processor Instructions
- 3.9.1 Slave Processor Protocol
- 3.9.2 Floating Point Instructions
- 3.9.3 Memory Management Instructions
- 3.9.4 Custom Slave Instructions

4.0 DEVICE SPECIFICATIONS

4.1 Pin Descriptions
- 4.1.1 Supplies
- 4.1.2 Input Signals
- 4.1.3 Output Signals
- 4.1.4 Input/Output Signals

4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics
- 4.4.1 Definitions
- 4.4.2 Timing Tables
  - 4.4.2.1 Output Signals: Internal Propagation Delays
  - 4.4.2.2 Input Signals Requirements
  - 4.4.2.3 Clocking Requirements
- 4.4.3 Timing Diagrams

Appendix A: Instruction Formats
Appendix B: Interfacing Suggestions

List of Illustrations

CPU Block Diagram 1-1
The General and Dedicated Registers 2-1
Processor Status Register 2-2
CFG Register 2-3
Module Descriptor Format 2-4
A Sample Link Tabale 2-5
General Instruction Format 2-6
Index Byte Format 2-7
Displacement Encodings 2-8
Recommended Supply Connections 3-1
Clock Timing Relationships 3-2
Power-On Reset Requirements 3.3
General Reset Timing 3-4
Recommended Reset Connections, Non-Memory-Managed System 3-5a
Recommended Reset Connections, Memory-Managed System 3-5b

List of Illustrations (Continued)

Bus Connections ... 3-6
Read Cycle Timing ... 3-7
Write Cycle Timing ... 3-8
RDY Pin Timing ... 3-9
Extended Cycle Example ... 3-10
Memory Interface ... 3-11
Slave Processor Connections ... 3-12
CPU Read from Slave Processor ... 3-13
CPU Write to Slave Processor ... 3-14
Read Cycle with Address Translation (CPU Action) ... 3-15
Write Cycle with Address Translation (CPU Action) ... 3-16
Memory-Managed Read Cycle ... 3-17
Memory-Managed Write Cycle ... 3-18
$\overline{FLT}$ Timing ... 3-19
$\overline{HOLD}$ Timing, Bus Initially Idle ... 3-20
$\overline{HOLD}$ Timing, Bus Initially Not Idle ... 3-21
Interrupt Dispatch and Cascade Tables ... 3-22
Interrupt/Trap Service Routine Calling Sequence ... 3-23
Return from Trap (RETT n) Instruction Flow ... 3-24
Return from Interrupt (RET) Instruction Flow ... 3-25
Interrupt Control Connections (16 levels) ... 3-26
Cascaded Interrupt Control Unit Connections ... 3-27
Service Sequence ... 3-28
Slave Processor Protocol ... 3-29
Slave Processor Status Word Format ... 3-30
NS32032 Connection Diagram ... 4-1
Timing Specification Standard (Signal Valid After Clock Edge) ... 4-2
Timing Specification Standard (Signal Valid Before Clock Edge) ... 4-3
Write Cycle ... 4-4
Read Cycle ... 4-5
Floating by $\overline{HOLD}$ Timing (CPU Not Initially Idle) ... 4-6
Floating by $\overline{HOLD}$ Timing (CPU Initially Idle) ... 4-7
Release from Hold ... 4-8
$\overline{FLT}$ Initiated Float Cycle Timing ... 4-9
Release from $\overline{FLT}$ Timing ... 4-10
Ready Sampling (CPU Initially READY) ... 4-11
Ready Sampling (CPU Initially NOT READY) ... 4-12
Slave Processor Write Timing ... 4-13
Slave Processor Read Timing ... 4-14
$\overline{SPC}$ Timing ... 4-15
Reset Configuration Timing ... 4-16
Clock Waveforms ... 4-17
Relationship of $\overline{PFS}$ to Clock Cycles ... 4-18
Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch ... 4-19a
Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$ ... 4-19b
Relationship of $\overline{ILO}$ to First Operand of an Interlocked Instruction ... 4-20a
Relationship of $\overline{ILO}$ to Last Operand of an Interlocked Instruction ... 4-20b
Relationship of $\overline{ILO}$ to Any Clock Cycle ... 4-21
U/$\overline{S}$ Relationship to any Bus Cycle — Guaranteed Valid Interval ... 4-22
Abort Timing, $\overline{FLT}$ Not Applied ... 4-23
Abort Timing, $\overline{FLT}$ Applied ... 4-24
Power-On Reset ... 4-25
Non-Power-On Reset ... 4-26
$\overline{INT}$ Interrupt Signal Detection ... 4-27
$\overline{MNI}$ Interrupt Signal Timing ... 4-28
Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction ... 4-29
Processor System Connection Diagram ... B-1

List of Tables

NS32032 Addressing Modes 2-1
NS32032 Instruction Set Summary 2-2
Bus Access Type 3-1
Access Sequence 3-2
Interrupt Sequences 3-3
Floating Point Instruction Protocols 3-4
Memory Management Instruction Protocols 3-5
Custom Slave Instruction Protocols 3-6

1.0 Product Introduction

The Series 32000 microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from a minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operations. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes. Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types. The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set. While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations. The Series 32000 CPUs represent two-operand machines with each operand addressable by all addressing modes. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management. Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing. The NS32032 has 24-bit address pointers that can address up to 16 megabytes without requiring any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support. Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept. The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32032 CPU.

2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

2.1.2 Dedicated Registers

The eight dedicated registers of the NS32032 are assigned specific functions.

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32032 the upper eight bits of this register are always zero.)

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 the SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32032 the upper eight bits of these registers are always zero).

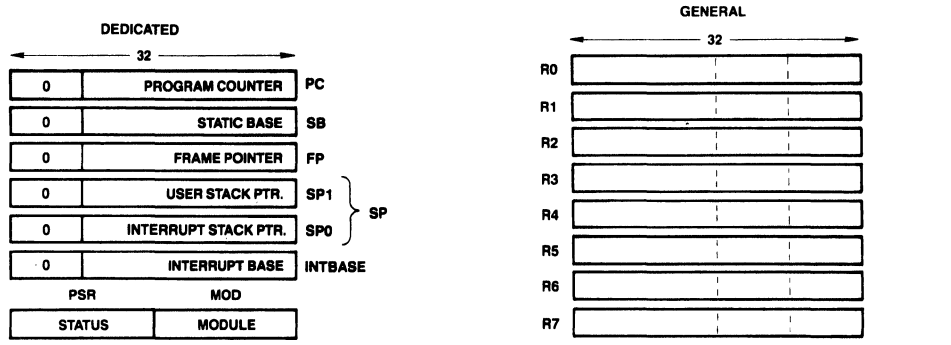
Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.

FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32032 the upper eight bits of this register are always zero.)

SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules.

2.0 Architectural Description (Continued)



TL/EE/5491–3

FIGURE 2-1. The General and Dedicated Registers

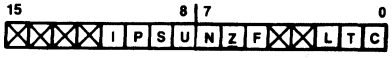
The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32032 the upper eight bits of this register are always zero.)

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32032 the upper eight bits of this register are always zero.)

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

PSR: The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the NS32032 microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/5491–4

FIGURE 2-2. Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Sec. 3.8.5).

L: The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the NS32032 is said to be in Supervisor Mode; when U = 1 the NS32032 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Sec. 3.8.5.). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I = 1, then all interrupts will be accepted (Sec. 3.8.). If I = 0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32032 CPU is the four-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3*.

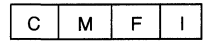


FIGURE 2-3. CFG Register

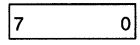
2.0 Architectural Description (Continued)

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{INT}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

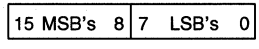
2.1.4 Memory Organization

The main memory of the NS32032 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at 2^{24} - 1. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.



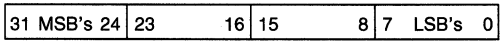
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Sec. 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.



Word at Address A

Two contiguous words are called a double word. Except where noted (Sec. 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



Double Word at Address A

Although memory is addressed as bytes, it is actually organized as double-words. Note that access time to a word or a double-word depends upon its address, e.g. double-words that are aligned to start at addresses that are multiples of four will be accessed more quickly than those not so aligned. This also applies to words that cross a double-word boundary.

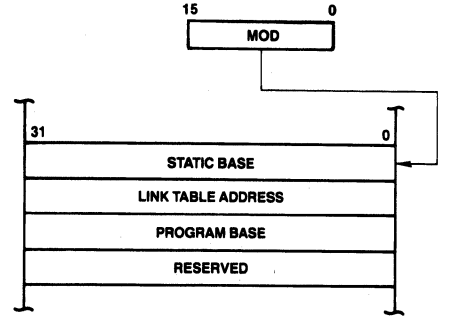
2.1.5 Dedicated Tables

Two of the NS32032 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Sec. 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by NS32032. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically up-dated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in *Figure 2-4*. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.



TL/EE/5491–5

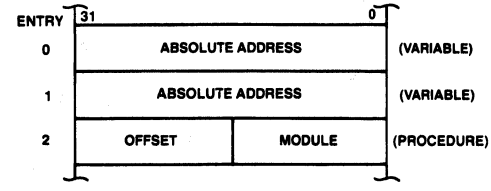
FIGURE 2-4. Module Descriptor Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-5*. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/5491–6

FIGURE 2-5. A Sample Link Table

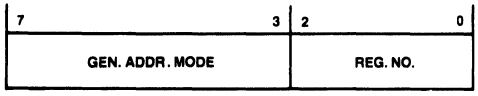
2.0 Architectural Description (Continued)

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-7*.



TL/EE/5491-8

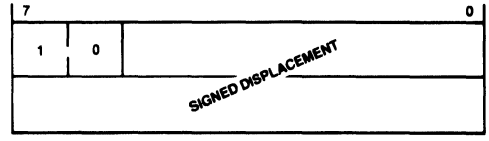
FIGURE 2-7. Index Byte Format

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected address modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded with the top bits of that field, as shown in *Figure 2-8*, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first. Note that this is different from the memory representation of data (Sec. 2.1.4).

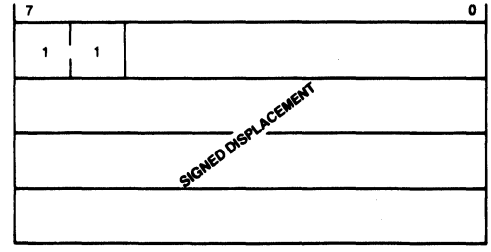
Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Sec. 2.2.3).

Byte Displacement: Range −64 to +63

Word Displacement: Range −8192 to +8191



Double Word Displacement: Range (Entire Addressing Space)

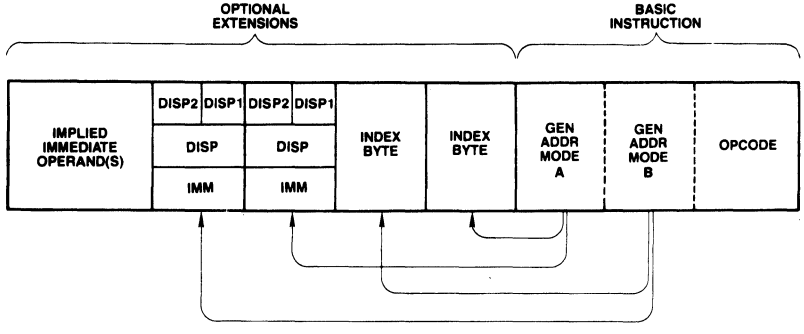


TL/EE/5491-11

FIGURE 2-8. Displacement Encodings

2.2.2 Addressing Modes

The NS32032 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."



TL/EE/5491-7

FIGURE 2-6. General Instruction Format

2.0 Architectural Description (Continued)

Addressing modes in the NS32032 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32032 Addressing Modes fall into nine basic types:

Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

Memory Space. Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode. Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32032 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0–R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

2.0 Architectural Description (Continued)

TABLE 2-1
NS32032 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2× Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4× Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn. 'Mode' and 'n' are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

TABLE 2-2
NS32032 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move Effective Address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDI	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to Extended Integer.
7	DEIi	gen,gen	Divide Extended Integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add Packed.
6	SUBPi	gen,gen	Subtract Packed.

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare Multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical Exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

2.0 Architectural Description (Continued)

TABLE 2-2 (Continued)
NS32032 Instruction Set Summary (Continued)

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical Shift, left or right.
6	ASHi	gen,gen	Arithmetic Shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to Bit Field Pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 - Comparison Value

R3 - Translation Table Pointer

R2 - String 2 Pointer

R1 - String 1 Pointer

R0 - Limit Count

Options on all string instructions are:

B (Backward): Decrement string pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Descriptions
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare translating, String 1 bytes.
5	SKPSi	options	Skip over String 1 entries
	SKPST	options	Skip, translating bytes for Until/While.

2.0 Architectural Description (Continued)

TABLE 2-2 (Continued)
NS32032 Instruction Set Summary (Continued)

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor Call.
1	FLAG		Flag Trap.
1	BPT		Breakpoint Trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save General Purpose Registers.
1	RESTORE	[reg list]	Restore General Purpose Registers.
2	LPRi	areg,gen	Load Dedicated Register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store Dedicated Register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust Stack Pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set Configuration Register. (Privileged)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a Floating Point value.
9	MOVLF	gen,gen	Move and shorten a Long value to Standard.
9	MOVFL	gen,gen	Move and lengthen a Standard value to Long.
9	MOVif	gen,gen	Convert any integer to Standard or Long Floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load Memory Management Register. (Privileged)
14	SMR	mreg,gen	Store Memory Management Register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUi	gen,gen	Move a value from Supervisor Space to User Space. (Privileged)
8	MOVUSi	gen,gen	Move a value from User Space to Supervisor Space. (Privileged)

2.0 Architectural Description (Continued)

TABLE 2-2 (Continued)
NS32032 Instruction Set Summary (Continued)

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No Operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom Calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom Move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
	CMOV3c	gen,gen	
15.5	CCMP0c	gen,gen	Custom Compare.
	CCMP1c	gen,gen	
15.1	CCV0ci	gen,gen	Custom Convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load Custom Status Register.
15.1	SCSR	gen	Store Custom Status Register.
15.0	CATST0	gen	Custom Address/Test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load Custom Register. (Privileged)
15.0	SCR	creg,gen	Store Custom Register. (Privileged)

3.0 Functional Description

3.1 POWER AND GROUNDING

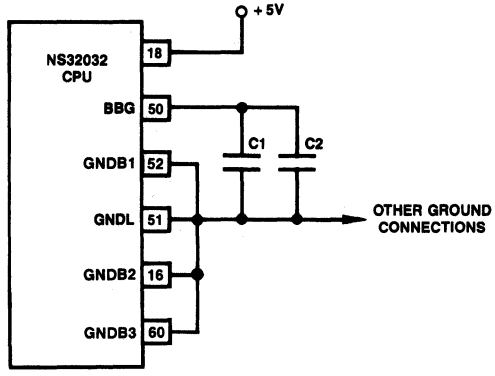
The NS32032 requires a single 5-volt power supply, applied on pin 18 (V_{CC}).

Grounding connections are made on four pins. Logic Ground (GNDL, pin 54) is the common pin for on-chip logic, and Buffer Grounds (GNDB1, pin 52 and GNDB2, pin 16 and GNDB3, pin 60) (16) are the common pins for the output drivers. For optimal noise immunity it is recommended that GNDB1 and GNDB2 be connected together through a single conductor, and GNDL be directly connected to the middle point of this conductor. All other ground connections should be made to the common line as shown in *Figure 3-1*.

In addition to V_{CC} and Ground, the NS32032 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Fig. 3-1*) from the BBG pin to ground. Recommended values for these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.



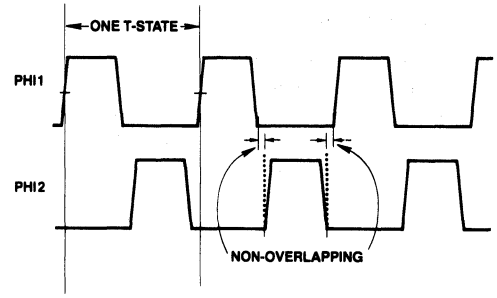
TL/EE/5491-12

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32032 inputs clocking signals from the Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in *Figure 3-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See Section 4 for complete specifications of PHI1 and PHI2.



TL/EE/5491-13

FIGURE 3-2. Clock Timing Relationships

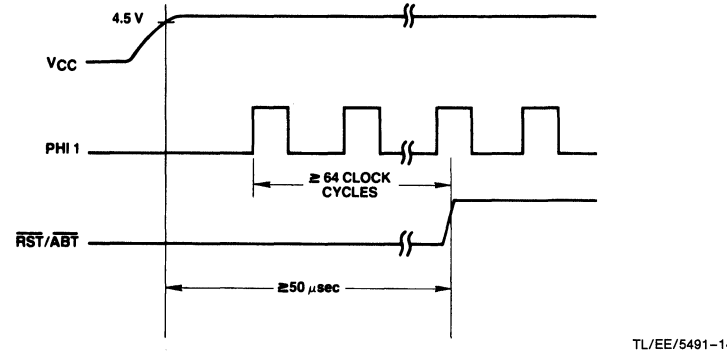
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 RESETTING

The $\overline{RST}/\overline{ABT}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.4.

The CPU may be reset at any time by pulling the $\overline{RST}/\overline{ABT}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, $\overline{RST}/\overline{ABT}$ must be held low for at least 50 μsec after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain



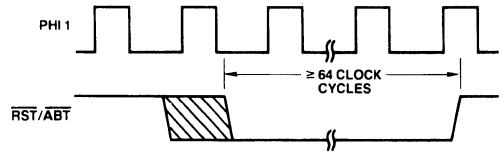
TL/EE/5491-14

FIGURE 3-3. Power-on Reset Requirements

3.0 Functional Description (Continued)

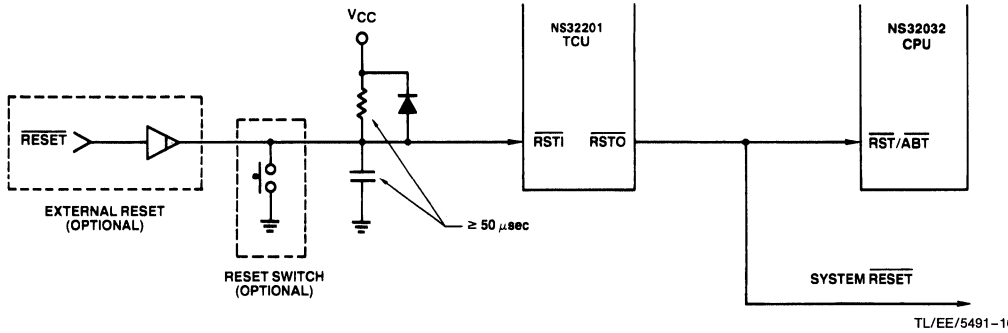
active for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4*.

The NS32201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32032 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



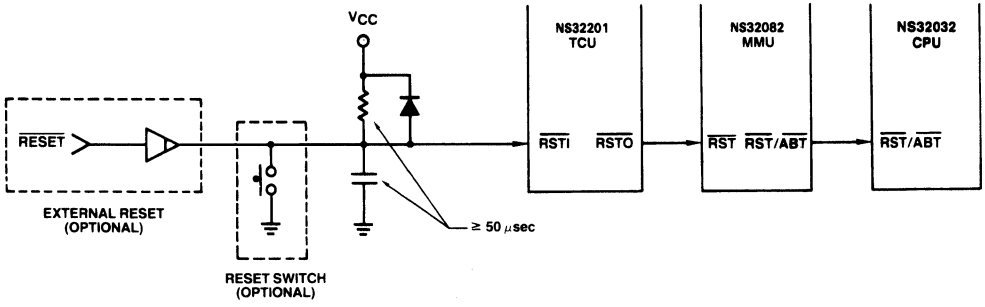
TL/EE/5491–15

FIGURE 3-4. General Reset Timing



TL/EE/5491–16

FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System



TL/EE/5491–17

FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

3.4 BUS CYCLES

The NS32032 CPU has a strap option which defines the Bus Timing Mode as either With or Without Address Translation. This section describes only bus cycles under the No Address Translation option. For details of the use of the strap and of bus cycles with address translation, see Sec. 3.5.

The CPU will perform a bus cycle for one of the following reasons:

1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
2) To fetch instructions into the eight-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4) To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Sec. 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Sec. 3.4.6).

The sequence of events in a non-Slave bus cycle is shown below in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).

3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

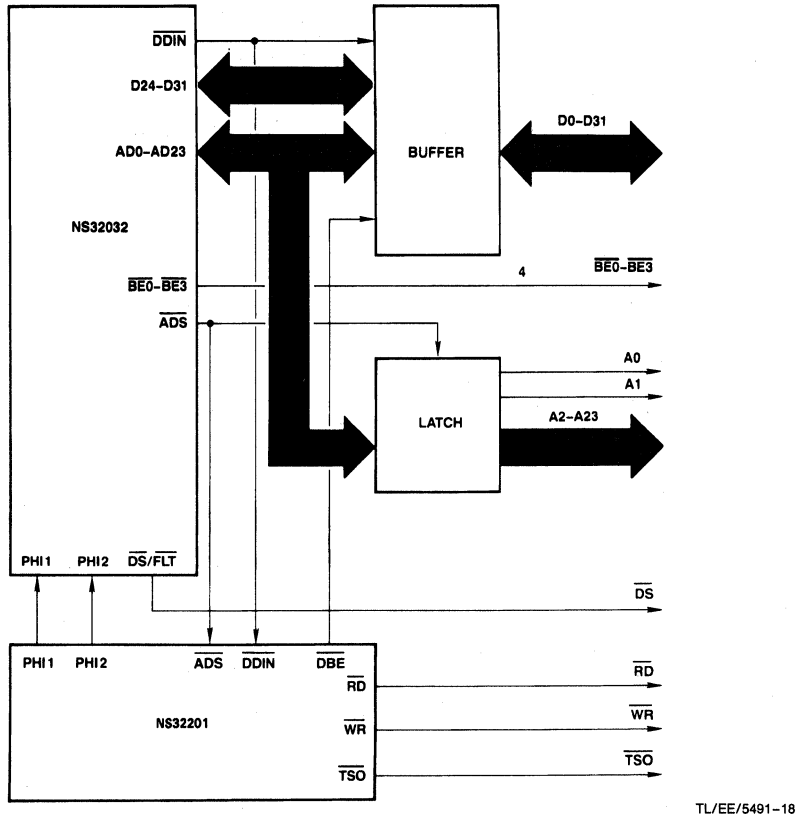
During T1, the CPU applies an address on pins AD0–AD23. It also provides a low-going pulse on the ADS pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–23 from the AD0–AD23 pins. See *Figure 3-6*. During this time also the status signals DDIN, indicating the direction of the transfer, and $\overline{BE0}$–$\overline{BE3}$, indicating which of the four bus bytes are to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD31 to either accept or present data. It also starts the data strobe ($\overline{DS}$), signalling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time: $\overline{RD}$ (Read Strobe) or $\overline{WR}$ (Write Strobe), $\overline{TSO}$ (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the end of T2 or T3, on the falling edge of the PHI2 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Sec. 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD31) is sampled at the falling edge of PHI2 of the last T3 state. See Section 4. Data must, however, be held at least until the beginning of T4. $\overline{DS}$ and $\overline{RD}$ are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

The T4 state finishes the bus cycle. At the beginning of T4, the $\overline{DS}$, $\overline{RD}$ or $\overline{WR}$, and $\overline{TSO}$ signals go inactive, and at the rising edge of PHI2, $\overline{DBE}$ goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).



TL/EE/5491–18

FIGURE 3-6. Bus Connections

3.0 Functional Description (Continued)

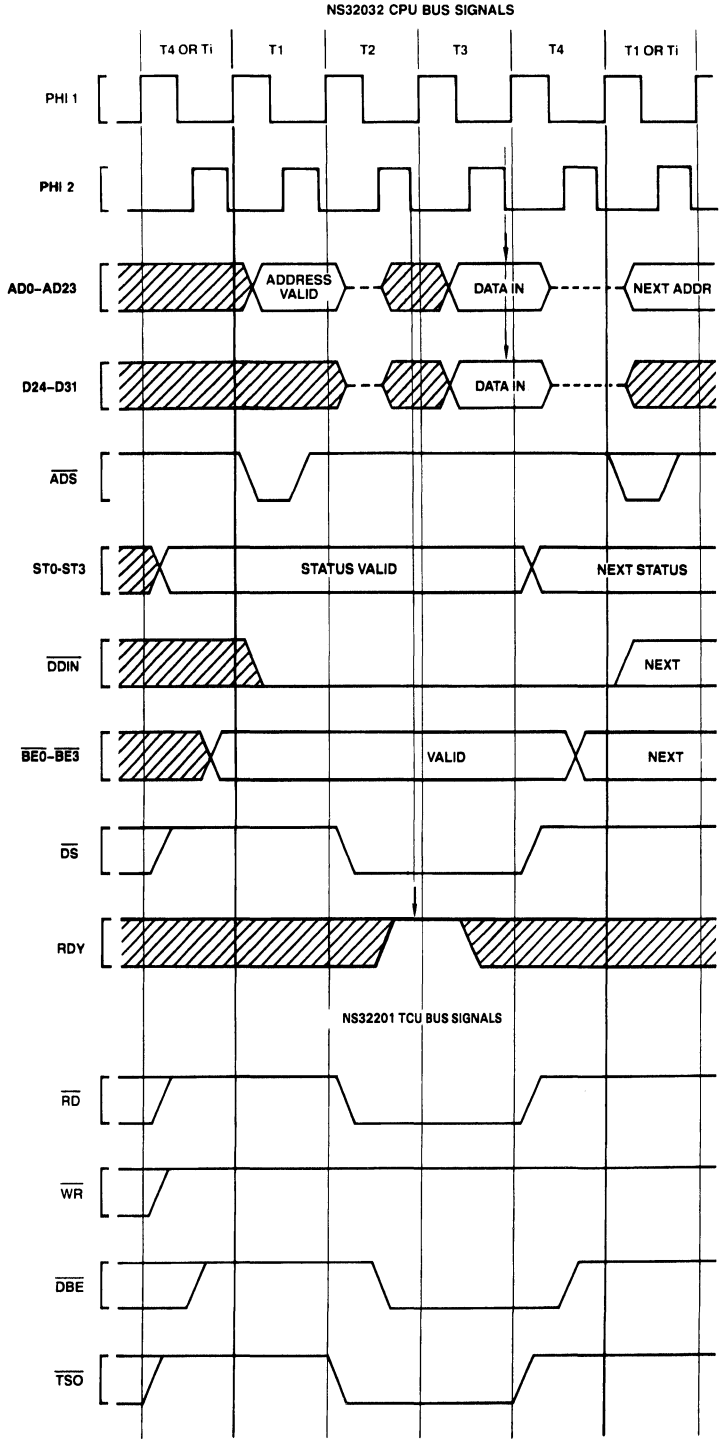
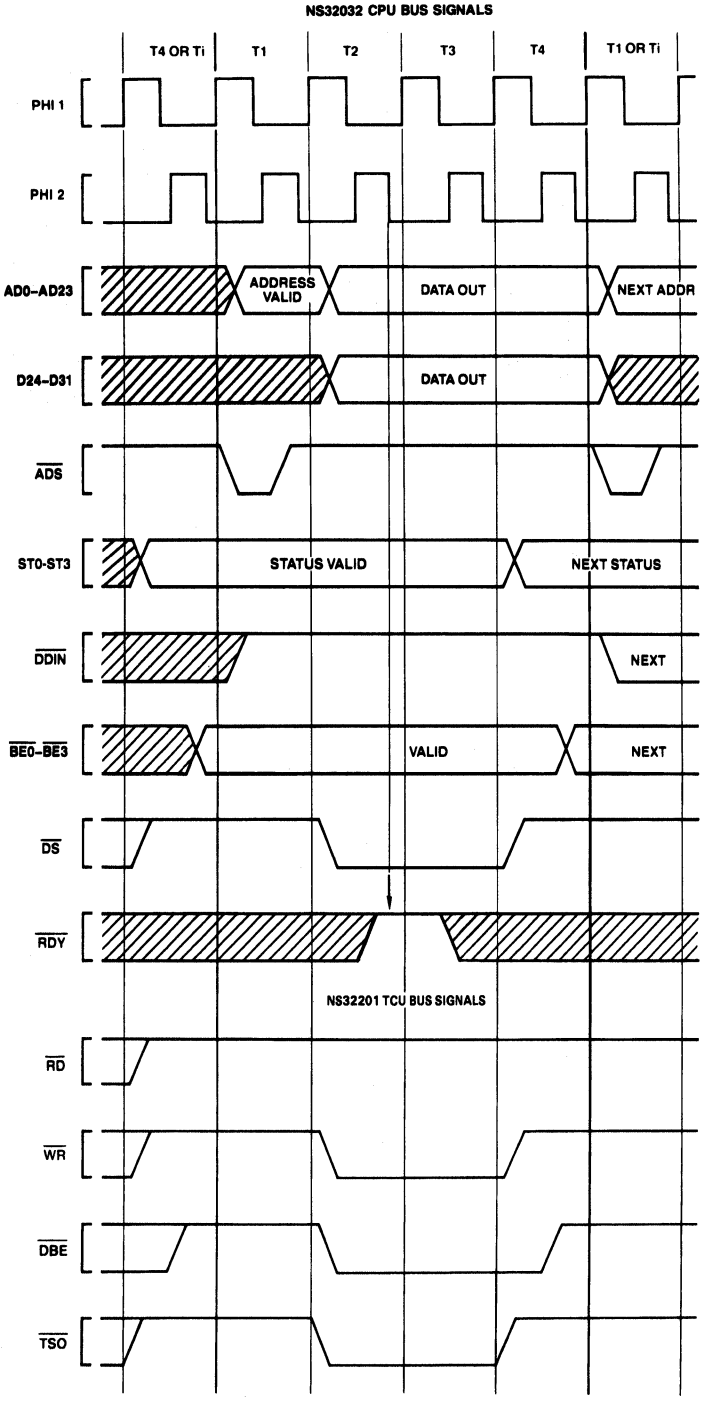


FIGURE 3-7. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/5491–19

FIGURE 3-8. Write Cycle Timing

3.0 Functional Description (Continued)

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32016 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "WAIT STATE". See *Figure 3-9*.

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pin:

1) $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT states until removed.
2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
3) $\overline{\text{PER}}$ (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU $\overline{\text{WAITn}}$ pins.

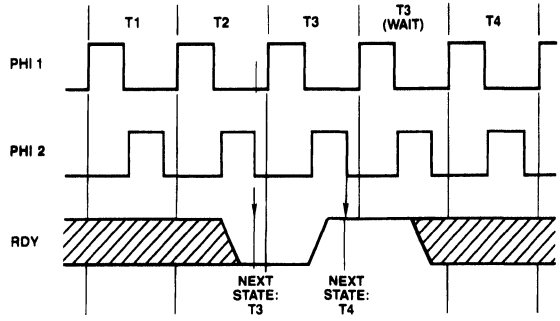


FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS32032 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why is it idle.

Referring to *Figures 3-7* and *3-8*, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before $\overline{\text{ADS}}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 – The bus is idle because the CPU does not need to perform a bus access.

0001 – The bus is idle because the CPU is executing the WAIT instruction.

0010 – (Reserved for future use.)

0011 – The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 – Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Sec. 3.4.5.

0110 – End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Sec. 3.4.5.

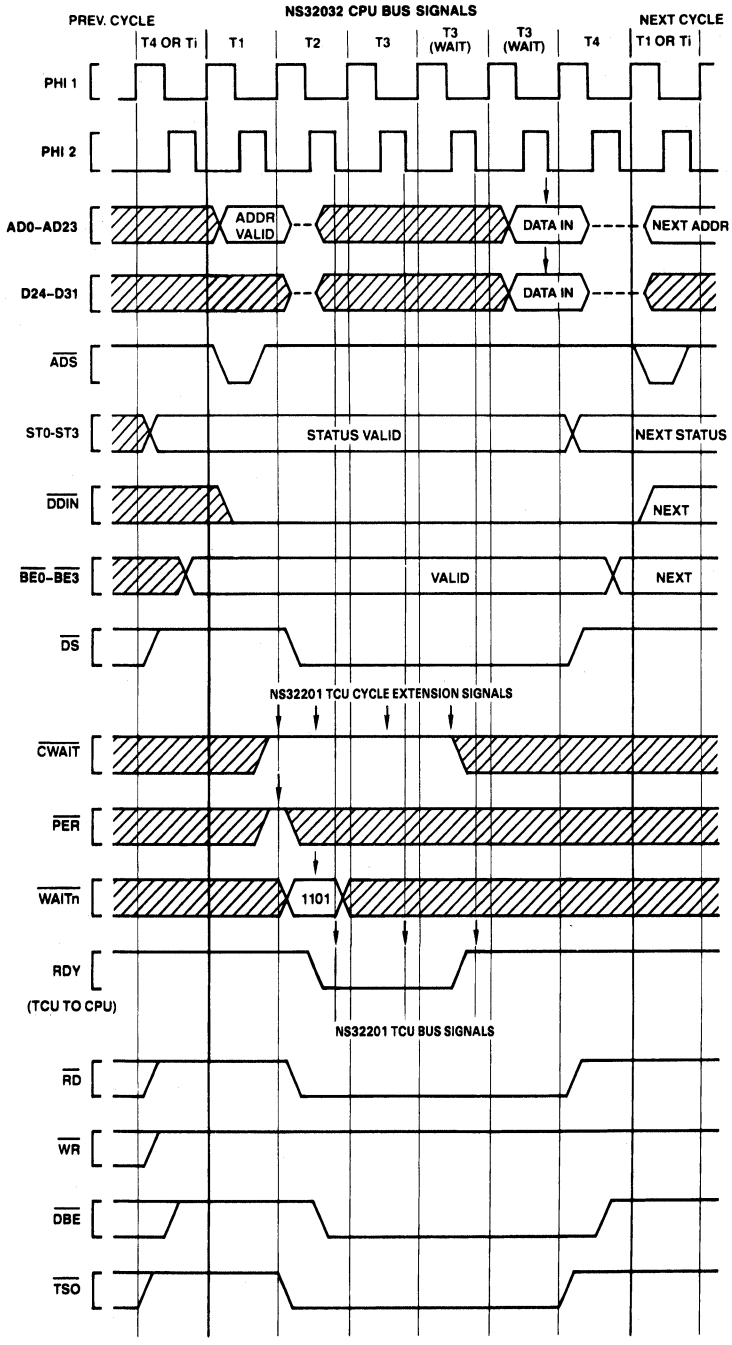
0111 – End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Sec. 3.4.5.

1000 – Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction

3.0 Functional Description (Continued)



TL/EE/5491–22

FIGURE 3-10. Extended Cycle Example

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAIT}}$n indicate points at which the TCU samples. Arrows on AD0–AD15 and RDY indicate points at which the CPU samples.

3.0 Functional Description (Continued)

Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.

The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

1110 – Read Slave Processor Status.

The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Sec. 3.9.1.

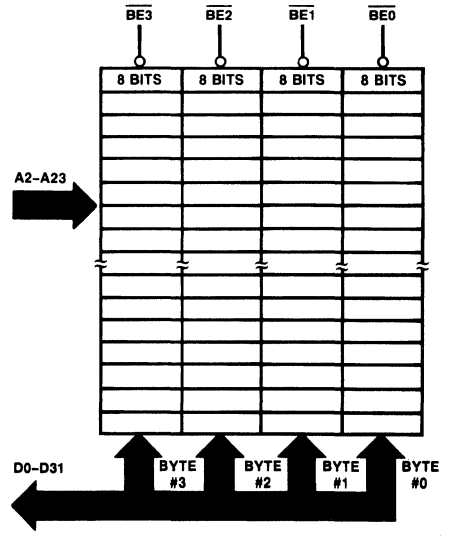
1111 – Broadcast Slave ID.

The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS32032 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32032 is that the presence of a 32-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32032 provides special control signals. Byte Enable ($\overline{BE0}$–$\overline{BE3}$) which facilitate individual byte accessing on a 32-bit bus.

Memory is organized as four eight-bit banks, each bank receiving the double-word address (A2–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7 is enabled when $\overline{BE0}$ is low. The second bank, connected to data bus pins AD8–AD15 is enabled when $\overline{BE1}$ is low. The third and fourth banks are enabled by $\overline{BE2}$ and $\overline{BE3}$, respectively. See *Figure 3-11*.



TL/EE/5491–23

FIGURE 3-11. Memory Interface

Since operands do not need to be aligned with respect to the double-word bus accessed performed by the CPU, a given double-word access can contain one, two, three, or four bytes of the operand being addressed, and these bytes can begin at various positions, as determined by A1, A0. Table 3-1 lists the 10 resulting access types.

TABLE 3-1
Bus Access Types

Type	Bytes Accessed	A1,A0	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$
1	1	00	1	1	1	0
2	1	01	1	1	0	1
3	1	10	1	0	1	1
4	1	11	0	1	1	1
5	2	00	1	1	0	0
6	2	01	1	0	0	1
7	2	10	0	0	1	1
8	3	00	1	0	0	0
9	3	01	0	0	0	1
10	4	00	0	0	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment. Table 3-2 lists the bus cycles performed for each situation.

3.0 Functional Description (Continued)

TABLE 3-2
Access Sequences

Cycle	Type	Address	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$	Data Bus: Byte 3	Byte 2	Byte 1	Byte 0
A. *Word at address ending with 11*									BYTE 1 | BYTE 0	← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	1	A + 1	1	1	1	0	X	X	X	Byte 1
B. *Double word at address ending with 01*							BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
C. *Double word at address ending with 10*							BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
D. *Double word at address ending with 11*							BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
E. *Quad word at address ending with 00*							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	10	A	0	0	0	0	Byte 3	Byte 2	Byte 1	Byte 0
Other bus cycles (instruction prefetch or slave) can occur here.										
2.	10	A + 4	0	0	0	0	Byte 7	Byte 6	Byte 5	Byte 4
F. *Quad word at address ending with 01*							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	9	A + 4	0	0	0	1	Byte 6	Byte 5	Byte 4	X
4.	1	A + 7	1	1	1	0	X	X	X	Byte 7
G. *Quad word at address ending with 10*							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	7	A + 4	0	0	1	1	Byte 5	Byte 4	X	X
4.	5	A + 6	1	1	0	0	X	X	Byte 7	Byte 6
H. *Quad word at address ending with 11*							BYTE 7 | BYTE 6 | BYTE 5 | BYTE 4 | BYTE 3 | BYTE 2 | BYTE 1 | BYTE 0			← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
Other bus cycles (instruction prefetch or slave) can occur here.										
1.	4	A + 4	0	1	1	1	Byte 4	X	X	X
2.	8	A + 5	1	0	0	0	X	Byte 7	Byte 6	Byte 5

X = Don't Care

3.0 Functional Description (Continued)

3.4.3.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.3.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

3.4.4 Instruction Fetches

Instructions for the NS32032 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the eight-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Sec. 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always type 10 Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle depends on the destination address.

Note: During non-sequential fetches, $\overline{BE0}$–$\overline{BE3}$ are all active regardless of the alignment.

3.4.5 Interrupt Control Cycles

Activating the $\overline{INT}$ or $\overline{NMI}$ pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status pesented on pins ST0–ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32032 interrupt structure, see Sec. 3.8.

3.0 Functional Description (Continued)

TABLE 3-3
Interrupt Sequences

								Data Bus			
Cycle	**Status**	**Address**	**$\overline{DDIN}$**	**$\overline{BE3}$**	**$\overline{BE2}$**	**$\overline{BE1}$**	**$\overline{BE0}$**	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**
				A. Non-Maskable Interrupt Control Sequences							
Interrupt Acknowledge											
1	0100	$FFFF00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
None: Performed through Return from Trap (RETT) instruction.											
				B. Non-Vectored Interrupt Control Sequences							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	X
				C. Vectored Interrupt Sequences: Non-Cascaded.							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Range: 0-127
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Same as in Previous Int. Ack. Cycle
				D. Vectored Interrupt Sequences: Cascaded							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: range -16 to -1
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0101	Cascade Address	0			See Note		Vector, range 9–255; on appropriate byte of data bus.			
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: Same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address)											
2	0111	Cascade Address	0			See Note		X	X	X	X

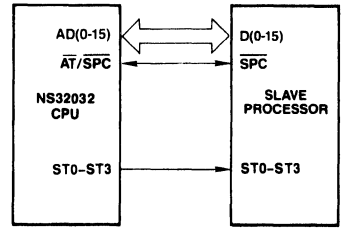
X = Don't Care

Note: $\overline{BE0}$-$\overline{BE3}$ signals will be activated according to the cascaded ICU address. The cycle type can be 1, 2, 3 or 4, when reading the interrupt vector. The vector value can be in the range 0–255.

3.0 Functional Description (Continued)

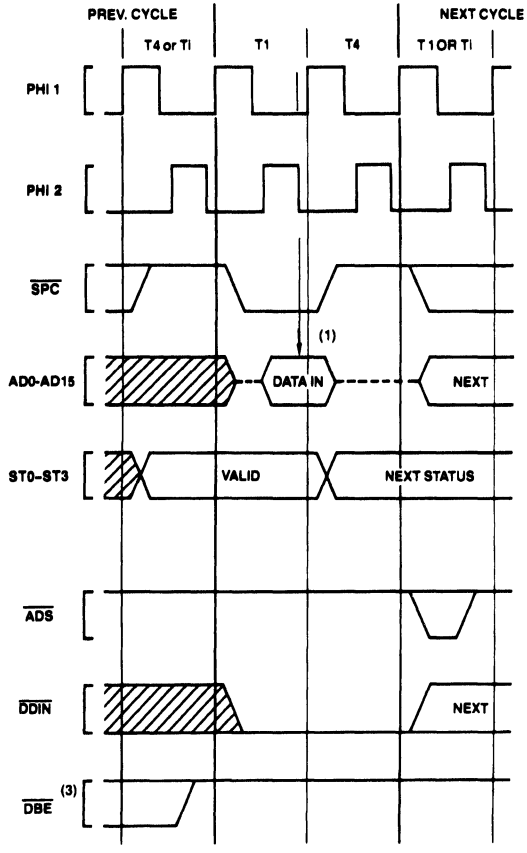
3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Sec. 3.5.1), the $\overline{AT}/\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{SPC}$). In a Slave Processor bus cycle, data is transferred on the Data Bus (AD0–AD15), and the status lines (ST0–ST3) are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.



TL/EE/5491–24

FIGURE 3-12. Slave Processor Connections



TL/EE/5491–25

Note:

(1) CPU samples Data Bus here.

(2) $\overline{DBE}$ and all other NS32201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor

3.0 Functional Description (Continued)

3.4.6.1 Slave Processor Bus Cycles

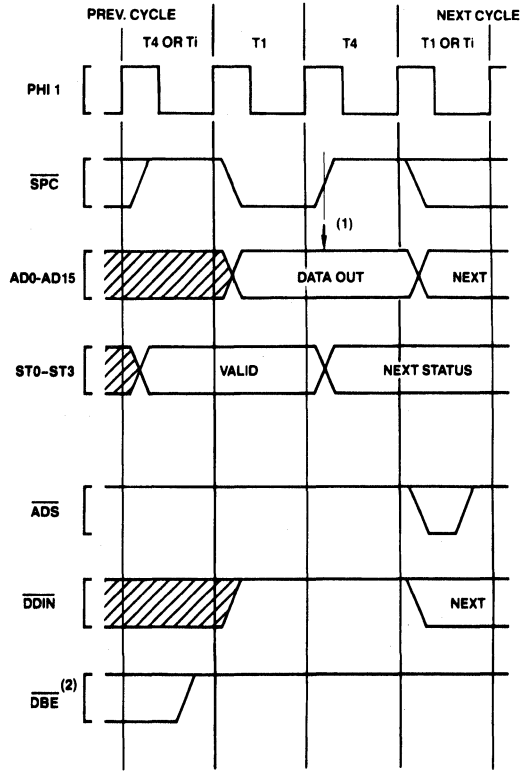
A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-13* and *3-14*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ($\overline{ADS}$), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.6.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0–AD7), and a Word operand is transferred on bits AD0–AD15. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.

Note that the NS32032 uses only the two least significant bytes of the data bus for slave cycles. This is to maintain compatibility with existing slave processors.



TL/EE/5491–26

Note:

(1) Slave Processor samples Data Bus here.

(2) $\overline{DBE}$, being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32032 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

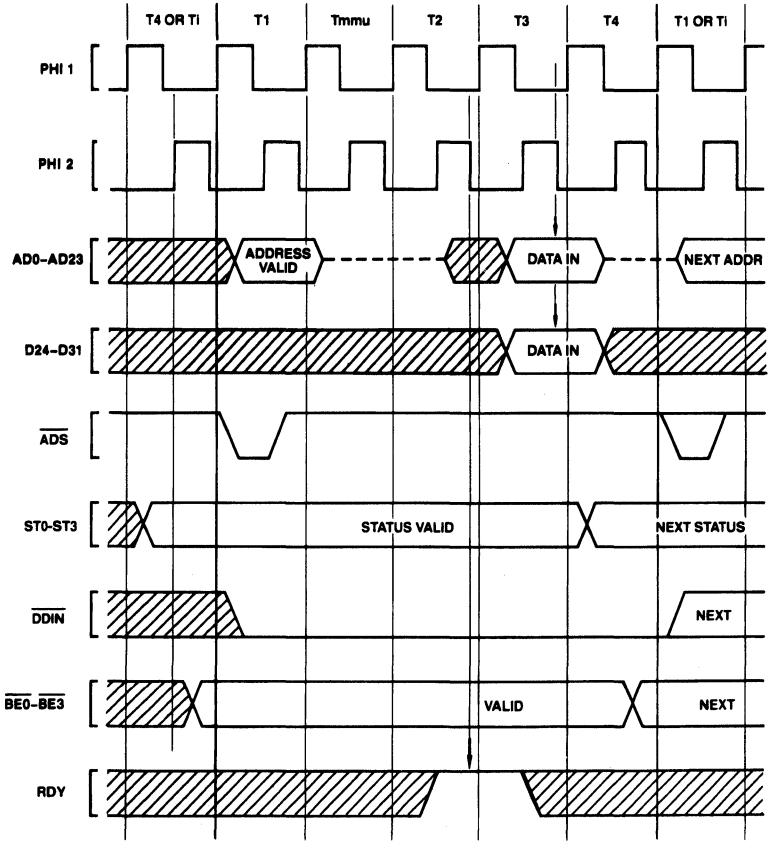
3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32032 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the $\overline{AT}/\overline{SPC}$ (Address Translation/Slave Processor Control) pin on the rising edge of the $\overline{RST}$ (Reset) pulse. If $\overline{AT}/\overline{SPC}$ is sampled as high, the bus timing is as previously described in Sec. 3.4. If it is sampled as low, two changes occur:

1) An extra clock cycle, Tmmu, is inserted into all bus cycles except Slave Processor transfers.
2) The $\overline{DS}/\overline{FLT}$ pin changes in function from a Data Strobe output ($\overline{DS}$) to a Float Command input ($\overline{FLT}$).

The NS32082 MMU will itself pull the CPU $\overline{AT}/\overline{SPC}$ pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the



TL/EE/5491-27

FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

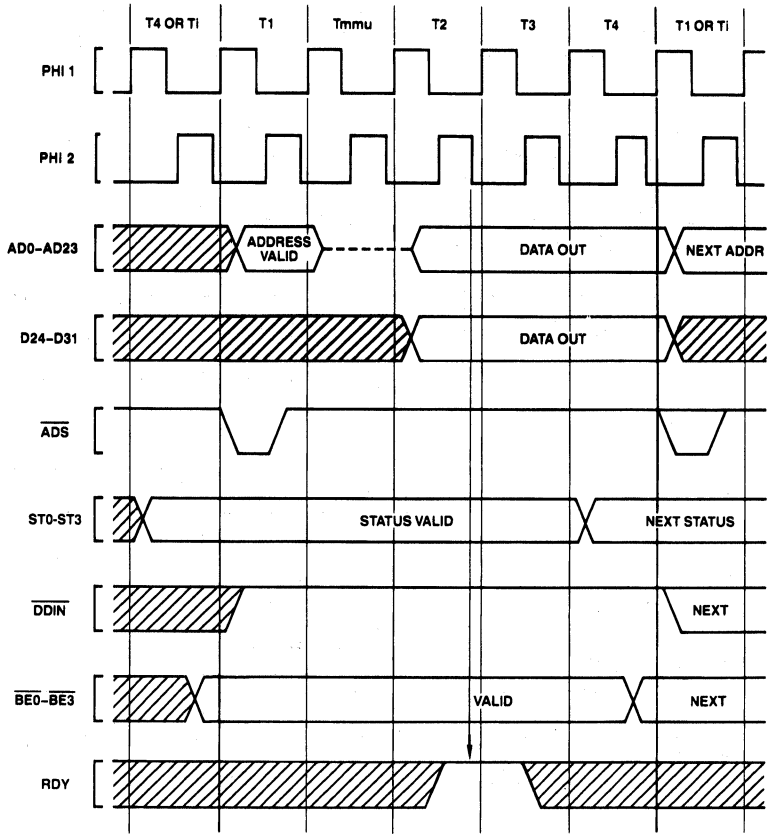
3.0 Functional Description (Continued)

presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Sec. 2.1.3.

3.5.2 Translated Bus Timing

Figures 3-15 and *3-16* illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0–AD23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{PAV}$. T2 through T4 of the cycle are identical to their counterparts without Address Translation. Note that in order for the NS32082 MMU to operate correctly it must be set to the 32032 mode by forcing A24/$\overline{HBF}$ low during reset. In this mode the bus lines AD16–AD23 are floated after the MMU address has been latched, since they are used by the CPU to transfer data.

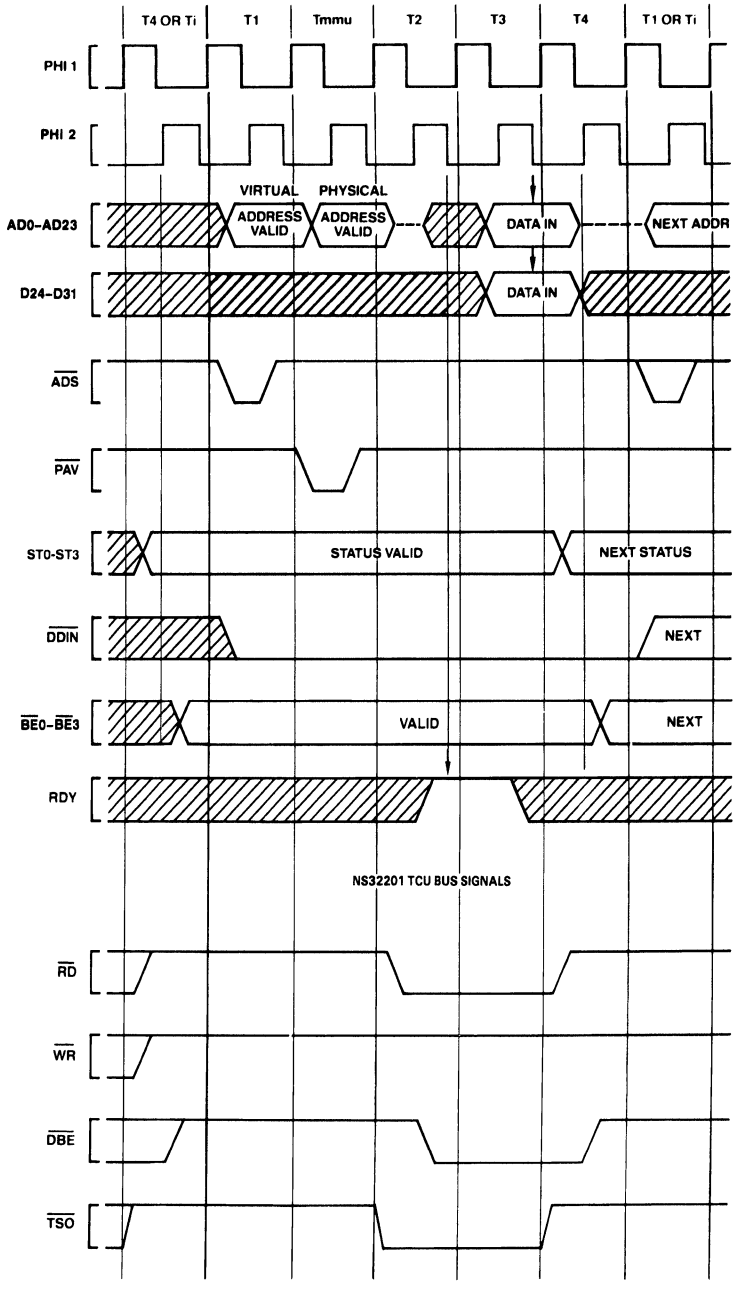
Figures 3-17 and *3-18* show a Read cycle and a Write cycle as generated by the 32032/32082/32201 group. Note that with the CPU $\overline{ADS}$ signal going only to the MMU, and with the MMU $\overline{PAV}$ signal substituting for $\overline{ADS}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.



TL/EE/5491–28

FIGURE 3-16. Write Cycle with Address Translation (CPU Action)

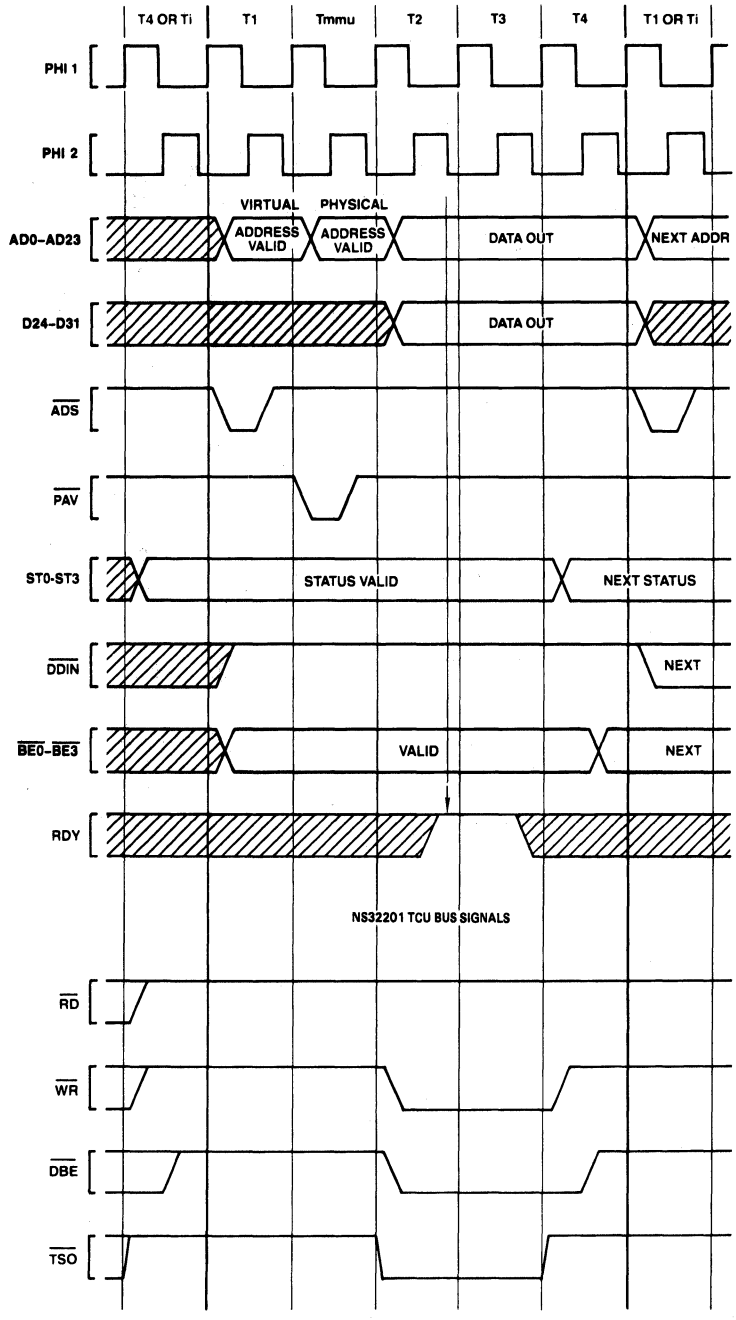
3.0 Functional Description (Continued)



TL/EE/5491-29

FIGURE 3-17. Memory-Managed Read Cycle

3.0 Functional Description (Continued)



TL/EE/5491–30

FIGURE 3-18. Memory-Managed Write Cycle

3.0 Functional Description (Continued)

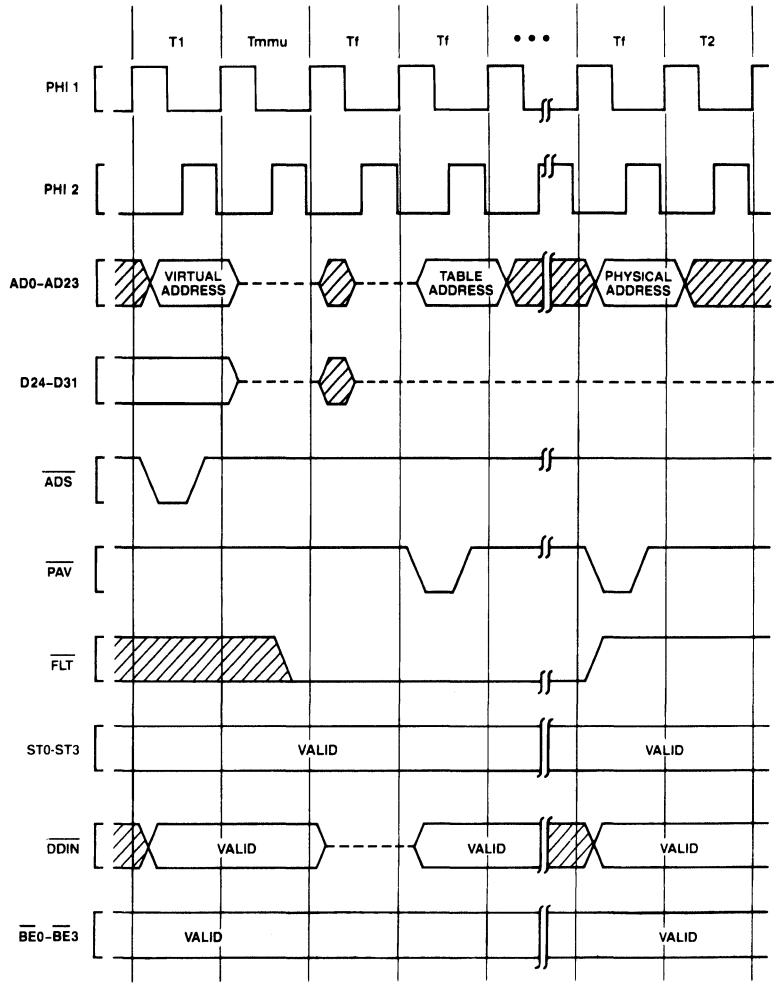
3.5.3 The $\overline{FLT}$ (Float) Pin

The $\overline{FLT}$ pin is used by the CPU for address translation support. Activating $\overline{FLT}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its translation look-aside buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effect of $\overline{FLT}$. Upon sampling $\overline{FLT}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD23, D24–D31 and $\overline{DDIN}$ to the TRI-STATE condition ("floating").
2) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{RST}/\overline{ABT}$ description, Sec. 3.5.4.)

Note that the AD0–AD23 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until $\overline{FLT}$ again goes high. See the Timing Specifications, Sec. 4.



TL/EE/5491–31

FIGURE 3-19. $\overline{FLT}$ Timing

3.0 Functional Description (Continued)

3.5.4 Aborting Bus Cycles

The $\overline{RST}/\overline{ABT}$ pin, apart from its Reset function (Sec. 3.3), also serves as the means to "abort", or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the $\overline{RST}/\overline{ABT}$ pin is held active for only one clock cycle.

If $\overline{RST}/\overline{ABT}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then Ti, thereby terminating the cycle. Since it is the MMU $\overline{PAV}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.
2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

3.5.4.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the NS32082 Memory Management Unit.

1) If $\overline{FLT}$ has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, *Figure 4-22*.
2) If $\overline{FLT}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{FLT}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{FLT}$ is removed. See *Figure 4-23*.
3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If $\overline{RST}/\overline{ABT}$ is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

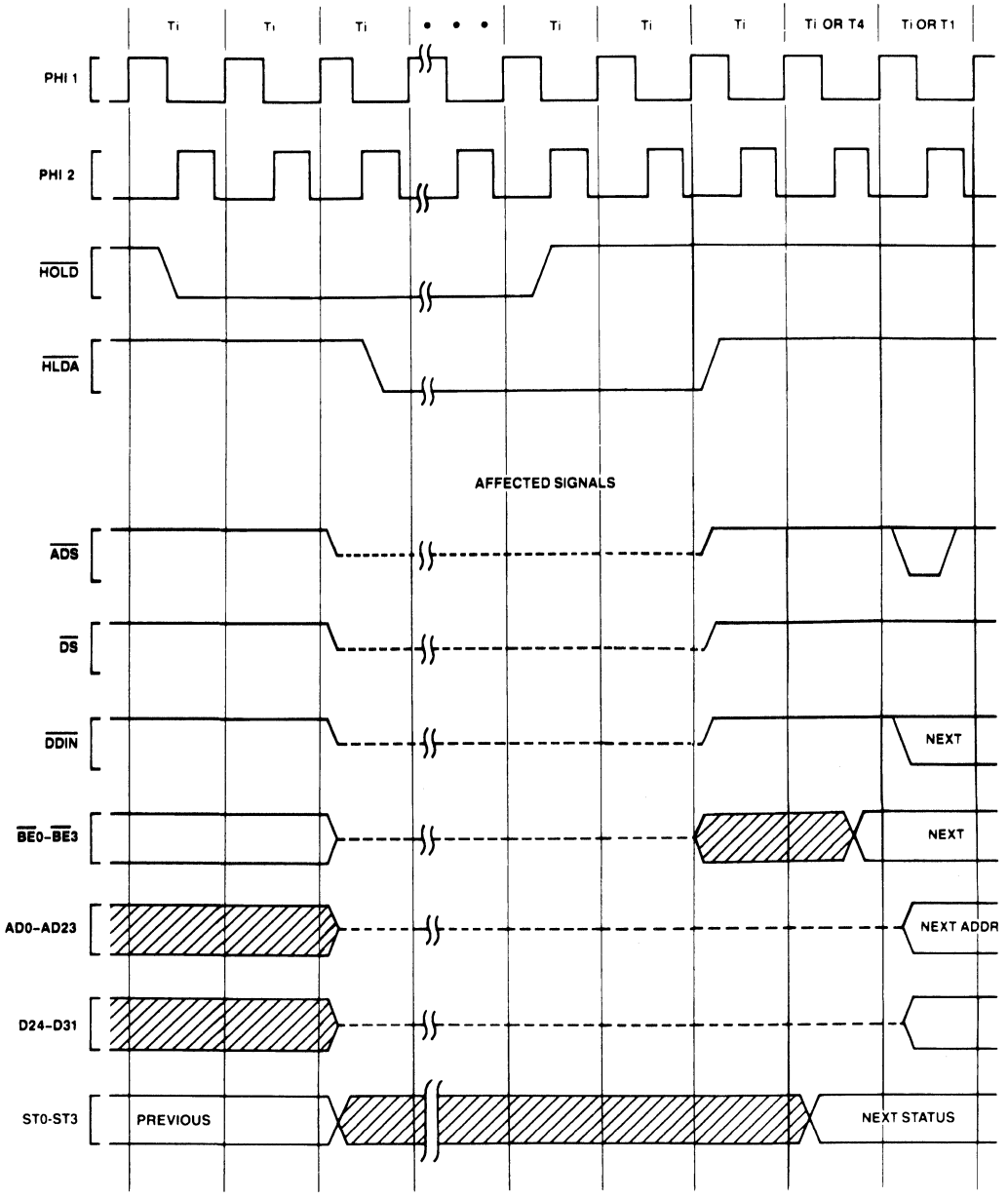
3.6 BUS ACCESS CONTROL

The NS32032 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{HOLD}$ (Hold Request) and $\overline{HLDA}$ (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of $\overline{HLDA}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD23, D24–D31, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{BE0}$–$\overline{BE3}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{HOLD}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{HLDA}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{HOLD}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-20* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-21* shows the sequence if the CPU is using the bus at the time that the $\overline{HOLD}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{HLDA}$ signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.

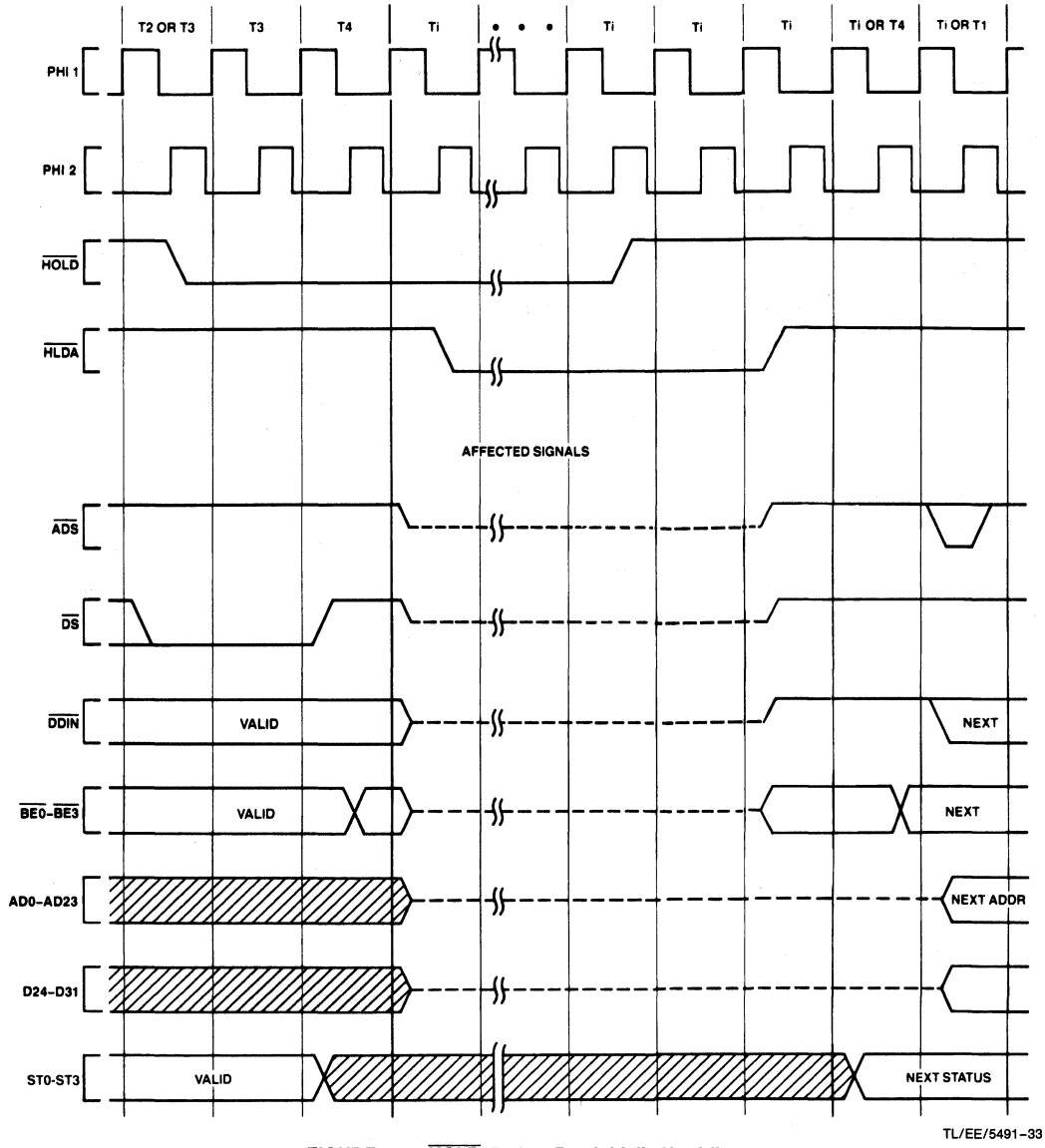
3.0 Functional Description (Continued)



TL/EE/5491–32

FIGURE 3-20. HOLD Timing, Bus Initially Idle

3.0 Functional Description (Continued)



TL/EE/5491–33

FIGURE 3-21. HOLD Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32032 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/$\overline{S}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection, and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-21*.

$\overline{ILO}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing. As with the U/$\overline{S}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, *Figure 4-19*.

3.8 NS32032 INTERRUPT STRUCTURE

$\overline{INT}$, on which maskable interrupts may be requested,

$\overline{NMI}$, on which non-maskable interrupts may be requested, and

$\overline{RST}$/$\overline{ABT}$, which may be used to abort a bus cycle and any associated instruction. See Sec. 3.5.4.

In addition there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

1) Adjustment of Registers.

   Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2) Saving Processor Status.

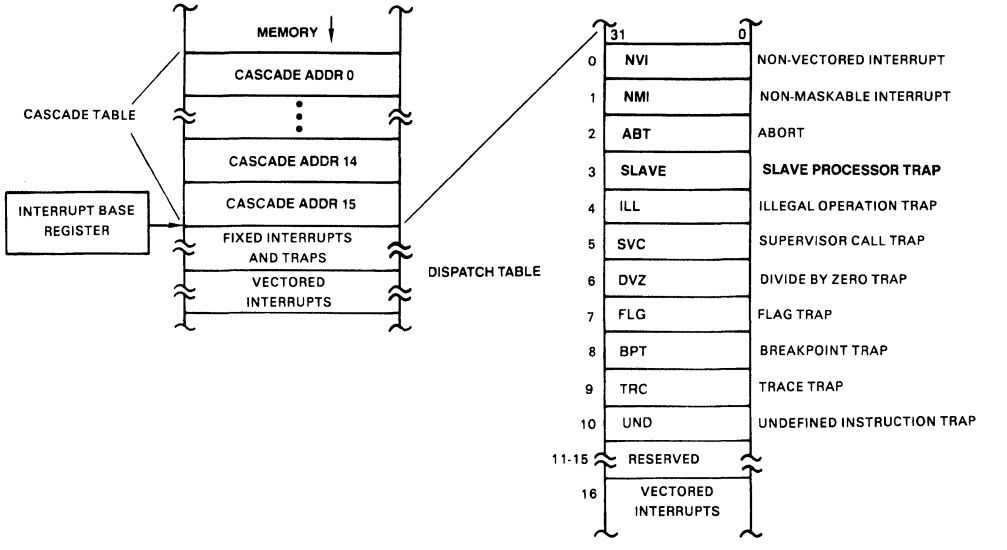
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

3) Vector Acquisition.

   A Vector is either obtained from the Data Bus or is supplied by default.

4) Service Call.

   The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-22*. A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.



TL/EE/5491–34

FIGURE 3-22. Interrupt Dispatch and Cascade Tables

3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-23*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Sec. 3.8.7.1.
Abort Interrupt:	Sec. 3.8.7.4.
Traps (except Trace):	Sec. 3.8.7.2.
Trace Trap:	Sec. 3.8.7.3.

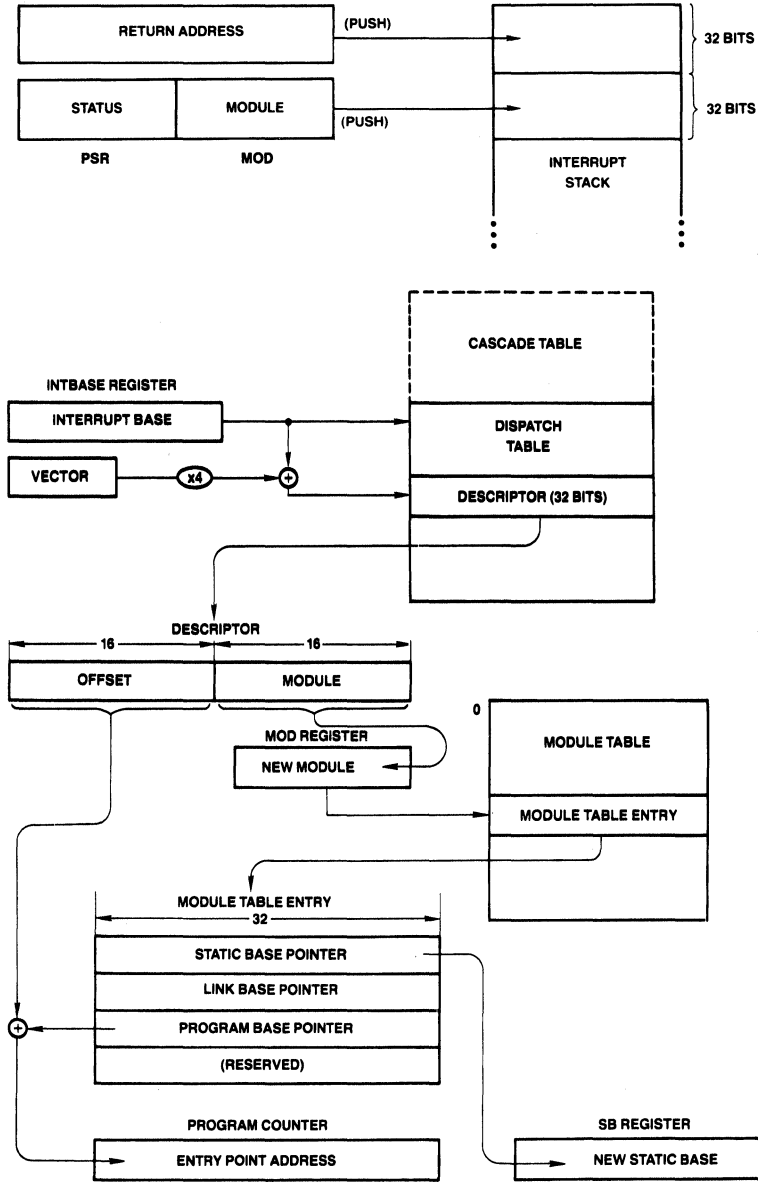


FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction (*Figure 3-24*) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-25*.

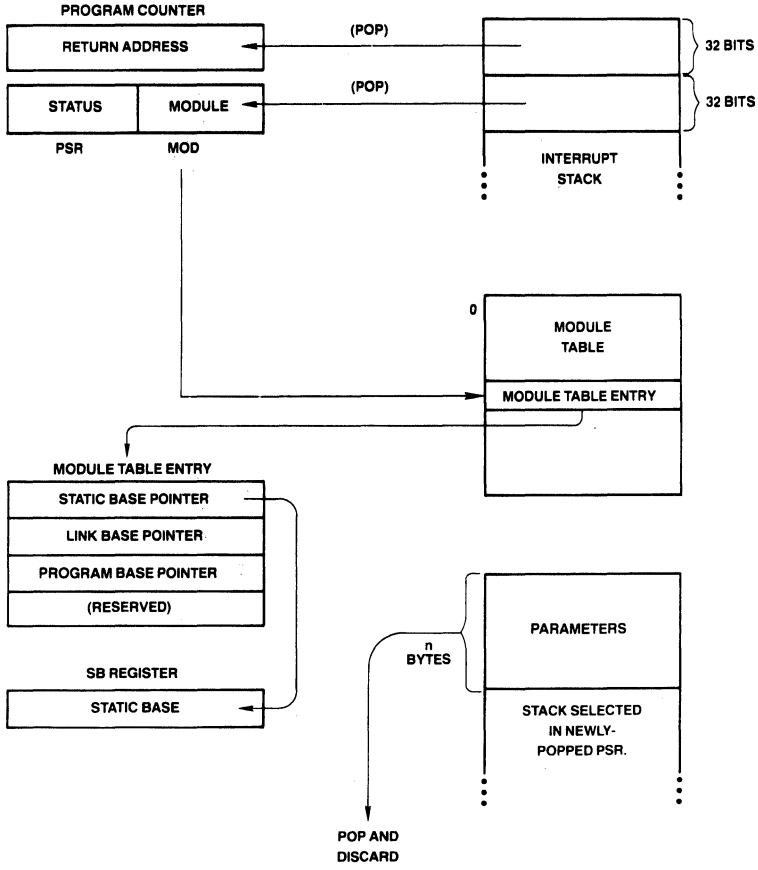
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$, $\overline{\text{NMI}}$ or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = C) or Vectored (bit I = 1).

3.8.3.1 Non-Vectored Mode

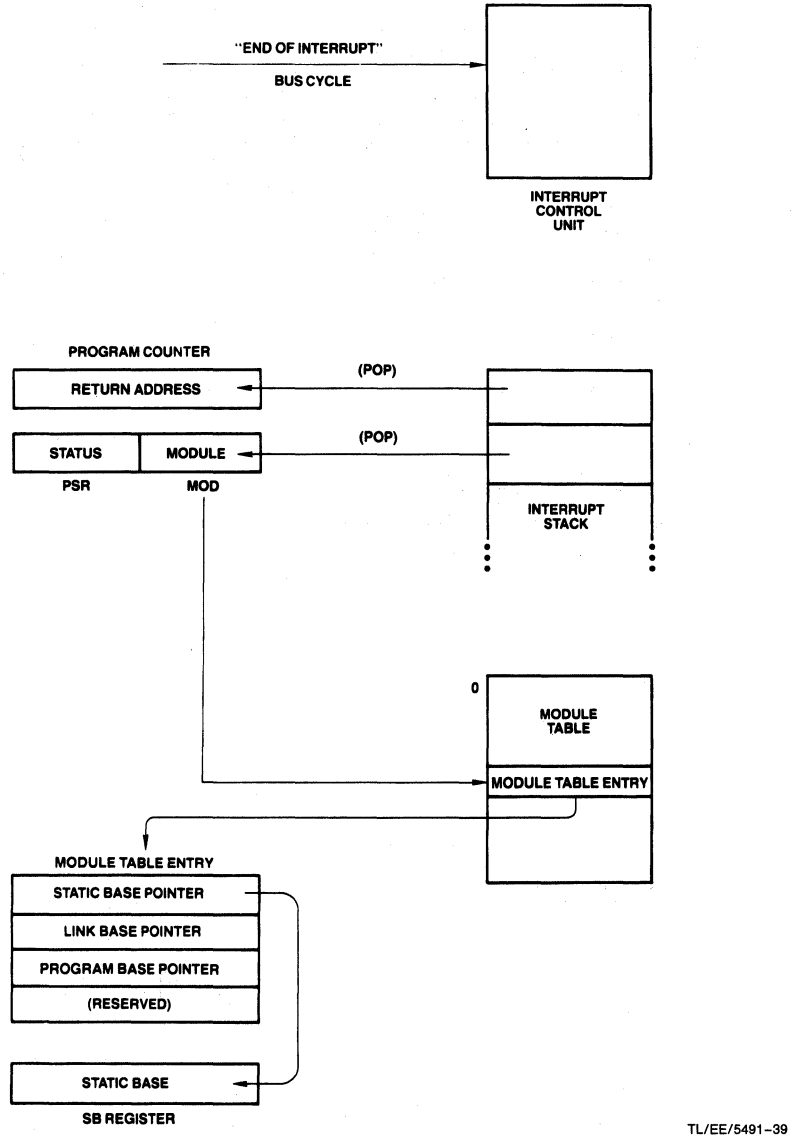
In the Non-Vectored mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.



TL/EE/5491–37

FIGURE 3-24. Return from Trap (RETT n) Instruction Flow

3.0 Functional Description (Continued)



TL/EE/5491–39

FIGURE 3-25. Return from Interrupt (RETI) Instruction Flow

3.0 Functional Description (Continued)

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.8.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-27*, shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{INT}$ pin.

In a system which uses cascading, two tasks must be performed upon initialization:

1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.

2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

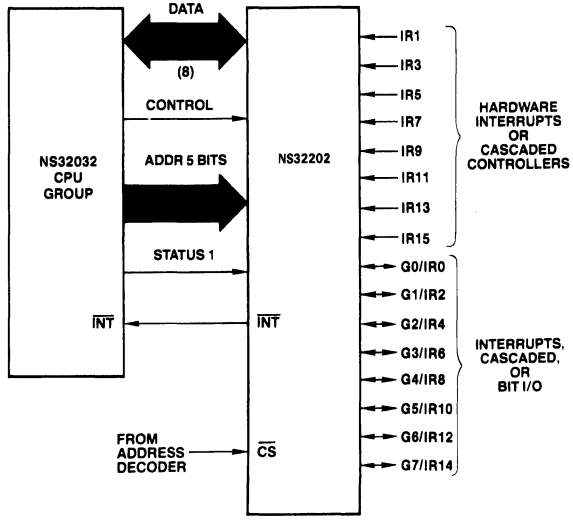
Figure 3-22 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range −16 to −1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Sec. 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Sec. 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the Interrupt Mask Register of the Interrupt Controller.

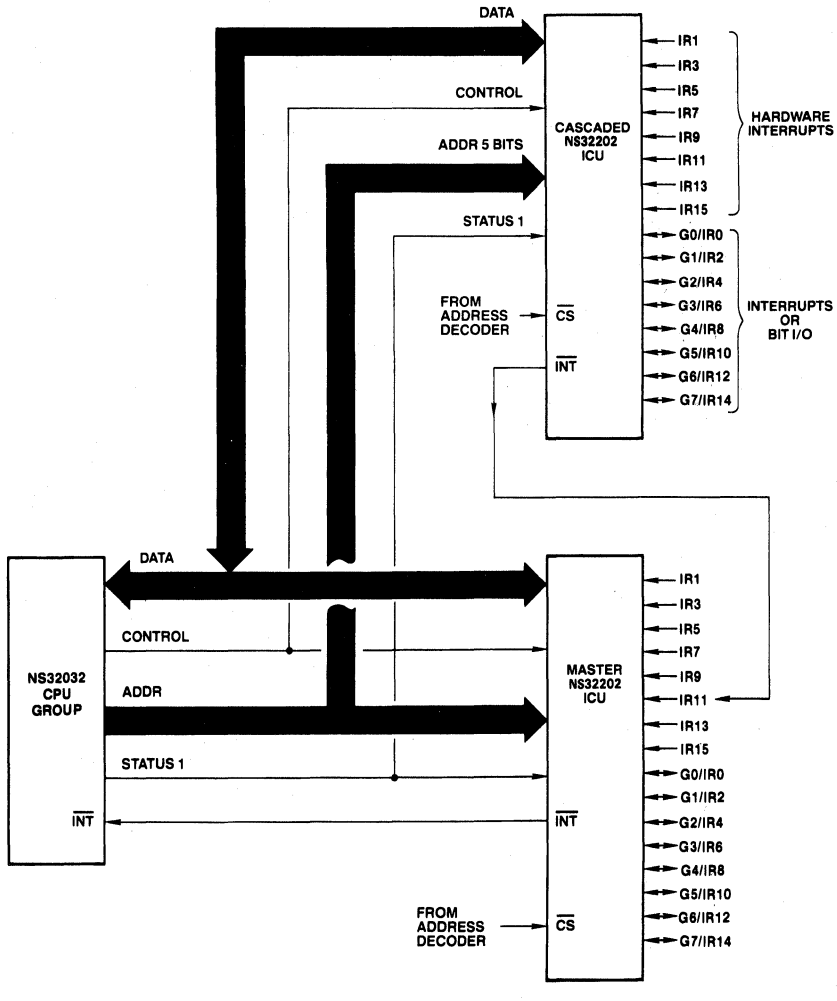
However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.



TL/EE/5491-40

FIGURE 3-26. Interrupt Control Unit Connections (16 Levels)

3.0 Functional Description (Continued)



TL/EE/5491–41

FIGURE 3-27. Cascaded Interrupt Control Unit Connections

3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the $\overline{\text{NMI}}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is FFFF00_{16}. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Sec. 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32032 CPU are:

Trap (SLAVE): An exceptional condition was detected by the Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).

3.0 Functional Description (Continued)

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

3.8.6 Prioritization

The NS32016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1) Traps other than Trace (Highest priority)
2) Abort
3) Non-Maskable Interrupt
4) Maskable Interrupts
5) Trace Trap (Lowest priority)

3.8.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-28*. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{INT}$ or $\overline{NMI}$ pins, respectively), see Sec. 3.8.7.1 For Abort Interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{NMI}$ pin receives a falling edge, or the $\overline{INT}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master, Sec. 3.4.2). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.2). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.2).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE $+4*$ Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Sec. 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-28*.

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector* 4 + INTBASE Register contents.

2) Move the Module field of the Descriptor into the MOD Register.

3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.

4) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.

5) Flush queue: Non-sequentially fetch first instruction of Interrupt routine.

6) Push MOD Register into the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)

7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-28. Service Sequence

Invoked during all interrupt/trap sequences.

3.0 Functional Description (Continued)

3.8.7.2 Trap Sequence: Traps Other Than Trace

1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2) Set "Vector" to the value corresponding to the trap type.
   SLAVE: Vector = 3.
   ILL: Vector = 4.
   SVC: Vector = 5.
   DVZ: Vector = 6.
   FLG: Vector = 7.
   BPT: Vector = 8.
   UND: Vector = 10.
3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Return Address" to the address of the first byte of the trapped instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

3.8.7.3 Trace Trap Sequence

1) In the Processor Status Register (PSR), clear the P bit.
2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4) Set "Vector" to 9.
5) Set "Return Address" to the address of the next instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

3.8.7.4 Abort Sequence

1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
2) Clear the PSR P bit.
3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Vector" to 2.
6) Set "Return Address" to the address of the first byte of the aborted instruction.
7) Perform Service (Vector, Return Address), *Figure 3-28.*

3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32032 CPU recognizes three groups of instructions being executable by external Slave Processor:

Floating Point Instruction Set
Memory Management Instruction Set
Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

3.9.1 Slave Processor Protocol

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.
2) It specifies which Slave Processor will execute it.
3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-29.* While applying Status Code 1111 (Broadcast ID, Sec. 3.4.2), the CPU transfers the ID Byte on the least-significant byte of the Data Bus (AD0–AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The operation Word is swapped on the Data Bus, that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Address Mode fields within the Operation Word, the CPU starts fetching operand and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible

Status Combinations:
Send ID (ID): Code 1111
Xfer Operand (OP): Code 1101
Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operaton Word.
3	OP	CPY Sends Required Operands
4	—	Slave Starts Execution. CPU Pre-fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-29. Slave Processor Protocol

3.0 Functional Description (Continued)

for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.2).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this, and for the Address Translation strap function, $\overline{AT}/\overline{SPC}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Sec. 3.4.2).

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.2). This word has the format shown in *Figure 3-30*. If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the Slave vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.9.2 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

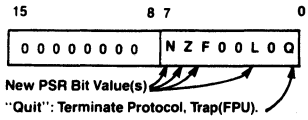
The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-30*).

TABLE 3-4
Floating Point Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note:
D = Double Word
i = Integer size (B,W,D) specified in mnemonic.
f = Floating Point type (F,L) specified in mnemonic.
N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)



TL/EE/5491–42

FIGURE 3-30. Slave Processor Status Word Format

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

3.9.3 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Instruction Set Reference Manual and the NS32082 MMU Data Sheet.

TABLE 3-5
Memory Management Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Instruction Set Reference Manual and the NS32082 Memory Management Unit Data Sheet.

D = Double Word

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)

3.9.4 Custom Slave Instructions

Provided in the NS32032 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

TABLE 3-6

Custom Slave Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op.2	none
CCMP0c	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to OP. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

Note:

D = Double Word

i = Integer size (B,W,D) specified in mnemonic.

c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.

* = Privileged instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 NS32032 PIN DESCRIPTIONS

The following is a brief description of all NS32032 pins. The descriptions reference portions of the Functional Description. Sec. 3.

Unless otherwise indicated (see pin 34) reserved pins should be left open.

4.1.1 Supplies

Power (V_{CC}): +5V Positive Supply. Sec. 3.1.

Logic Ground (GNDL): Ground reference for on-chip logic. Sec. 3.1.

Buffer Grounds #1 (GNDB1, GNDB2, GNDB3): Ground references for the on-chip output drivers connected to output pins. Sec. 3.1.

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Sec. 3.1.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Sec. 3.2.

Ready (RDY): Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Sec. 3.4.1.

Hold Request ($\overline{HOLD}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Sec. 3.6.

Note 1: $\overline{HOLD}$ must not be asserted until $\overline{HLDA}$ from a previous $\overline{HOLD}$/$\overline{HLDA}$ sequence is deasserted.

Note 2: If the $\overline{HOLD}$ signal is generated asynchronously, it's set up and hold times may be violated.

In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{HLDA}$ latency. This is to avoid speed degradations in cases of heavy $\overline{HOLD}$ activity (i.e., DMA controller cycles interleaved with CPU cycles.)

Interrupt ($\overline{INT}$): Active low. Maskable Interrupt request. Sec. 3.8.

Non-Maskable Interrupt ($\overline{NMI}$): Active low. Non-Maskable Interrupt request. Sec. 3.8.

Reset/Abort ($\overline{RST}$/$\overline{ABT}$): Active low. If held active for one clock cycle and released, this pin causes an Abort Command, Sec. 3.5.4. If held longer, it initiates a Reset. Sec. 3.3.

4.1.3 Output Signals

Address Strobe ($\overline{ADS}$): Active low. Controls address latches: indicates start of a bus cycle. Sec. 3.4.

Data Direction in ($\overline{DDIN}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Sec. 3.4.

Byte Enable ($\overline{BE0}$–$\overline{BE3}$): Active low. Four control signals enabling data transfers on individual bus bytes. Sec. 3.4.3.

Status (ST0–ST3): Bus cycle status code, ST0 least significant. Sec. 3.4.2. Encodings are:

0000 — Idle: CPU Inactive on Bus.
0001 — Idle: WAIT Instruction.
0010 — (Reserved).
0011 — Idle: Waiting for Slave.
0100 — Interrupt Acknowledge, Master.
0101 — Interrupt Acknowledge, Cascaded.
0110 — End of Interrupt, Master.
0111 — End of Interrupt, Cascaded.
1000 — Sequential Instruction Fetch.
1001 — Non-Sequential Instruction Fetch.
1010 — Data Transfer.
1011 — Read Read-Modify-Write Operand.
1100 — Read for Effective Address.
1101 — Transfer Slave Operand.
1110 — Read Slave Status Word.
1111 — Broadcast Slave ID.

Hold Acknowledge ($\overline{HLDA}$): Active low. Applied by the CPU in response to $\overline{HOLD}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Sec. 3.6.

User/Supervisor (U/$\overline{S}$): User or Supervisor Mode status. Sec. 3.7. High state indicates User Mode, low indicates Supervisor Mode. Sec. 3.7.

Interlocked Operation ($\overline{ILO}$): Active low. Indicates that an interlocked instruction is being executed. Sec. 3.7.

Program Flow Status ($\overline{PFS}$): Active low. Pulse indicates beginning of an instruction execution. Sec. 3.7.

4.1.4 Input-Output Signals

Address/Data 0–23 (AD0–AD23): Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Sec. 3.4.

Data Bits 24–31 (D24–D31): The high order 8 bits of the data bus.

Address Translation/Slave Processor Control ($\overline{AT}$/$\overline{SPC}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of a slave instruction. Sec. 3.4.6; Sec. 3.9. Sampled on the rising edge of Reset pulse as Address Translation Strap. Sec. 3.5.1.

In non-memory-managed systems, this pin should be pulled-up to V_{CC} through a 10 kΩ resistor.

Data Strobe/Float ($\overline{DS}$/$\overline{FLT}$): Active low. Data Strobe output, Sec. 3.4, or Float Command input, Sec. 3.5.3. Pin function is selected on $\overline{AT}$/$\overline{SPC}$ pin, Sec. 3.5.1.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

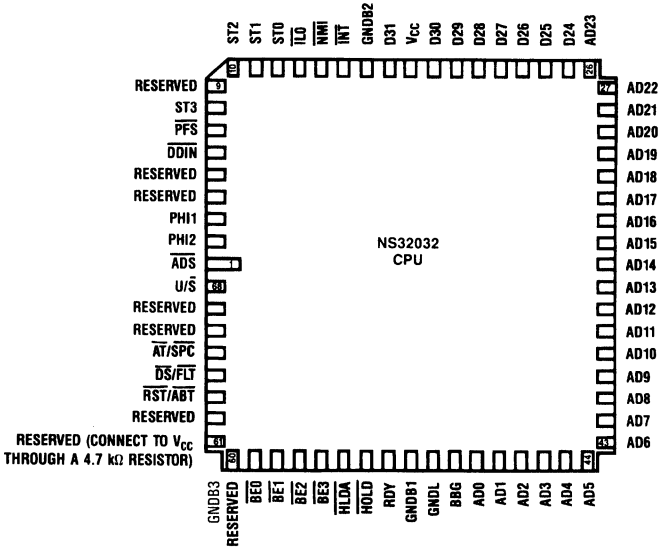
Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages With Respect to GND	−0.5V to +7V
Power Dissipation	1.5 Watt

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS $T_A = 0°$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC}+0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC}-0.35$		$V_{CC}+0.5$	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage. Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\ \mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2$ mA			0.45	V
I_{ILS}	$\overline{AT}/\overline{SPC}$ Input Current (low)	$V_{IN} = 0.4V$, $\overline{AT}/\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current (Output pins in TRI-STATE condition.)	$0.4 \le V_{OUT} \le V_{CC}$	−20		30	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		180	300	mA



TL/EE/5491–2

Bottom View

FIGURE 4-1. NS32032 Connection Diagram

Order Number NS32032E-6, NS32032E-8, NS32032E-10, NS32032V-6, NS32032V-8 or NS32032V-10
See NS Package E68B or V68A

4.0 Device Specifications (Continued)

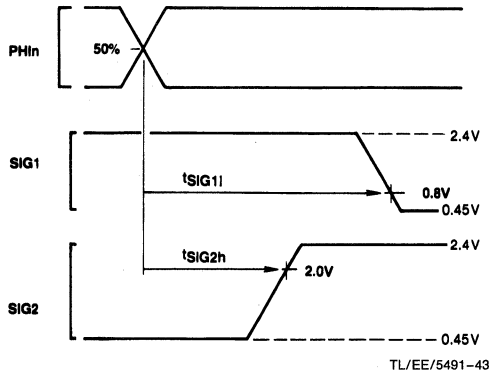
4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 and PHI2 and 0.8V or 2.0V on all other signals as illustrated in *Figures 4-2* and *4-3*, unless specifically stated otherwise.

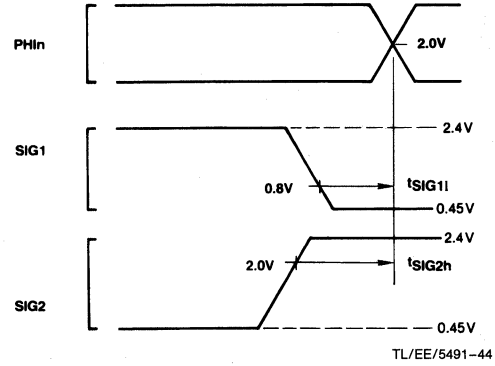
ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge
T.E. — trailing edge F.E. — falling edge



TL/EE/5491–43

FIGURE 4-2. Timing Specification Standard (Signal Valid After Clock Edge)



TL/EE/5491–44

FIGURE 4-3. Timing Specification Standard (Signal Valid Before Clock Edge)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4-4	Address bits 0–23 valid	after R.E., PHI1 T1		65		55		40	ns
t_{ALh}	4-4	Address bits 0–23 hold	after R.E., PHI1 Tmmu or T2	5		5		5		ns
t_{Dv}	4-4	Data valid (write cycle)	after R.E., PHI1 T2		70		60		50	ns
t_{Dh}	4-4	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	4-5	Address bits 0–23 setup	before $\overline{ADS}$ T.E.	25		25		25		ns
t_{ALADSh}	4-10	Address bits 0–23 hold	after $\overline{ADS}$ T.E.	25		20		15		ns
t_{ALf}	4-5	Address bits 0–23 floating (no MMU)	after R.E., PHI1 T2		25		25		25	ns
t_{ADf}	4-5	Data bits D24–D31 floating (no MMU)	after R.E., PHI1 T2		25		25		25	ns
t_{ALMf}	4-9	Address bits 0–23 floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
t_{ADMf}	4-9	Data bits 21–31 floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
t_{BEv}	4-4	$\overline{BEn}$ signals valid	after R.E., PHI2 T4		70		60		45	ns
t_{BEh}	4-4	$\overline{BEn}$ signals hold	after R.E., PHI2 T4 or Ti	0		0		0		ns
t_{STv}	4-4	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		65		55		45	ns
t_{STh}	4-4	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t_{DDINv}	4-5	$\overline{DDIN}$ signal valid	after R.E., PHI1 T1		83		62		50	ns
t_{DDINh}	4-5	$\overline{DDIN}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	4-4	$\overline{ADS}$ signal active (low)	after R.E., PHI1 T1		55		45		35	ns
t_{ADSia}	4-4	$\overline{ADS}$ signal inactive	after R.E., PHI2 T1		60		55		45	ns
t_{ADSw}	4-4	$\overline{ADS}$ pulse width	at 0.8V (both edges)	50		40		30		ns
t_{DSa}	4-4	$\overline{DS}$ signal active (low)	after R.E., PHI1 T2		70		60		45	ns
t_{DSia}	4-4	$\overline{DS}$ signal inactive	after R.E., PHI1 T4		50		50		40	ns
t_{ALf}	4-6	AD0–AD23 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{ADf}	4-6	D24–D31 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{DSf}	4-6	$\overline{DS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{ADSf}	4-6	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{BEf}	4-6	$\overline{BEn}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-6	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HLDAa}	4-6	$\overline{HLDA}$ signal active (low)	after R.E., PHI1 Ti		100		90		75	ns
t_{HLDAia}	4-8	$\overline{HLDA}$ signal inactive	after R.E., PHI1 Ti		100		90		75	ns
t_{DSr}	4-8	$\overline{DS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{ADSr}	4-8	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{BEr}	4-8	$\overline{BEn}$ signals return from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINr}	4-8	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-9	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		80		65		50	ns
t_{DDINr}	4-10	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		75		65		50	ns
t_{SPCa}	4-13	$\overline{SPC}$ output active (low)	after R.E., PHI1 T1		50		45		35	ns
t_{SPCia}	4-13	$\overline{SPC}$ output inactive	after R.E., PHI1 T4		50		45		35	ns
t_{SPCnf}	4-15	$\overline{SPC}$ output nonforcing	after R.E., PHI2 T4		40		25		10	ns
t_{Dv}	4-13	Data valid (slave processor write)	after R.E., PHI1 T1		80		65		50	ns
t_{Dh}	4-13	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	4-18	$\overline{PFS}$ pulse width	at 0.8V (both edges)	70		70		70		ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t_{PFSa}	4-18	$\overline{PFS}$ pulse active (low)	after R.E., PHI2		70		60		50	ns
t_{PFSia}	4-18	$\overline{PFS}$ pulse inactive	after R.E., PHI2		70		60		50	ns
t_{ILOs}	4-20a	$\overline{ILO}$ signal setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns
t_{ILOh}	4-20b	$\overline{ILO}$ signal hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t_{ILOa}	4-21	$\overline{ILO}$ signal active (low)	after R.E., PHI1		70		65		55	ns
t_{ILOia}	4-21	$\overline{ILO}$ signal inactive	after R.E., PHI1		70		65		55	ns
t_{USv}	4-22	U/$\overline{S}$ signal valid	after R.E., PHI1 T4		70		60		45	ns
t_{USh}	4-22	U/$\overline{S}$ signal hold	after R.E., PHI1 T4	10		10		10		ns
t_{NSPF}	4-19b	Nonsequential fetch to next $\overline{PFS}$ clock cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	4-19a	$\overline{PFS}$ clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	4-29	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	before R.E., PHI1 T1 of first of first bus cycle of transfer	0		0		0		t_{Cp}

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

4.4.2.2 Input Signal Requirements: NS32032-6, NS32032-8, NS32032-10

Name	Figure	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t_{PWR}	4-25	Power stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	4-5	Data in setup (read cycle)	before F.E., PHI2 T3	20		15		10		ns
t_{DIh}	4-5	Data in hold (read cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	4-6	$\overline{HOLD}$ active (low) setup time (see note)	before F.E., PHI2 TX1	25		25		25		ns
t_{HLDia}	4-8	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	4-6	$\overline{HOLD}$ hold time	after R.E., PHI1 TX2	0		0		0		ns
t_{FLTa}	4-9	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25		25		25		ns
t_{FLTia}	4-10	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T2	25		25		25		ns
t_{RDYs}	4-11, 4-12	RDY setup time	before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	4-11, 4-12	RDY hold time	after F.E., PHI1 T3	0		0		0		ns
t_{ABTs}	4-23	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t_{ABTs}	4-24	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 T_f	30		25		20		ns
t_{ABTh}	4-23	$\overline{ABT}$ hold time	after R.E., PHI1	0		0		0		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements NS32032-6, NS32032-8, NS32032-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t_{RSTs}	4-25, 4-26	$\overline{RST}$ setup time	before F.E., PHI1	20		20		15		ns
t_{RSTw}	4-26	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	4-27	$\overline{INT}$ setup time	before F.E., PHI1	25		25		25		ns
t_{NMIw}	4-28	$\overline{NMI}$ pulse width	at 0.8V (both edges)	70		70		70		ns
t_{DIs}	4-14	Data setup (slave read cycle)	before F.E., PHI2 T1	30		25		20		ns
t_{DIh}	4-14	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	4-15	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	40		35		25		ns
t_{SPCs}	4-15	$\overline{SPC}$ setup time	before F.E., PHI1	35		30		25		ns
t_{SPCw}	4-15	$\overline{SPC}$ pulse width from slave processor (async input)	at 0.8V (both edges)	30		25		20		ns
t_{ATs}	4-16	$\overline{AT}/\overline{SPC}$ setup for address translation strap	before R.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1		1		1		t_{Cp}
t_{ATh}	4-16	$\overline{AT}/\overline{SPC}$ hold for address translation strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		t_{Cp}

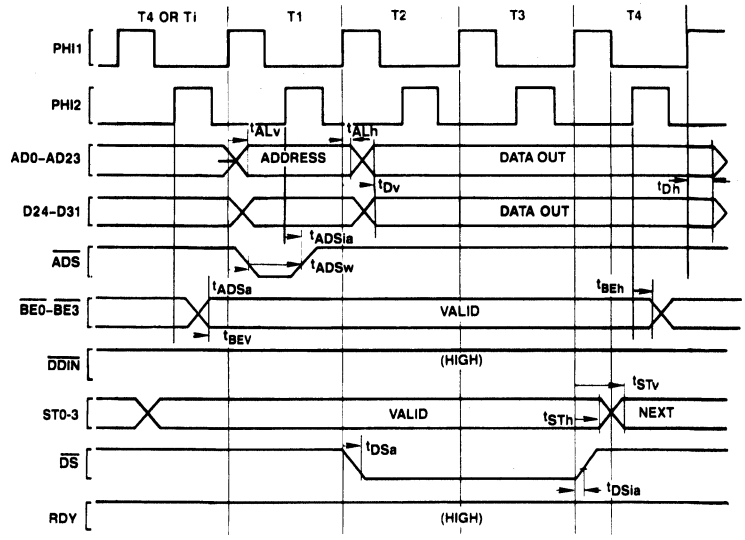
Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

4.4.2.3 Clocking Requirements: NS32032-6, NS32032-8, NS32032-10

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4-17	PHI1, PHI2 rise time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	4-17	PHI1, PHI2 fall time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	4-17	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4-17	PHI1, PHI2 pulse width	At 2.0V on PHI1, PHI2 (both edges)	$0.5t_{Cp}$ −14ns		$0.5t_{Cp}$ −12ns		$0.5t_{Cp}$ −10ns		
$t_{CLh(1,2)}$	4-17	PHI1, PHI2 high time	At V_{CC} − 0.9V on PHI1, PHI2 (both edges)	$0.5t_{Cp}$ −18ns		$0.5t_{Cp}$ −17ns		$0.5t_{Cp}$ −15ns		
$t_{nOVL(1,2)}$	4-17	Non-overlap time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	at 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}		PHI1, PHI2 asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	at 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

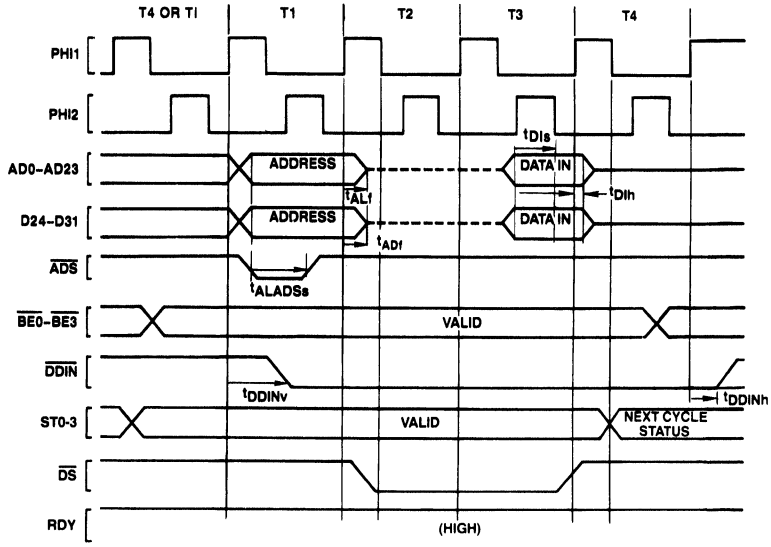
4.0 Device Specifications

4.4.3 Timing Diagrams



TL/EE/5491–45

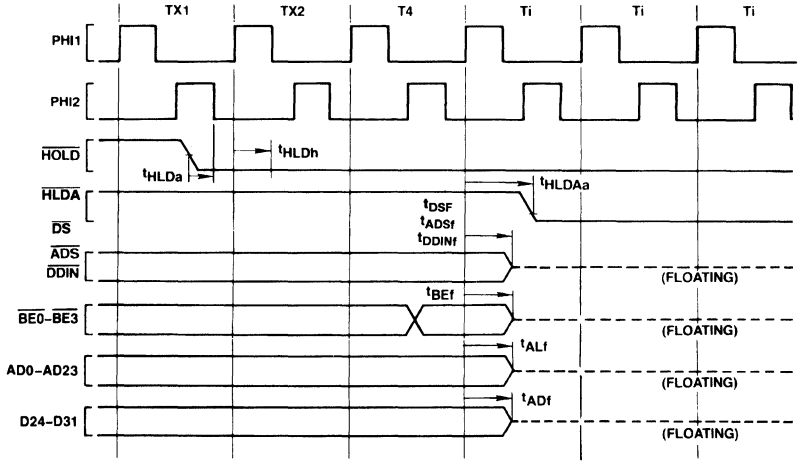
FIGURE 4-4. Write Cycle



TL/EE/5491–46

FIGURE 4-5. Read Cycle

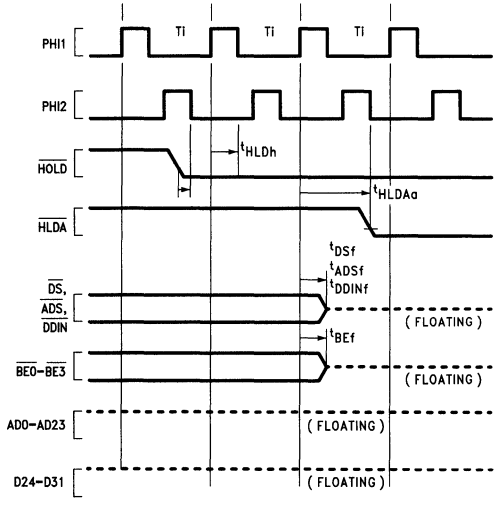
4.0 Device Specifications (Continued)



TL/EE/5491-47

FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially).

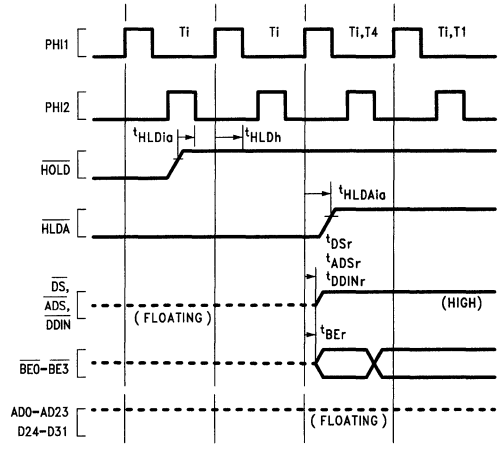
Note that whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ request ($\overline{\text{HOLD}}$ low) must be active tHLDa before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until tHLDh after the rising edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.



TL/EE/5491-48

FIGURE 4-7. Floating by $\overline{\text{HOLD}}$ Timing (CPU initially idle)

Note that during Ti1 the CPU is already idling.

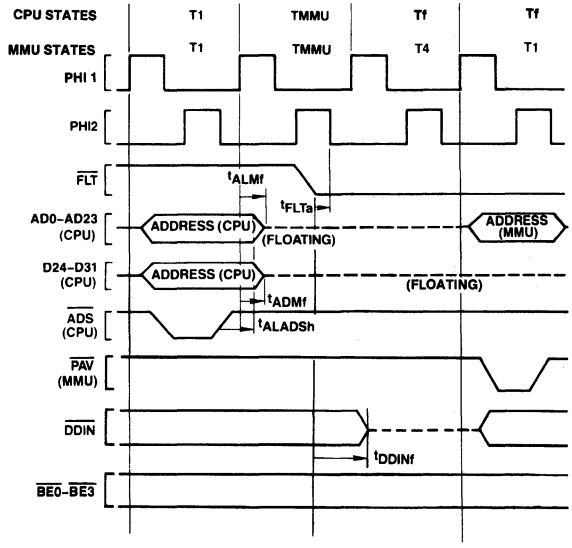


TL/EE/5491-49

FIGURE 4-8. Release from $\overline{\text{HOLD}}$

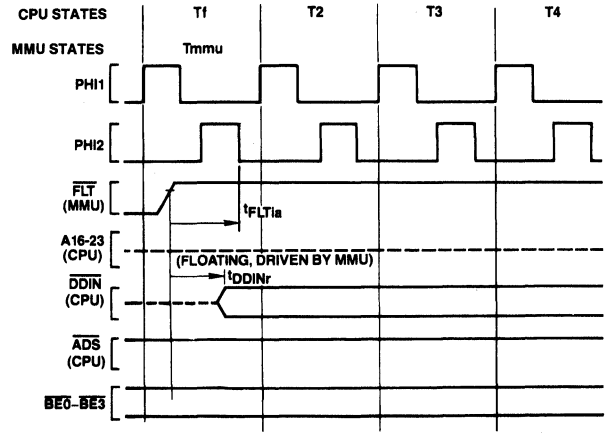
2

4.0 Device Specifications (Continued)



TL/EE/5491–50

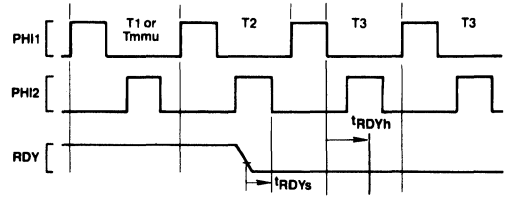
FIGURE 4-9. FLT Initiated Float Cycle Timing



TL/EE/5491–51

FIGURE 4-10. Release from FLT Timing

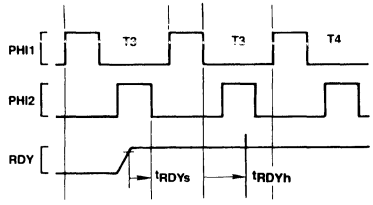
Note that when FLT is deasserted the CPU restarts driving DDIN before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force DDIN to the same logic level.



TL/EE/5491–52

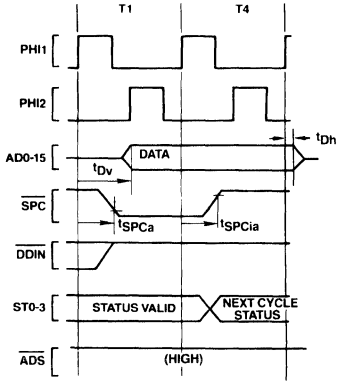
FIGURE 4-11. Ready Sampling (CPU Initially READY)

4.0 Device Specifications (Continued)



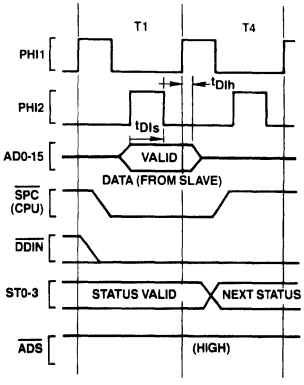
TL/EE/5491–53

FIGURE 4-12. Ready Sampling (CPU Initially NOT READY)



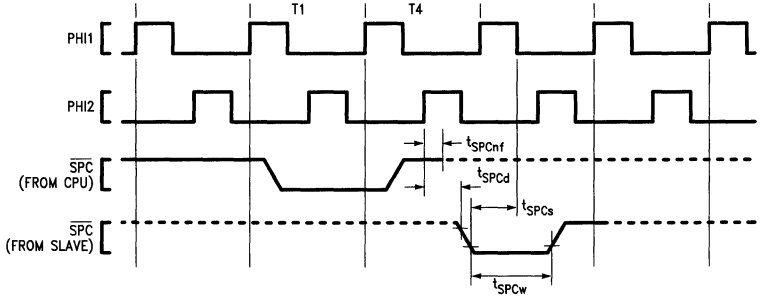
TL/EE/5491–54

FIGURE 4-13. Slave Processor Write Timing



TL/EE/5491–55

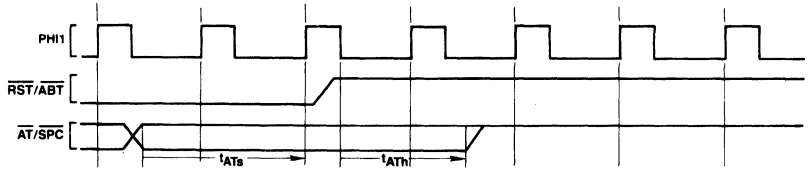
FIGURE 4-14. Slave Processor Read Timing



TL/EE/5491–82

FIGURE 4-15. $\overline{\text{SPC}}$ Timing

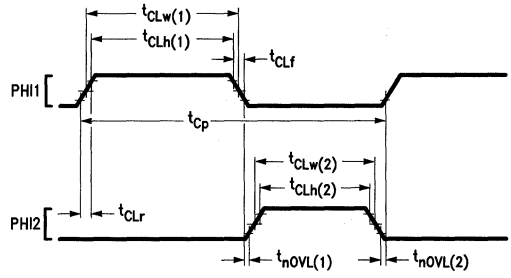
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{\text{SPC}}$ high with internal 5 kΩ pullup.



TL/EE/5491–57

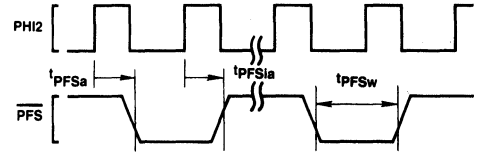
FIGURE 4-16. Reset Configuration Timing

4.0 Device Specifications (Continued)



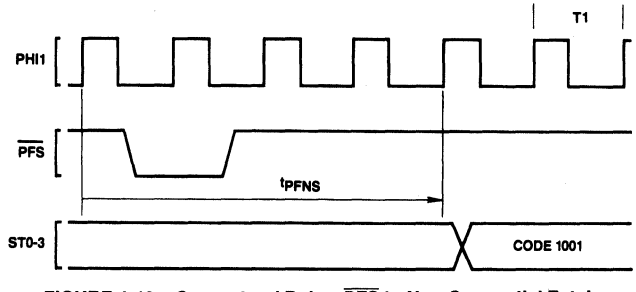
TL/EE/5491–58

FIGURE 4-17. Clock Waveforms



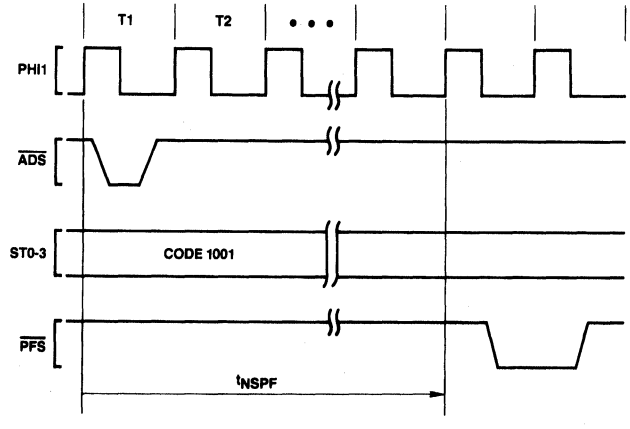
TL/EE/5491–59

FIGURE 4-18. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/5491–60

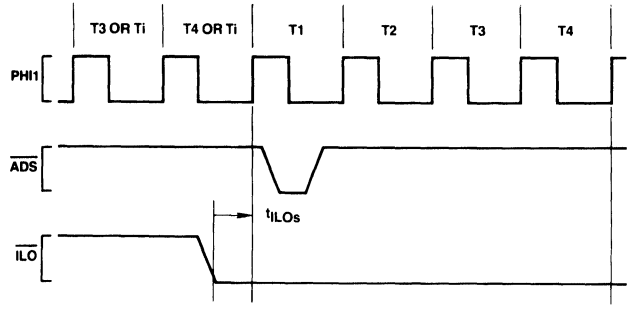
FIGURE 4-19a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/5491–61

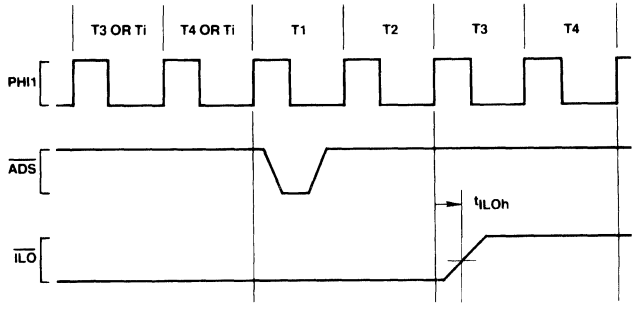
FIGURE 4-19b. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



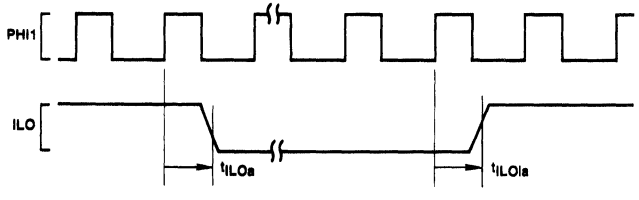
TL/EE/5491–62

FIGURE 4-20a. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction



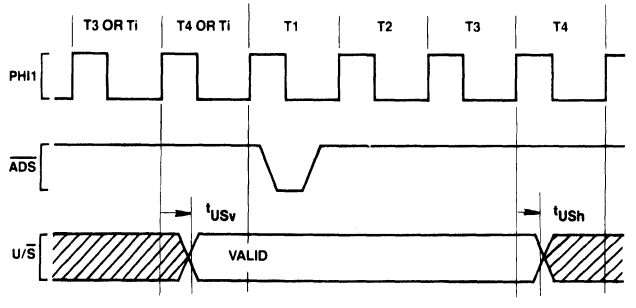
TL/EE/5491–63

FIGURE 4-20b. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction



TL/EE/5491–64

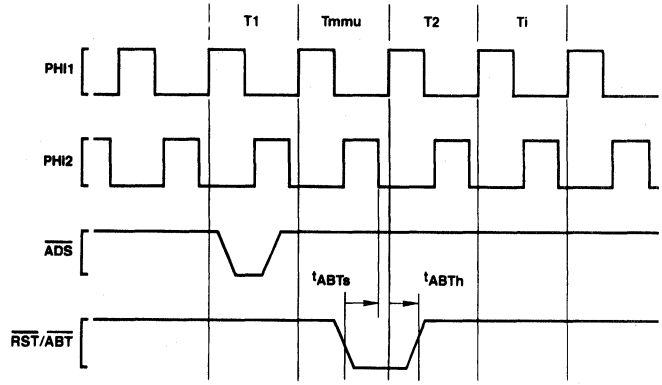
FIGURE 4-21. Relationship of $\overline{ILO}$ to Any Clock Cycle



TL/EE/5491–65

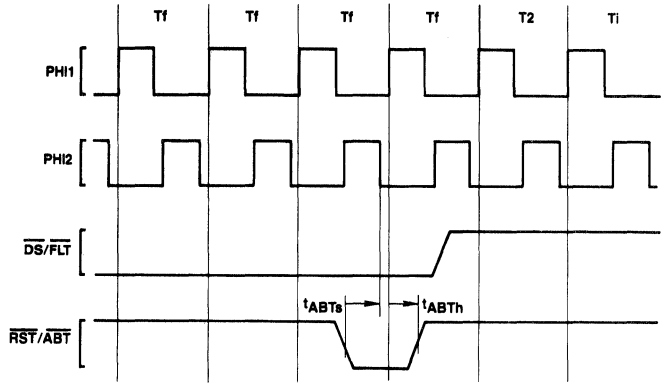
FIGURE 4-22. U/$\overline{S}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval

4.0 Device Specifications (Continued)



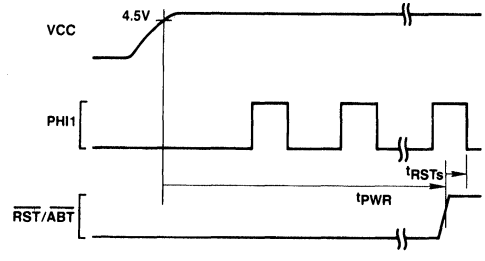
TL/EE/5491–66

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Not Applied



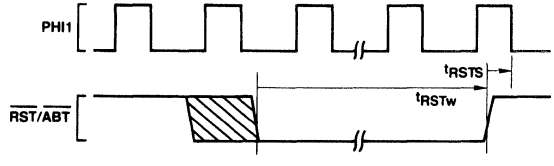
TL/EE/5491–67

FIGURE 4-24. Abort Timing, $\overline{FLT}$ Applied



TL/EE/5491–68

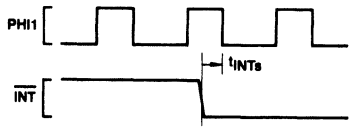
FIGURE 4-25. Power-On Reset



TL/EE/5491–69

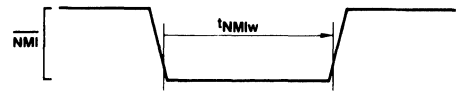
FIGURE 4-26. Non-Power-On Reset

4.0 Device Specifications (Continued)



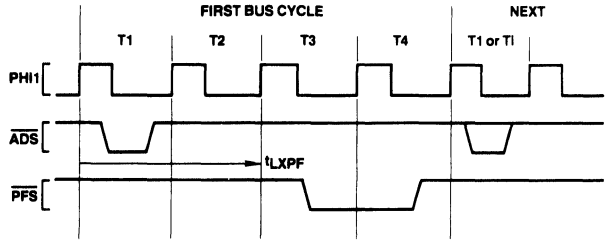
TL/EE/5491–70

FIGURE 4-27. $\overline{INT}$ Interrupt Signal Detection



TL/EE/5491–71

FIGURE 4-28. $\overline{NMI}$ Interrupt Signal Timing



TL/EE/5491–72

FIGURE 4-29. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction

Note: In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

2

Appendix A: Instruction Formats

NOTATIONS

i= Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f= Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c= Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op= Operation Code
Valid encodings shown with each format.

gen, gen 1, gen 2= General Addressing Mode Field
See Sec. 2.2 for encodings.

reg= General Purpose Register Number

cond= Condition Code Field
- 0000 = EQual: Z = 1
- 0001 = Not Equal: Z = 0
- 0010 = Carry Set: C = 1
- 0011 = Carry Clear: C = 0
- 0100 = HIgher: L = 1
- 0101 = Lower or Same: L = 0
- 0110 = Greater Than: N = 1
- 0111 = Less or Equal: N = 0
- 1000 = Flag Set: F = 1
- 1001 = Flag Clear: F = 0
- 1010 = LOwer: L = 0 and Z = 0
- 1011 = Higher or Same: L = 1 or Z = 1
- 1100 = Less Than: N = 0 and Z = 0
- 1101 = Greater or Equal: N = 1 or Z = 1
- 1110 = (Unconditionally True)
- 1111 = (Unconditionally False)

short= Short Immediate value. May contain
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.
  - 0000 = US
  - 0001 – 0111 = (Reserved)
  - 1000 = FP
  - 1001 = SP
  - 1010 = SB
  - 1011 = (Reserved)
  - 1100 = (Reserved)
  - 1101 = PSR
  - 1110 = INTBASE
  - 1111 = MOD
- Options: in String Instructions

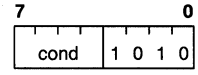
U/W	B	T

  - T = Translated
  - B = Backward
  - U/W = 00: None
    - 01: While Match
    - 11: Until Match

Configuration bits, in SETCFG:

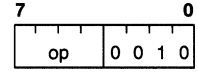
C	M	F	I

mreg: MMU Register number, in LMR, SMR.
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = PF0
- 0101 = PF1
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = SC
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA



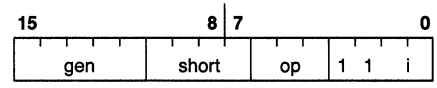
Format 0

Bcond	(BR)		



Format 1

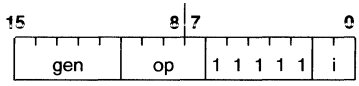
BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111



Format 2

ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Scond	-011		

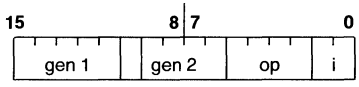
Appendix A: Instruction Formats (Continued)



Format 3

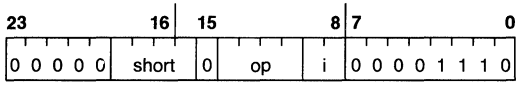
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

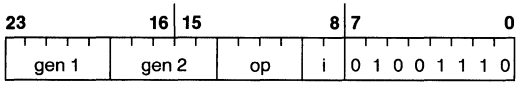
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

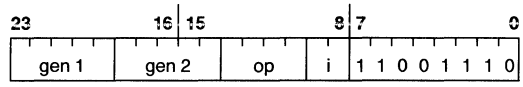
MOVS	-0000	SETCFG	-0010
CMPS	-0001	SKPS	-0011

Trap (UND) on 1XXX, 01XX



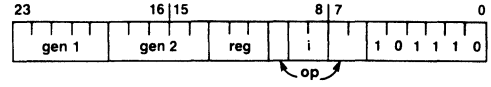
Format 6

ROT	-0000	NEG	-1000
ASH	-0001	NOT	-1001
CBIT	-0010	Trap (UND)	-1010
CBITI	-0011	SUBP	-1011
Trap (UND)	-0100	ABS	-1100
LSH	-0101	COM	-1101
SBIT	-0110	IBIT	-1110
SBITI	-0111	ADDP	-1111



Format 7

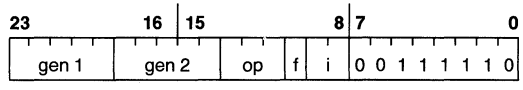
MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZiD	-0110	MOD	-1110
MOVXiD	-0111	DIV	-1111



TL/EE/5491–73

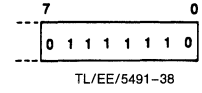
Format 8

EXT	-0 00	INDEX	-1 00
CVTP	-0 01	FFS	-1 01
INS	-0 10		
CHECK	-0 11		
MOVSU	-110, reg = 001		
MOVUS	-110, reg = 011		



Format 9

MOVif	-000	ROUND	-100
LFSR	-001	TRUNC	-101
MOVLF	-010	SFSR	-110
MOVFL	-011	FLOOR	-111

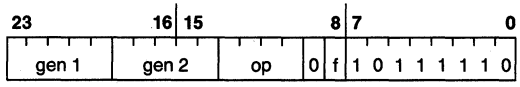


TL/EE/5491–38

Format 10

Trap (UND) Always

Appendix A: Instruction Formats (Continued)



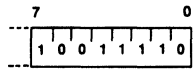
Format 11

ADDf	-0000	DIVf	-1000
MOVf	-0001	Trap (SLAVE)	-1001
CMPf	-0010	Trap (UND)	-1010
Trap (SLAVE)	-0011	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

TL/EE/5491–75

Format 12

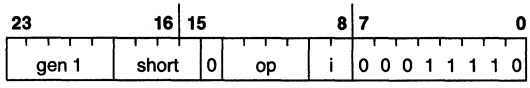
Trap (UND) Always



TL/EE/5491–76

Format 13

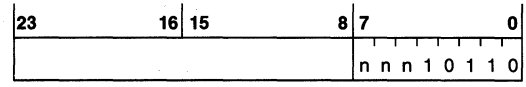
Trap (UND) Always



Format 14

RDVAL	-0000	LMR	-0010
WRVAL	-0001	SMR	-0011

Trap (UND) on 01XX, 1XXX



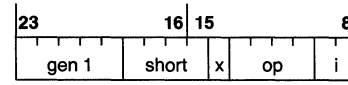
Operation Word ID Byte

Format 15

(Custom Slave)

nnn **Operation Word Format**

000

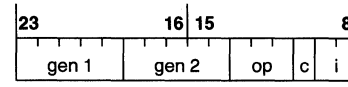


Format 15.0

CATST0	-0000	LCR	-1010
CATST1	-0001	SCR	-1011

Trap (UND) on all others

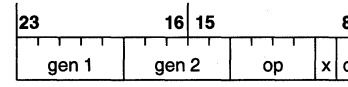
001



Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111

101

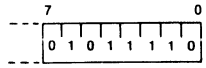


Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	CMOV3	-1001
CCMP0	-0010	Trap (UND)	-1010
CCMP1	-0011	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

If nnn = 010, 011, 100, 110, 111
then Trap (UND) Always

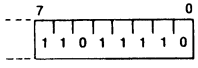
Appendix A: Instruction Formats (Continued)



TL/EE/5491–77

Format 16

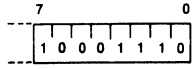
Trap (UND) Always



TL/EE/5491–78

Format 17

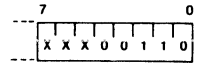
Trap (UND) Always



TL/EE/5491–79

Format 18

Trap (UND) Always

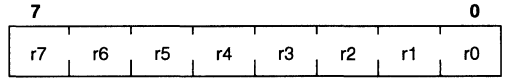


TL/EE/5491–80

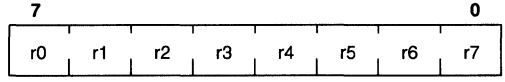
Format 19

Trap (UND) Always

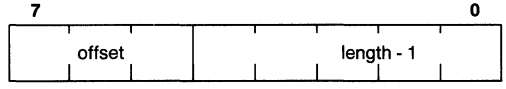
Implied Immediate Encodings:



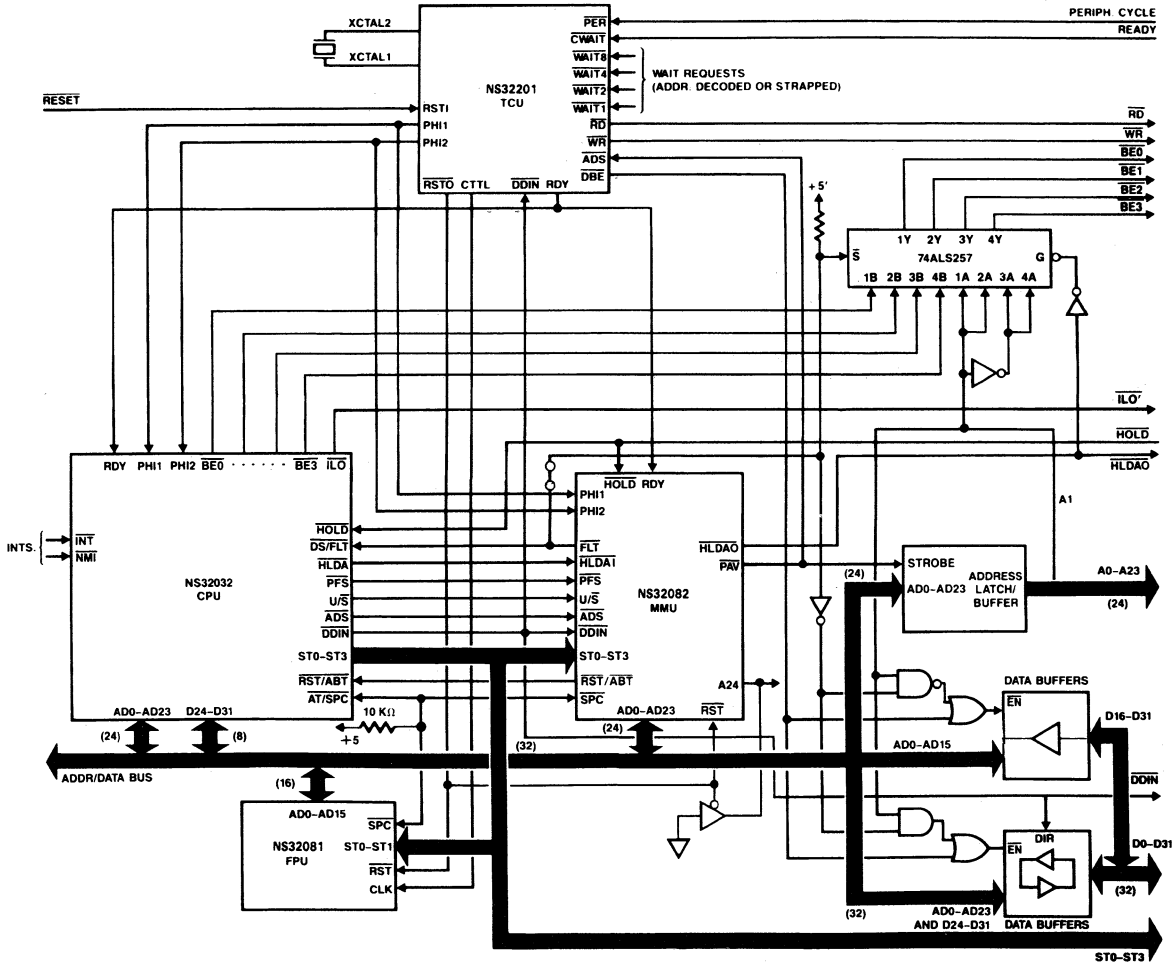
Register Mark, appended to SAVE, ENTER



Register Mark, appended to RESTORE, EXIT



Offset/Length Modifier appended to INSS, EXTS



TL/EE/5491–74

FIGURE B-1. System Connection Diagram

National Semiconductor Corporation

PRELIMINARY

NS32C016-6/NS32C016-10/NS32C016-15 High-Performance Microprocessors

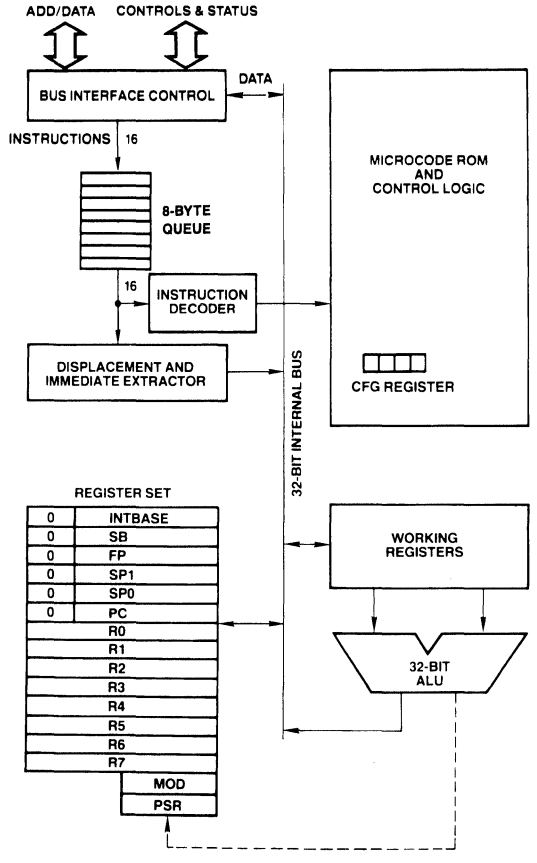
General Description

The NS32C016 is a 32-bit, CMOS microprocessor with TTL compatible inputs. The NS32C016 has a 16M byte linear address space and a 16-bit external data bus. It is fabricated with National Semiconductor's advanced CMOS process and is fully object code compatible with other Series 32000® CPU's. The NS32C016 has a 32-bit ALU, eight 32-bit general purpose registers, an eight-byte prefetch queue and a highly symmetric architecture. It also incorporates a slave processor interface and provides for full virtual memory capability in conjunction with the NS32082 memory management unit (MMU). High performance floating-point instructions are provided with the NS32081 floating-point unit (FPU). The NS32C016 is intended for a wide range of high performance computer applications.

Features

- 32-bit architecture and implementation
- 16M byte uniform addressing space
- Powerful instruction set
  - — General 2-address capability
  - — Very high degree of symmetry
  - — Addressing modes optimized for high-level Language references
- High-speed CMOS technology
- TTL compatible inputs
- Single 5V supply
- 48-pin dual-in-line package

Block Diagram



TL/EE/8525-1

Table of Contents

1.0 PRODUCT INTRODUCTION

2.0 ARCHITECTURAL DESCRIPTION

- 2.1 Programming Model
  - 2.1.1 General Purpose Registers
  - 2.1.2 Dedicated Registers
  - 2.1.3 The Configuration Register (CFG)
  - 2.1.4 Memory Organization
  - 2.1.5 Dedicated Tables
- 2.2 Instruction Set
  - 2.2.1 General Instruction Format
  - 2.2.2 Addressing Modes
  - 2.2.3 Instruction Set Summary

3.0 FUNCTIONAL DESCRIPTION

- 3.1 Power and Grounding
- 3.2 Clocking
- 3.3 Resetting
- 3.4 Bus Cycles
  - 3.4.1 Cycle Extension
  - 3.4.2 Bus Status
  - 3.4.3 Data Access Sequences
    - 3.4.3.1 Bit Accesses
    - 3.4.3.2 Bit Field Accesses
    - 3.4.3.3 Extending Multiply Accesses
  - 3.4.4 Instruction Fetches
  - 3.4.5 Interrupt Control Cycles
  - 3.4.6 Slave Processor Communication
    - 3.4.6.1 Slave Processor Bus Cycles
    - 3.4.6.2 Slave Operand Transfer Sequences
- 3.5 Memory Management Option
  - 3.5.1 Address Translation Strap
  - 3.5.2 Translated Bus Timing
  - 3.5.3 The $\overline{\text{FLT}}$ (Float) Pin
  - 3.5.4 Aborting Bus Cycles
    - 3.5.4.1 The Abort Interrupt
    - 3.5.4.2 Hardware Considerations
- 3.6 Bus Access Control
- 3.7 Instruction Status
- 3.8 NS32C016 Interrupt Structure
  - 3.8.1 General Interrupt/Trap Sequence
  - 3.8.2 Interrupt/Trap Return
  - 3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)
    - 3.8.3.1 Non-Vectored Mode
    - 3.8.3.2 Vectored Mode: Non-Cascaded Case
    - 3.8.3.3 Vectored Mode: Cascaded Case
  - 3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)
  - 3.8.5 Traps
  - 3.8.6 Prioritization
  - 3.8.7 Interrupt/Trap Sequences: Detail Flow
    - 3.8.7.1 Maskable/Non-Maskable Interrupt Sequence
    - 3.8.7.2 Trap Sequence: Traps Other Than Trace
    - 3.8.7.3 Trace Trap Sequence
    - 3.8.7.4 Abort Sequence

Table of Contents (Continued)

3.9 Slave Processor Instructions
3.9.1 Slave Processor Protocol
3.9.2 Floating Point Instructions
3.9.3 Memory Management Instructions
3.9.4 Custom Slave Instructions

4.0 DEVICE SPECIFICATIONS

4.1 NS32C016 Pin Descriptions
4.1.1 Supplies
4.1.2 Input Signals
4.1.3 Output Signals
4.1.4 Input-Output Signals
4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics
4.4.1 Definitions
4.4.2 Timing Tables
4.4.2.1 Output Signals: Internal Propagation Delays
4.4.2.2 Input Signal Requirements
4.4.2.3 Clocking Requirements

Appendix A: Instruction Formats

Appendix B: Interfacing Suggestions

List of Illustrations

The General and Dedicated Registers 2-1
Processor Status Register 2-2
CFG Register 2-3
Module Descriptor Format 2-4
A Sample Link Table 2-5
General Instruction Format 2-6
Index Byte Format 2-7
Displacement Encodings 2-8
Recommended Supply Connections 3-1
Clock Timing Relationships 3-2
Power-On Reset Requirements 3-3
General Reset Timing 3-4
Recommended Reset Connections, Non-Memory-Managed System 3-5a
Recommended Reset Connections, Memory-Managed System 3-5b
Bus Connections 3-6
Read Cycle Timing 3-7
Write Cycle Timing 3-8
RDY Pin Timing 3-9
Extended Cycle Example 3-10
Memory Interface 3-11
Slave Processor Connections 3-12
CPU Read from Slave Processor 3-13
CPU Write to Slave Processor 3-14
Read Cycle with Address Translation (CPU Action) 3-15
Write Cycle with Address Translation (CPU Action) 3-16
Memory-Managed Read Cycle 3-17
Memory-Managed Write Cycle 3-18
$\overline{\text{FLT}}$ Timing 3-19
$\overline{\text{HOLD}}$ Timing, Bus Initially Idle 3-20
$\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle 3-21

List of Illustrations (Continued)

Interrupt Dispatch and Cascade Tables 3-22
Interrupt/Trap Service Routine Calling Sequence 3-23
Return from Trap (RETT n) Instruction Flow 3-24
Return from Interrupt (RET I) Instruction Flow 3-25
Interrupt Control Unit Connections (16 Levels) 3-26
Cascaded Interrupt Control Unit Connections 3-27
Slave Processor Status Word Format 3-30
Connection Diagram 4-1
Timing Specification Standard (CMOS Output Signals) 4-2
Timing Specification Standard (TTL Input Signals) 4-3
Write Cycle 4-4
Read Cycle 4-5
Floating by $\overline{HOLD}$ Timing (CPU Not Idle Initially) 4-6
Floating by $\overline{HOLD}$ Timing (CPU Initially Idle) 4-7
Release from $\overline{HOLD}$ 4-8
$\overline{FLT}$ Initiated Cycle Timing 4-9
Release from $\overline{FLT}$ Timing 4-10
Ready Sampling (CPU Initially READY) 4-11
Ready Sampling (CPU Initially NOT READY) 4-12
Slave Processor Write Timing 4-13
Slave Processor Read Timing 4-14
$\overline{SPC}$ Non-Forcing Delay 4-15
Reset Configuration Timing 4-16
Clock Waveforms 4-17
Relationship of $\overline{PFS}$ to Clock Cycles 4-18
Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch 4-19a
Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$ 4-19b
Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction 4-20a
Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction 4-20b
Relationship of $\overline{ILO}$ to Any Clock Cycle 4-21
U/$\overline{S}$ Relationship to any Bus Cycle–Guaranteed Valid Interval 4-22
Abort Timing, $\overline{FLT}$ Not Applied 4-23
Abort Timing, $\overline{FLT}$ Applied 4-24
Power-On Reset 4-25
Non-Power-On Reset 4-26
$\overline{INT}$ Interrupt Signal Detection 4-27
$\overline{NMI}$ Interrupt Signal Timing 4-28
Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction 4-29

List of Tables

NS32C016 Addressing Modes 2-1
NS32C016 Instruction Set Summary 2-2
Bus Cycle Categories 3-1
Access Sequences 3-2
Interrupt Sequences 3-3
Floating Point Instruction Protocols 3-4
Memory Management Instruction Protocols 3-5
Custom Slave Instruction Protocols 3-6

1.0 Product Introduction

The Series 32000 Microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from a minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operations. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes. Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types. The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set. While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations. The Series 32000 CPUs represent two-operand machines with each operand addressable by all addressing modes. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management. Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing. The NS32C016 has 24-bit address pointers that can address up to 16 megabytes without any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support. Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept. The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32C016 CPU.

2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

2.1.2 Dedicated Registers

The eight dedicated registers of the NS32C016 are assigned specific functions.

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32C016 the upper eight bits of this register are always zero.)

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and

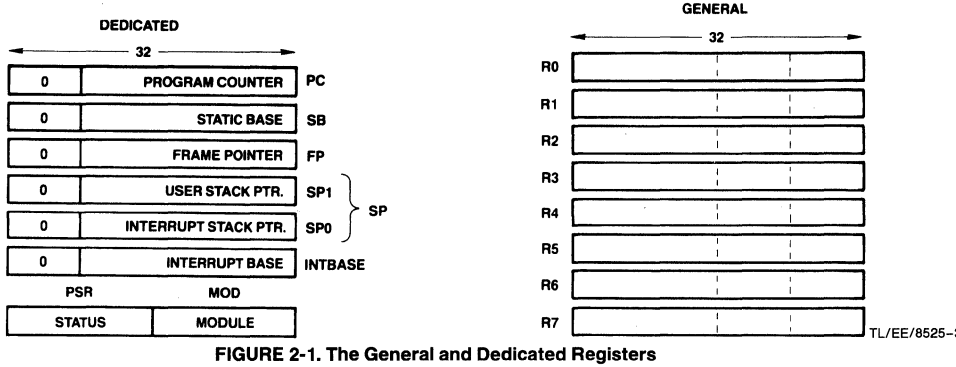


FIGURE 2-1. The General and Dedicated Registers

2.0 Architectural Description (Continued)

trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 then SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32C016 the upper eight bits of these registers are always zero.)

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.

FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32C016 the upper eight bits of this register are always zero.)

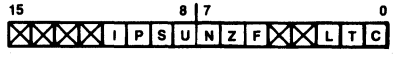
SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32C016 the upper eight bits of this register are always zero.)

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32C016 the upper eight bits of this register are always zero.)

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64k bytes of memory.

PSR: The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the NS32C016 microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/8525–78

FIGURE 2-2. Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Section 3.8.5).

L: The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U=0 the NS32C016 is said to be in Supervisor Mode; when U=1 the NS32C016 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Section 3.8.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I=1, then all interrupts will be accepted (Section 3.8). If I=0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32C016 CPU is the four-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3.*

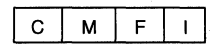


FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{\text{INT}}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Section 3.8.

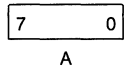
The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

2.1.4 Memory Organization

The main memory of the NS32C016 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at $2^{24} - 1$. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and

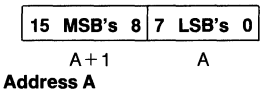
2.0 Architectural Description (Continued)

the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.



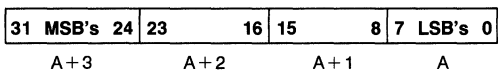
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Section 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.



Word at Address A

Two contiguous words are called a double word. Except where noted (Section 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



Double Word at Address A

Although memory is addressed as bytes, it is actually organized as words. Therefore, words and double words that are aligned to start at even addresses (multiples of two) are accessed more quickly than words and double words that are not so aligned.

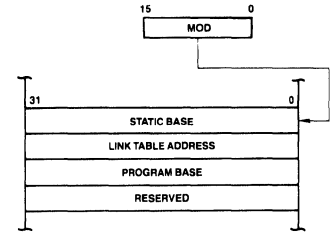
2.1.5 Dedicated Tables

Two of the NS32C016 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Section 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by the NS32C016. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically updated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in *Figure 2-4*. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.



TL/EE/8525–4

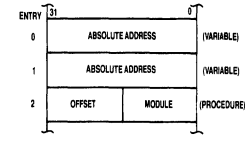
FIGURE 2-4. Module Descriptor Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-5*. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/8525–5

FIGURE 2-5. A Sample Link Table

2.2 INSTRUCTION SET

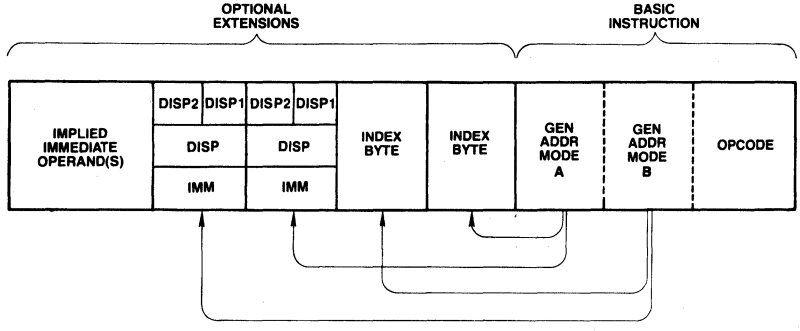
2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-7*.

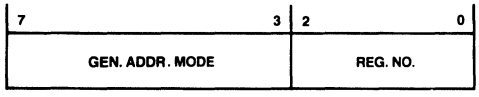
Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain

2.0 Architectural Description (Continued)



TL/EE/8525-6

FIGURE 2-6. General Instruction Format



TL/EE/8525-7

FIGURE 2-7. Index Byte Format

one of two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in *Figure 2-8,* with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most-significant byte first. Note that this is different from the memory representation of data (Section 2.1.4).

Some instructions require additional "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Section 2.2.3).

2.2.2 Addressing Modes

The NS32C016 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the NS32C016 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32C016 Addressing Modes fall into nine basic types:

Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

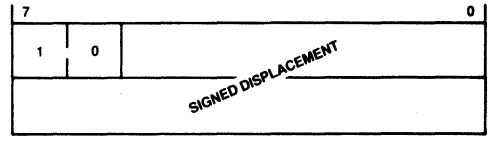
Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

Memory Space: Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

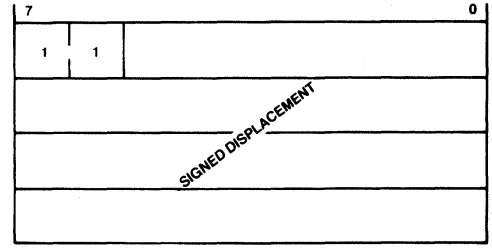
Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A

Byte Displacement: Range −64 to +63

Word Displacement: Range −8192 to +8191



Double Word Displacement: Range (Entire Addressing Space)



TL/EE/8525-8

FIGURE 2-8. Displacement Encodings

displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

2.0 Architectural Description (Continued)

Scaled Index: Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Series 32000 Instruction Set Reference Manual.

TABLE 2-1. NS32C016 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R6 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1 (FP))	Disp2 + Pointer; Pointer found at address Disp 1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1 (SP))	
10010	Static memory relative	disp2(disp1 (SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top Of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	*+ disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2×Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4×Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8×Rn. "Mode" and "n" are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32C016 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Series 32000 Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0–R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

TABLE 2-2. NS32C016 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move effective address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDi	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply.
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to extended integer.
7	DEIi	gen,gen	Divide extended integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add packed.
6	SUBPi	gen,gen	Subtract packed.

2.0 Architectural Description (Continued)

TABLE 2-2. NS32C016 Instruction Set Summary (Continued)

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical shift, left or right.
6	ASHi	gen,gen	Arithmetic shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked.
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit.

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to bit field pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

2.0 Architectural Description (Continued)

TABLE 2-2. NS32C016 Instruction Set Summary (Continued)

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 — Comparison Value
R3 — Translation Table Pointer
R2 — String 2 Pointer
R1 — String 1 Pointer
R0 — Limit Count

Options on all string instructions are:

B (Backward): Decrement strong pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Description
5	MOVSi	options	Move string 1 to string 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare string 1 to string 2.
	CMPST	options	Compare, translating string 1 bytes.
5	SKPSi	options	Skip over string 1 entries.
	SKPST	options	Skip, translating bytes for until/while.

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor call.
1	FLAG		Flag trap.
1	BPT		Breakpoint trap.
1	ENTER	[reg list], disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save general purpose registers.
1	RESTORE	[reg list]	Restore general purpose registers.
2	LPRi	areg,gen	Load dedicated register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store dedicated register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust stack pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set configuration register. (Privileged)

2.0 Architectural Description (Continued)

TABLE 2-2. NS32C016 Instruction Set Summary (Continued)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a floating point value.
9	MOVLF	gen,gen	Move and shorten a long value to standard.
9	MOVFL	gen,gen	Move and lengthen a standard value to long.
9	MOVif	gen,gen	Convert any integer to standard or long floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load memory management register. (Privileged)
14	SMR	mreg,gen	Store memory management register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUi	gen,gen	Move a value from supervisor space to user space. (Privileged)
8	MOVUSi	gen,gen	Move a value from user space to supervisor space. (Privileged)

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

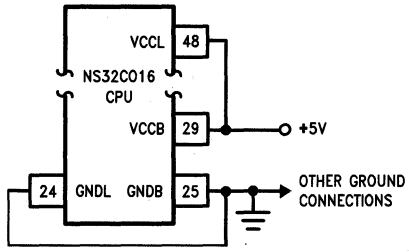
CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
15.5	CMOV3c	gen,gen	
15.5	CCMP0c	gen,gen	Custom compare.
15.5	CCMP1c	gen,gen	
15.1	CCV0ci	gen,gen	Custom convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load custom status register.
15.1	SCSR	gen	Store custom status register.
15.0	CATST0	gen	Custom address/test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load custom register. (Privileged)
15.0	SCR	creg,gen	Store custom register. (Privileged)

3.0 Functional Description

3.1 POWER AND GROUNDING

Power and ground connections for the NS32C016 are made on four pins. On-chip logic is connected to power through the logic power pin (VCCL, pin 48) and to ground through the logic ground pin (GNDL, pin 24). On-chip output drivers are connected to power through the buffer power pin (VCCP, pin 29) and to ground through the buffer ground pin (GNDB, pin 25). For optimal noise immunity, it is recommended that single conductors be connected directly from VCCL to VCCB and from GNDL to GNDB, as shown below (*Figure 3-1*).



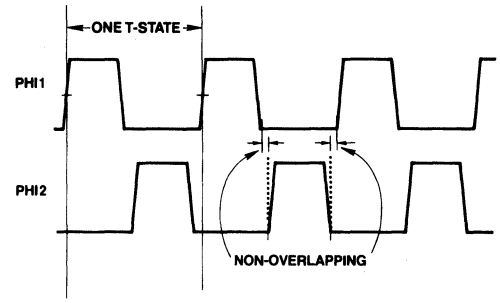
TL/EE/8525–9

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32C016 inputs clocking signals from the NS32C201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in *Figure 3-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See Section 4 for complete specifications of PHI1 and PHI2.



TL/EE/8525–10

FIGURE 3-2. Clock Timing Relationships

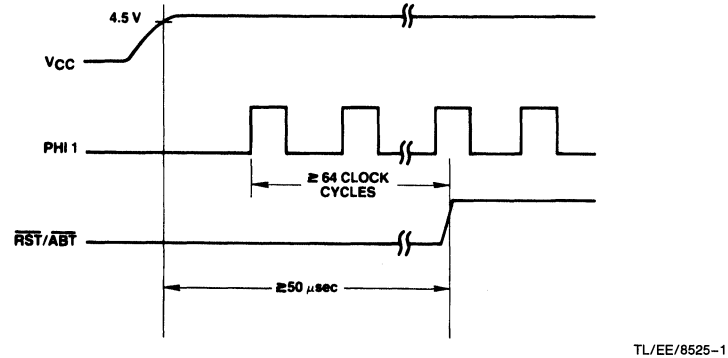
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 RESETTING

The $\overline{\text{RST}}/\overline{\text{ABT}}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Section 3.5.4.

The CPU may be reset at any time by pulling the $\overline{\text{RST}}/\overline{\text{ABT}}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, $\overline{\text{RST}}/\overline{\text{ABT}}$ must be held low for at least 50 μs after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active



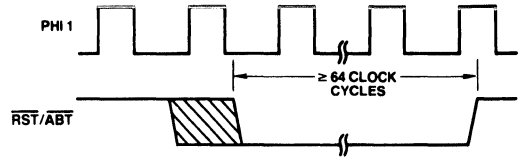
TL/EE/8525–11

FIGURE 3-3. Power-On Reset Requirements

3.0 Functional Description (Continued)

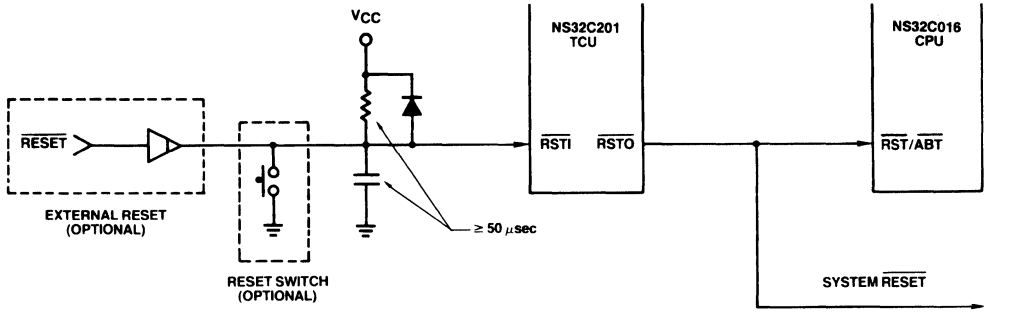
for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4*.

The NS32C201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32C016 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



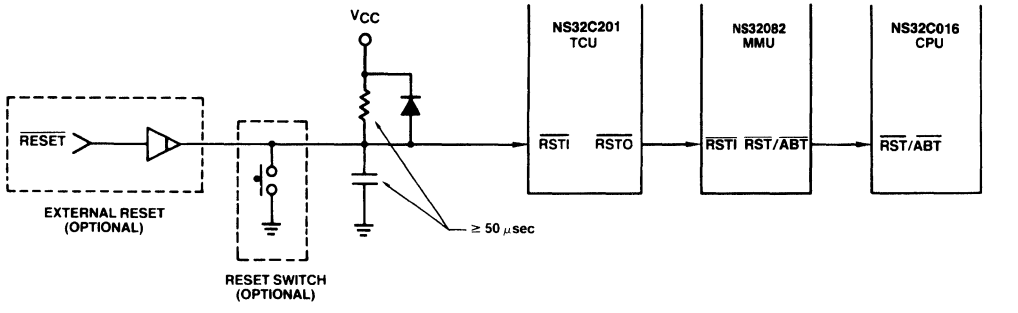
TL/EE/8525-12

FIGURE 3-4. General Reset Timing



TL/EE/8525-13

FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System



TL/EE/8525-14

FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

3.4 BUS CYCLES

The NS32C016 CPU has a strap option which defines the Bus Timing Mode as either With or Without Address Translation. This section describes only bus cycles under the No Address Translation option. For details of the use of the strap and of bus cycles with address translation, see Section 3.5.

The CPU will perform a bus cycle for one of the following reasons:

1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
2) To fetch instructions into the eight-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4) To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Section 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Section 3.4.6).

The sequence of events in a non-Slave bus cycle is shown in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Section 3.4.1).

3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

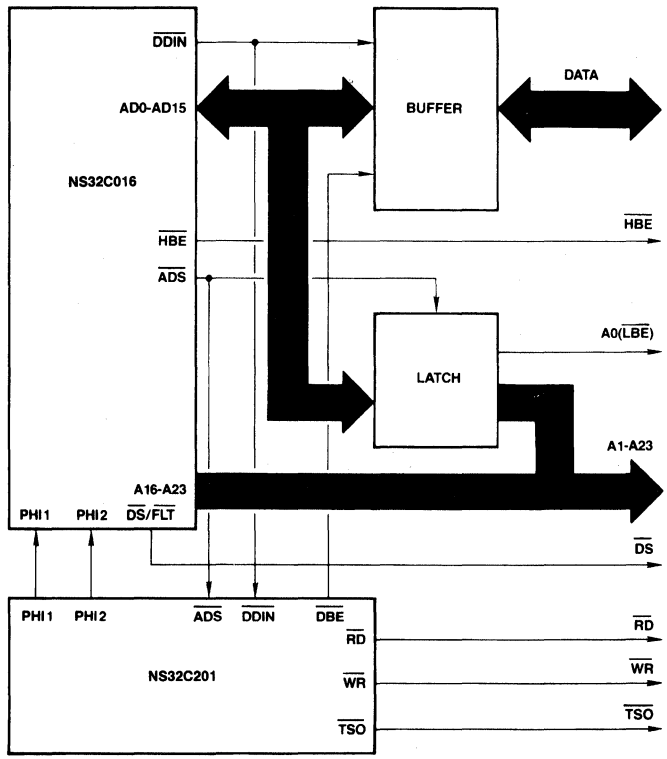
During T1, the CPU applies an address on pins AD0–AD15 and A16–A23. It also provides a low-going pulse on the $\overline{ADS}$ pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–15 from the AD0–AD15 pins. See *Figure 3-6*. During this time also the status signals $\overline{DDIN}$, indicating the direction of the transfer, and $\overline{HBE}$, indicating whether the high byte (AD8–AD15) is to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD15, to either accept or present data. Note that the signals A16–A23 remain valid, and need not be latched. It also starts the data strobe ($\overline{DS}$), signaling the beginning of the data transfer. Associated signals from the NS32C201 Timing Control Unit are also activated at this time: $\overline{RD}$ (Read Strobe) or $\overline{WR}$ (Write Strobe), $\overline{TSO}$ (Timing State Output, indicating that T2 has been reached) and $\overline{DBE}$ (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the end of T2, on the falling edge of the PHI2 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Section 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD15) is sampled at the falling edge of PHI2 of the last T3 state, see Section 4. Data must, however, be held at least until the beginning of T4. $\overline{DS}$ and $\overline{RD}$ are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

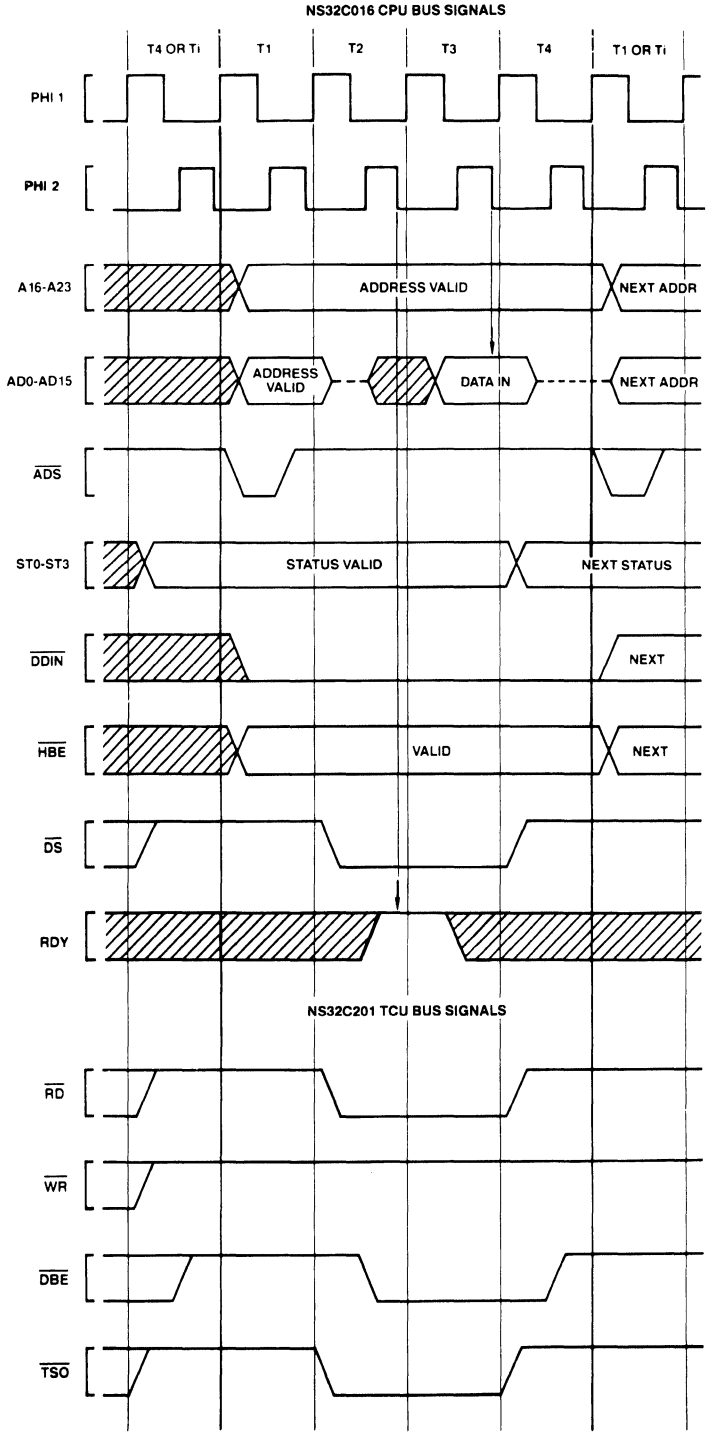
The T4 state finishes the bus cycle. At the beginning of T4, the $\overline{DS}$, $\overline{RD}$, or $\overline{WR}$, and $\overline{TSO}$ signals go inactive, and at the rising edge of PHI2, $\overline{DBE}$ goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).



TL/EE/8525–15

FIGURE 3-6. Bus Connections

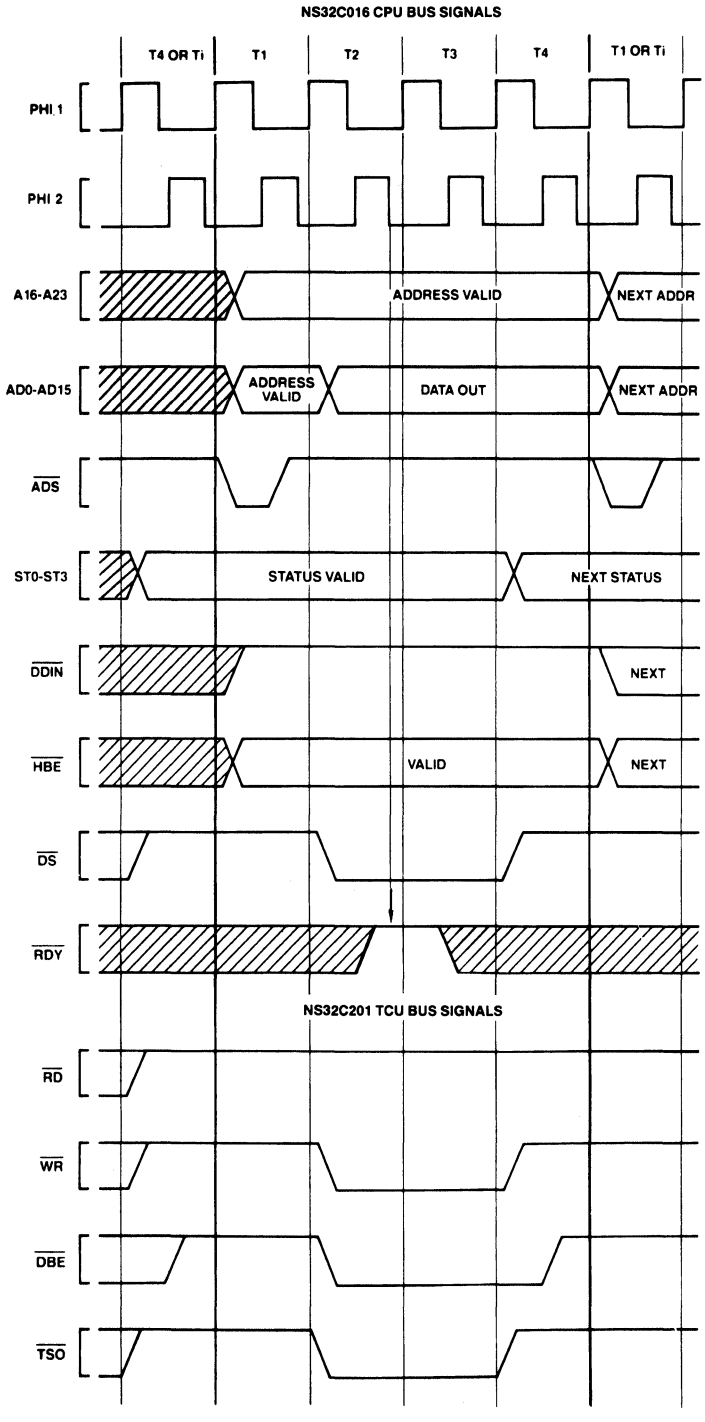
3.0 Functional Description (Continued)



TL/EE/8525-16

FIGURE 3-7. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/8525-17

FIGURE 3-8. Write Cycle Timing

3.0 Functional Description (Continued)

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32C016 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

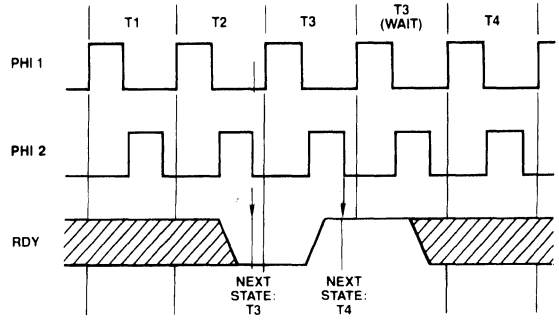
At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If it is sampled low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "wait state." See *Figure 3-9.*

The RDY pin is driven by the NS32C201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1) $\overline{\text{CWAIT}}$ (Continues WAIT), which holds the CPU in WAIT states until removed.
2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
3) $\overline{\text{PER}}$ (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32C201 TCU Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU $\overline{\text{WAITn}}$ pins.



TL/EE/8525–18

FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS32C016 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why it is idle.

Referring to *Figures 3-7* and *3-8,* note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before $\overline{\text{ADS}}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 — The bus is idle because the CPU does not need to perform a bus access.

0001 — The bus is idle because the CPU is executing the WAIT instruction.

0010 — (Reserved for future use.)

0011 — The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 — Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Section 3.4.5.

0101 — Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Section 3.4.5.

0110 — End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Section 3.4.5.

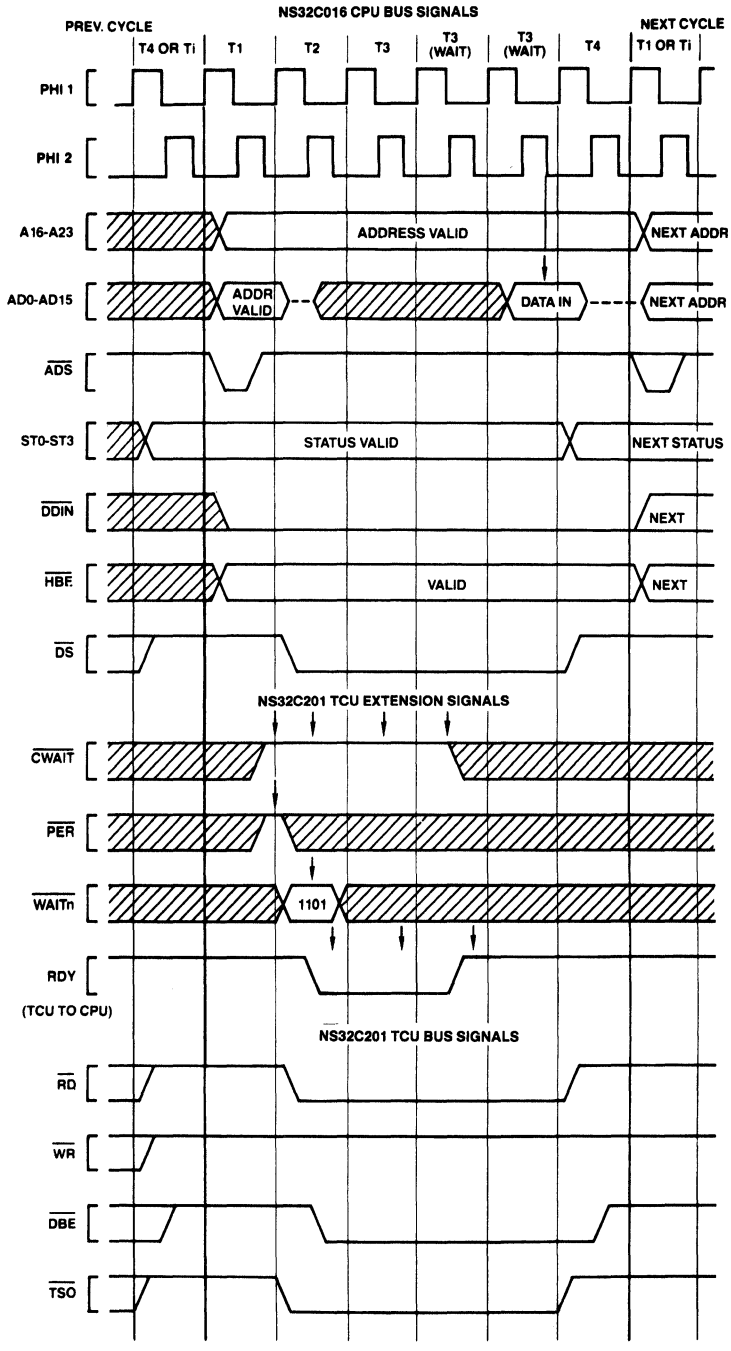
0111 — End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Section 3.4.5.

1000 — Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

3.0 Functional Description (Continued)



TL/EE/8525–19

FIGURE 3-10. Extended Cycle Example

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAITn}}$ indicate points at which the TCU samples. Arrows on AD0–AD15 and RDY indicate points at which the CPU samples.

3.0 Functional Description (Continued)

1001 — Non-Sequential Instruction Fetch.
The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 — Data Transfer.
The CPU is reading or writing an operand of an instruction.

1011 — Read RMW Operand.
The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 — Read for Effective Address Calculation.
The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 — Transfer Slave Processor Operand.
The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Section 3.9.1.

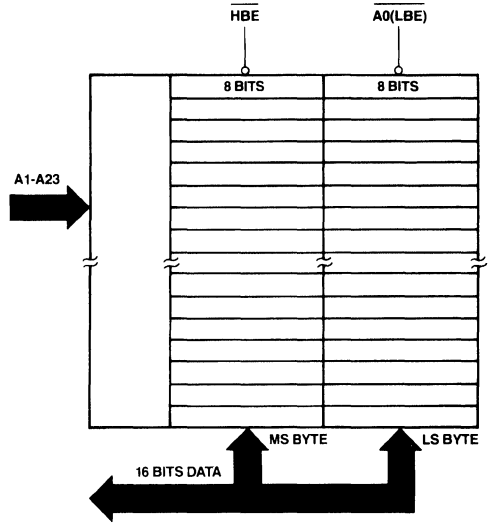
1110 — Read Slave Processor Status.
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Section 3.9.1.

1111 — Broadcast Slave ID.
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Section 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS32C016 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32C016 is that the presence of a 16-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32C016 provides a special control signal, High Byte Enable ($\overline{HBE}$), which facilitates individual byte addressing on a 16-bit bus.

Memory is intended to be organized as two eight-bit banks, each bank receiving the word address (A1–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7, is enabled to respond to even byte addresses; i.e., when the least significant address bit (A0) is low. The other bank, connected to Data Bus pins AD8–AD15, is enabled when $\overline{HBE}$ is low. See *Figure 3-11.*



TL/EE/8525–20

FIGURE 3-11. Memory Interface

Any bus cycle falls into one of three categories: Even Byte Access, Odd Byte Access, and Even Word Access. All accesses to any data type are made up of sequences of these cycles. Table 3-1 gives the state of A0 and $\overline{HBE}$ for each category.

TABLE 3-1. Bus Cycle Categories

Category	$\overline{HBE}$	A0
Even Byte	1	0
Odd Byte	0	1
Even Word	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment (i.e., whether it starts on an even byte address or an odd byte address). Table 3-2 lists the bus cycle performed for each situation. For the timing of A0 and $\overline{HBE}$, see Section 3.4.

3.0 Functional Description (Continued)

TABLE 3-2. Access Sequences

A. Odd Word Access Sequence

Cycle	Type	Address	$\overline{HBE}$	A0	High Bus	Low Bus
					BYTE 1	BYTE 0 ← A
1	Odd Byte	A	0	1	Byte 0	Don't Care
2	Even Byte	A+1	1	0	Don't Care	Byte 1

B. Even Double-Word Access Sequence

Cycle	Type	Address	$\overline{HBE}$	A0	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1	Even Word	A	0	0	Byte 1	Byte 0		
2	Even Word	A+2	0	0	Byte 3	Byte 2		

C. Odd Double-Word Access Sequence

Cycle	Type	Address	$\overline{HBE}$	A0	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1	Odd Byte	A	0	1	Byte 0	Don't Care		
2	Even Word	A+1	0	0	Byte 2	Byte 1		
3	Even Byte	A+3	1	0	Don't Care	Byte 3		

D. Even Quad-Word Access Sequence

Cycle	Type	Address	$\overline{HBE}$	A0	BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1	Even Word	A	0	0	Byte 3	Byte 2	Byte 1	Byte 0				
2	Even Word	A+2	0	0	Byte 7	Byte 6	Byte 5	Byte 4				
Other bus cycles (instruction prefetch or slave) can occur here.												
3	Even Word	A+4	0	0	Byte 3	Byte 2	Byte 1	Byte 0				
4	Even Word	A+6	0	0	Byte 7	Byte 6	Byte 5	Byte 4				

E. Odd Quad-Word Access Sequence

Cycle	Type	Address	$\overline{HBE}$	A0	BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1	Odd Byte	A	0	1	Byte 0	Don't Care						
2	Even Word	A+1	0	0	Byte 2	Byte 1						
3	Even Byte	A+3	1	0	Don't Care	Byte 3						
Other bus cycles (instruction prefetch or slave) can occur here.												
4	Odd Byte	A+4	0	1	Byte 4	Don't Care						
5	Even Word	A+5	0	0	Byte 6	Byte 5						
6	Even Byte	A+7	1	0	Don't Care	Byte 7						

3.0 Functional Description (Continued)

3.4.3.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.3.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

3.4.4 Instruction Fetches

Instructions for the NS32C016 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the eight-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Section 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always Even Word Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle is either an Even Word Read or an Odd Byte Read, depending on whether the destination address is even or odd.

3.4.5 Interrupt Control Cycles

Activating the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0–ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32C016 interrupt structure, see Section 3.8.

3.0 Functional Description (Continued)

TABLE 3-3. Interrupt Sequences

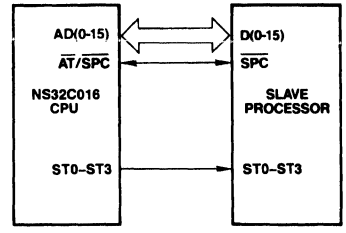
Cycle	Status	Address	$\overline{DDIN}$	$\overline{HBE}$	A0	High Bus	Low Bus
			A. Non-Maskable Interrupt Control Sequences.				
Interrupt Acknowledge							
1	0100	$FFFF00_{16}$	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
			B. Non-Vectored Interrupt Control Sequences.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
			C. Vectored Interrupt Sequences: Non-Cascaded.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Vector: Range: 0–127
Interrupt Return							
1	0110	$FFFE00_{16}$	0	1	0	Don't Care	Vector: Same as in Previous Int. Ack. Cycle
			D. Vectored Interrupt Sequences: Cascaded.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Cascade Index: range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)							
2	0101	Cascade Address	0	1 or 0*	0 or 1*	Vector, range 0–255; on appropriate half of Data Bus for even/odd address	
Interrupt Return							
1	0110	$FFFE00_{16}$	0	1	0	Don't Care	Cascade Index: same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address.)							
2	0111	Cascade Address	0	1 or 0*	0 or 1*	Don't Care	Don't Care

* If the Cascaded ICU Address is Even (A0 is low), then the CPU applies $\overline{HBE}$ high and reads the vector number from bits 0–7 of the Data Bus.
If the address is Odd (A0 is high), then the CPU applies $\overline{HBE}$ low and reads the vector number from bits 8–15 of the Data Bus. The vector number may be in the range 0–255.

3.0 Functional Description (Continued)

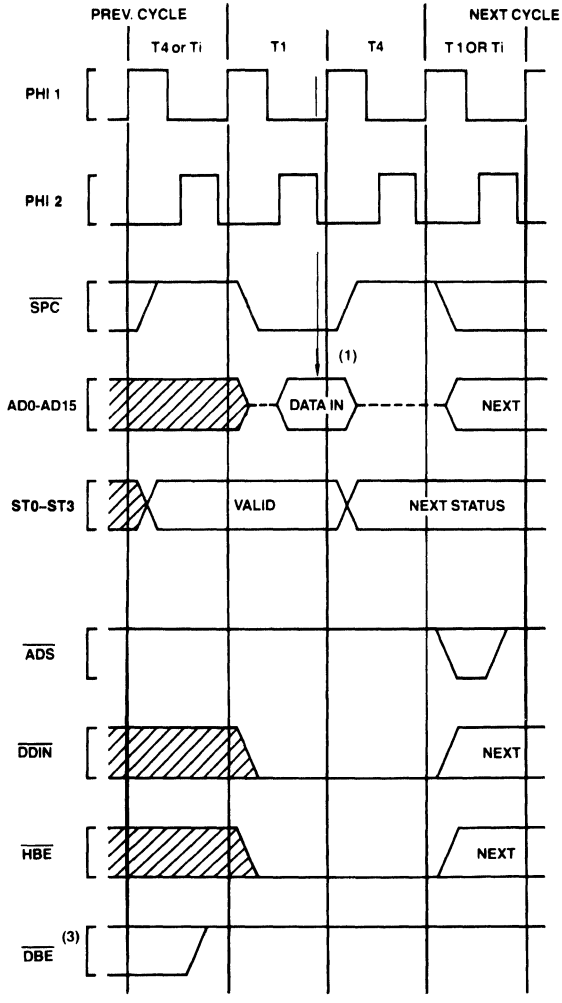
3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Section 3.5.1), the $\overline{AT}/\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{SPC}$). In a Slave Processor bus cycle, data is transferred on the Data Bus (AD0–AD15), and the status lines ST0–ST3 are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Section 3.9 for full protocol sequences.



TL/EE/8525–21

FIGURE 3-12. Slave Processor Connections



TL/EE/8525–22

Notes:

(1) CPU samples Data Bus here.

(2) $\overline{DBE}$ and all other NS32C201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor

3.0 Functional Description (Continued)

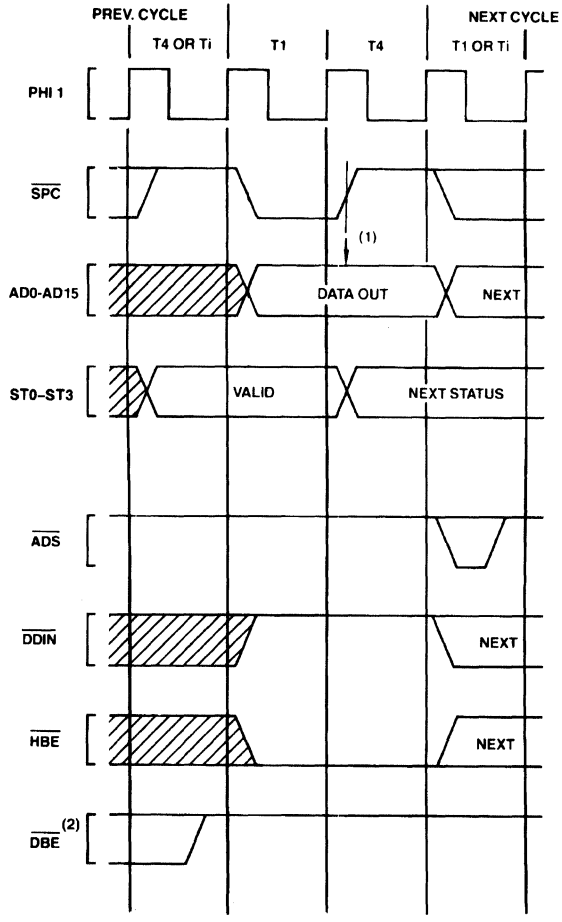
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-13* and *3-14*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ($\overline{ADS}$), no bus signals are generated by the NS32C201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.6.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0–AD7), and a Word operand is transferred on the entire bus. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.



TL/EE/8525–23

Notes:

(1) Slave Processor samples Data Bus here.

(2) $\overline{DBE}$, being provided by the NS32C201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. $\overline{TCU}$ signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32C016 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

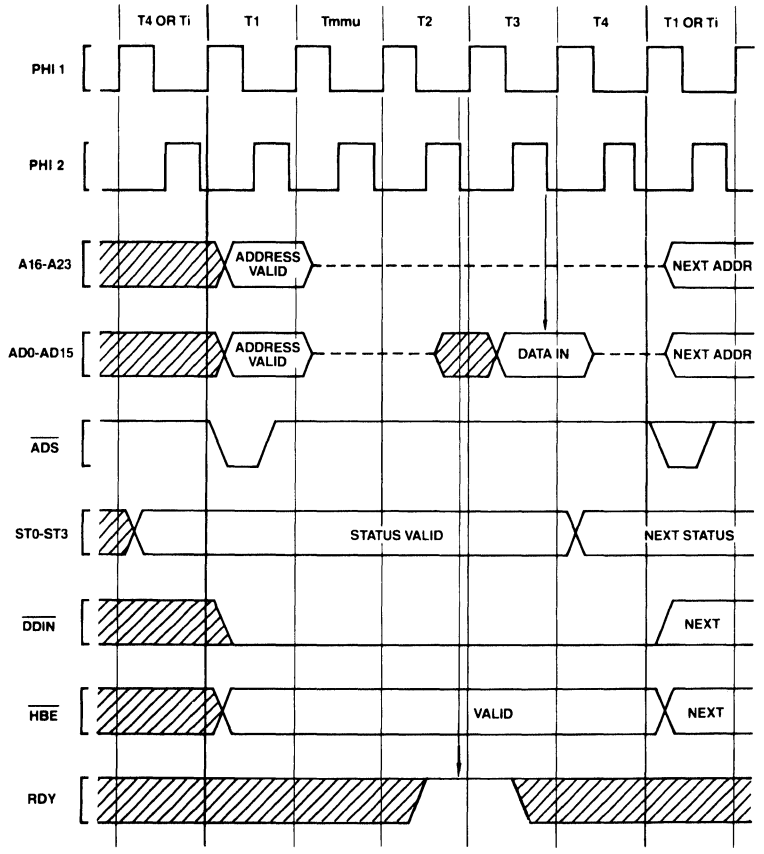
3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32C016 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the $\overline{AT}/\overline{SPC}$ (Address Translation/Slave Processor Control) pin on the rising edge of the $\overline{RST}$ (Reset) pulse. If $\overline{AT}/\overline{SPC}$ is sampled as high, the bus timing is as previously described in Section 3.4. If it is sampled as low, two changes occur:

1) An extra clock cycle, Tmmu, is inserted into all bus cycles except Slave Processor transfers.
2) The $\overline{DS}/\overline{FLT}$ pin changes in function from a Data Strobe output ($\overline{DS}$) to a Float Command input ($\overline{FLT}$).

The NS32082 MMU will itself pull the CPU $\overline{AT}/\overline{SPC}$ pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the



TL/EE/8525–24

FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

3.0 Functional Description (Continued)

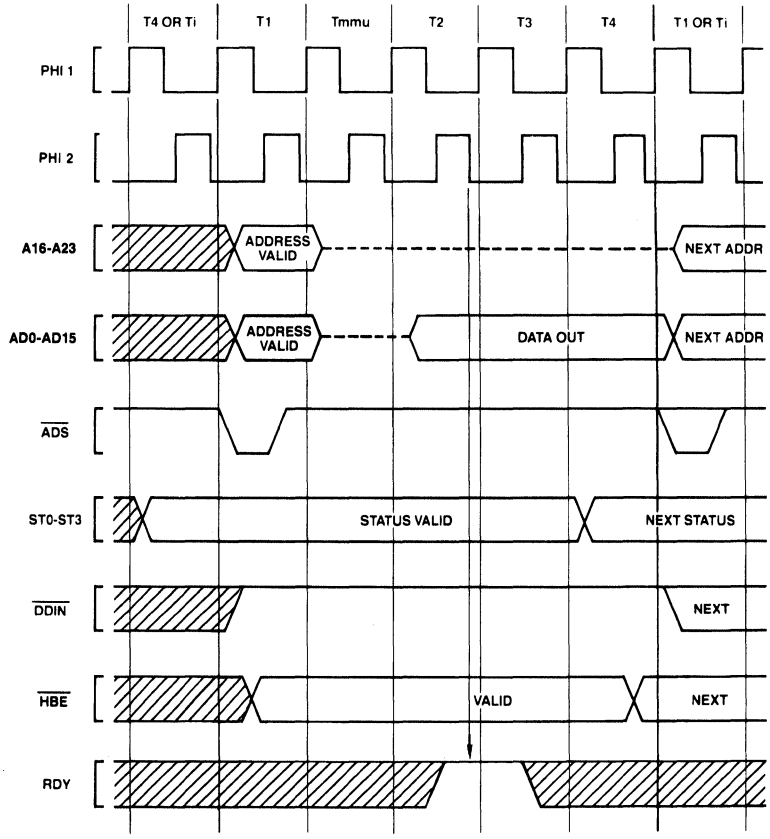
presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Section 2.1.3.

3.5.2 Translated Bus Timing

Figures 3-15 and *3-16* illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0–AD15 and A16–A23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{PAV}$. T2 through T4 of the cycle are identical to their counter-parts without Address Translation, with the exception that the CPU Address lines A16–A23 remain in the TRI-STATE condition. This allows the MMU to continue asserting the translated address on those pins.

Note that in order for the NS32082 MMU to operate correctly, it must be set to the 32C016 mode by forcing A24 high during reset.

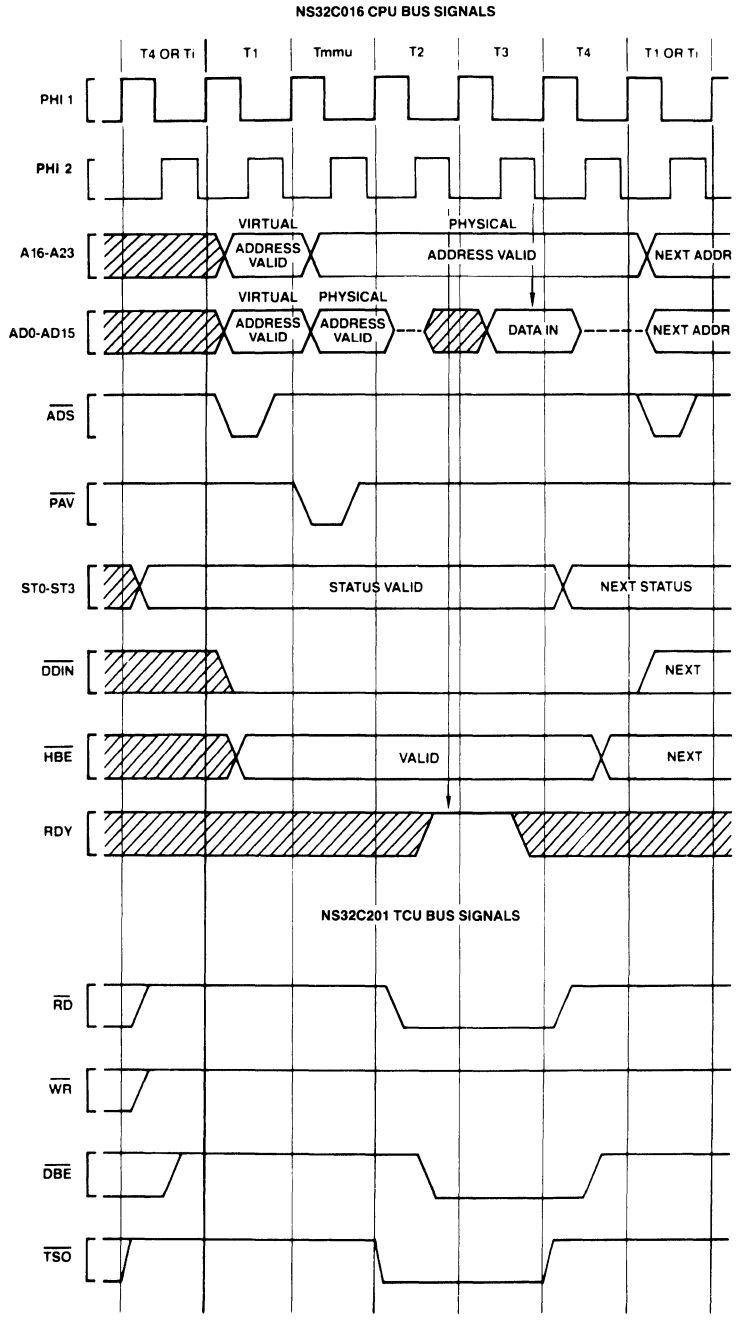
Figures 3-17 and *3-18* show a Read cycle and a Write cycle as generated by the 32C016/32082/32C201 group. Note that with the CPU $\overline{ADS}$ signal going only to the MMU, and with the MMU $\overline{PAV}$ signal substituting for $\overline{ADS}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.



TL/EE/8525–25

FIGURE 3-16. Write Cycle with Address Translation (CPU Action)

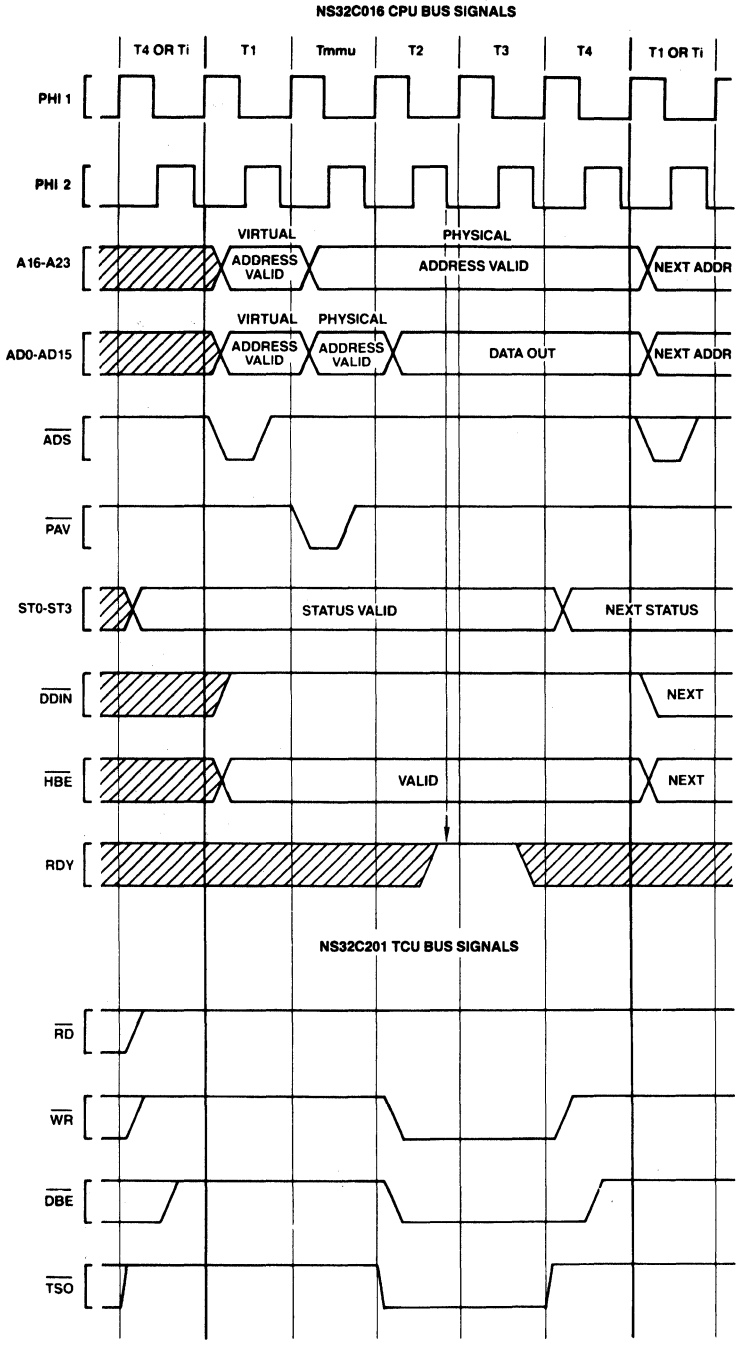
3.0 Functional Description (Continued)



TL/EE/8525–26

FIGURE 3-17. Memory-Managed Read Cycle

3.0 Functional Description (Continued)



TL/EE/8525–27

FIGURE 3-18. Memory-Managed Write Cycle

3.0 Functional Description (Continued)

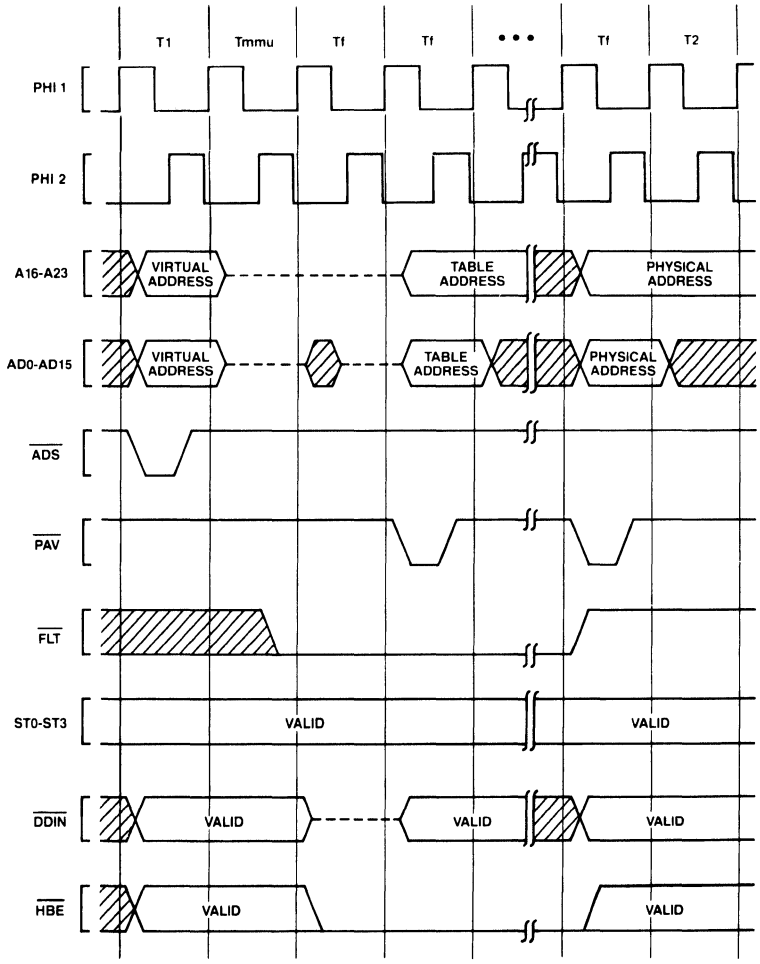
3.5.3 The $\overline{FLT}$ (Float) Pin

The $\overline{FLT}$ pin is used by the CPU for address translation support. Activating $\overline{FLT}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its internal translation Look-Aside Buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effects of $\overline{FLT}$. Upon sampling $\overline{FLT}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD15, A16–A23 and $\overline{DDIN}$ to the TRI-STATE condition ("floating").
2) Sets $\overline{HBE}$ low.
3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{RST}$/$\overline{ABT}$ description, Section 3.5.4.)

Note that the AD0–AD15 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until $\overline{FLT}$ again goes high. See the Timing Specifications, Section 4.



TL/EE/8525-28

FIGURE 3-19. $\overline{FLT}$ Timing

3.0 Functional Description (Continued)

3.5.4 Aborting Bus Cycles

The $\overline{RST}/\overline{ABT}$ pin, apart from its Reset function (Section 3.3), also serves as the means to "abort," or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the $\overline{RST}/\overline{ABT}$ pin is held active for only one clock cycle.

If $\overline{RST}/\overline{ABT}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then Ti, thereby terminating the cycle. Since it is the MMU $\overline{PAV}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The reference page must be brought into physical memory from mass storage to make it accessible to the CPU.
2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Section 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

3.5.4.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the NS32082 Memory Management Unit.

1) If $\overline{FLT}$ has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, *Figure 4-23.*
2) If $\overline{FLT}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{FLT}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{FLT}$ is removed. See *Figure 4-24.*
3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management funtions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If $\overline{RST}/\overline{ABT}$ is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

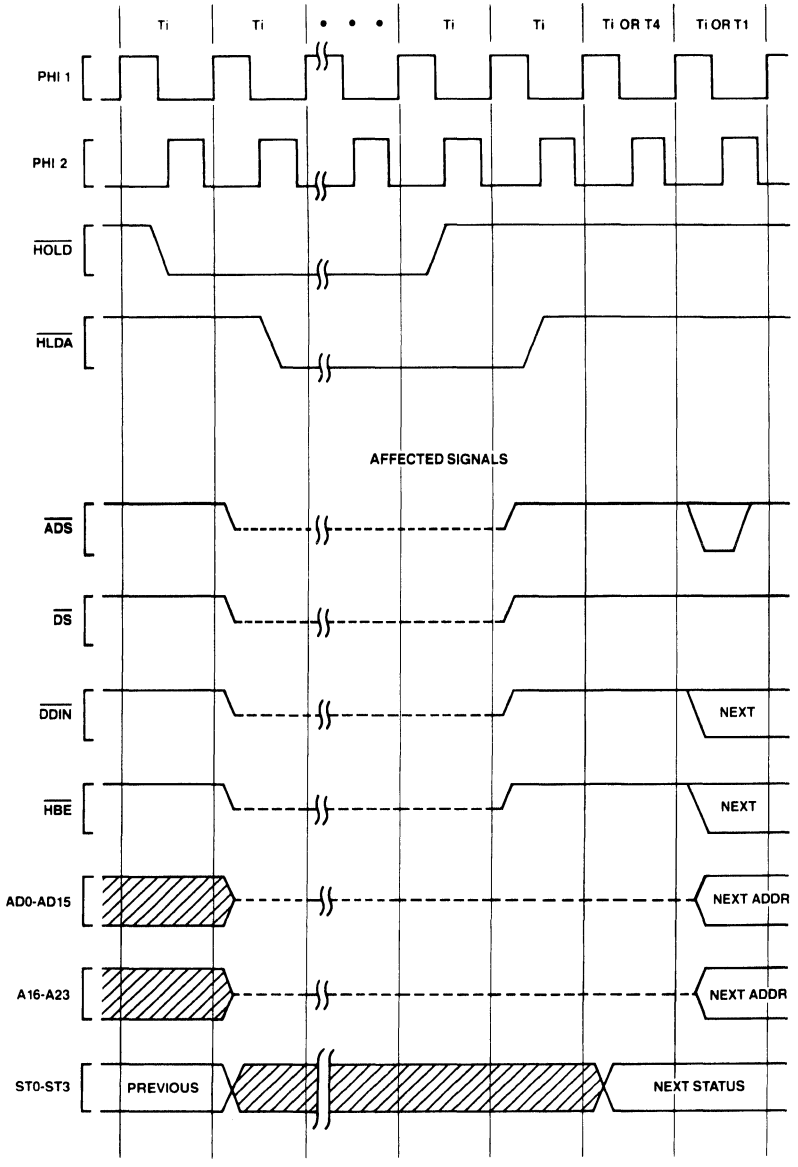
3.6 BUS ACCESS CONTROL

The NS32C016 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{HOLD}$ (Hold Request) and $\overline{HLDA}$ (Hold Acknowledge) pins. By asserting $\overline{HOLD}$ low, an external device requests access to the bus. On receipt of $\overline{HLDA}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD15, A16–A23, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{HBE}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{HOLD}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{HLDA}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{HOLD}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-20* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-21* shows the sequence if the CPU is using the bus at the time that the $\overline{HOLD}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{HLDA}$ signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.

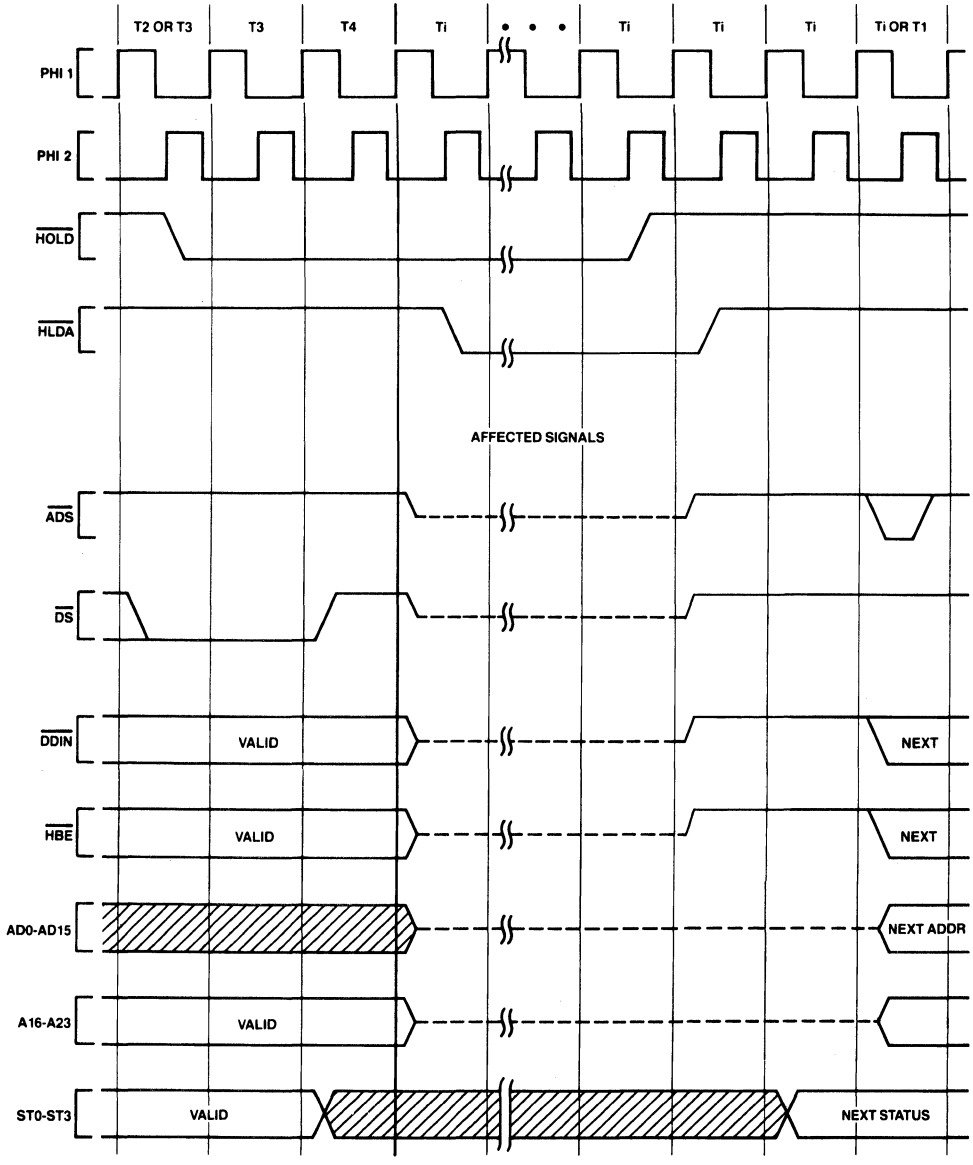
3.0 Functional Description (Continued)



TL/EE/8525–29

FIGURE 3-20. HOLD Timing, Bus Initially Idle

3.0 Functional Description (Continued)



TL/EE/8525–30

FIGURE 3-21. HOLD Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32C016 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/$\overline{S}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-22.*

$\overline{ILO}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing. As with the U/$\overline{S}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, *Figure 4-20.*

3.8 NS32C016 INTERRUPT STRUCTURE

$\overline{INT}$, on which maskable interrupts may be requested,

$\overline{NMI}$, on which non-maskable interrupts may be requested, and

$\overline{RST}$/$\overline{ABT}$, which may be used to abort a bus cycle and any associated instruction. See Section 3.5.4.

In addition, there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

1) Adjustment of Registers.

   Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2) Saving Processor Status.

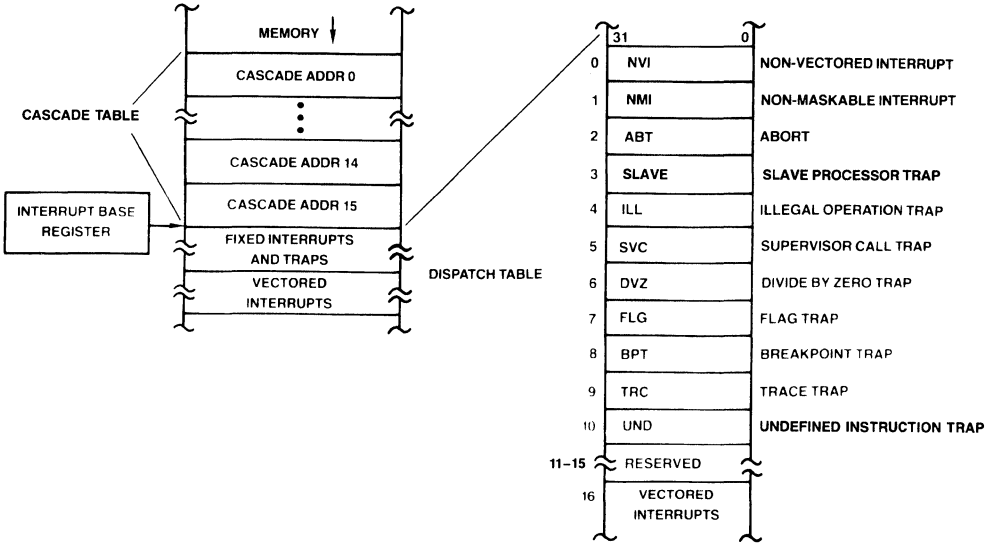
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

3) Vector Acquisition.

   A Vector is either obtained from the Data Bus or is supplied by default.

4) Service Call.

   The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-22.* A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.



TL/EE/8525–31

FIGURE 3-22. Interrupt Dispatch and Cascade Tables

3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-23,* from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Section 3.8.7.1.
Abort Interrupt:	Section 3.8.7.4.
Traps (except Trace):	Section 3.8.7.2.
Trace Trap:	Section 3.8.7.3.

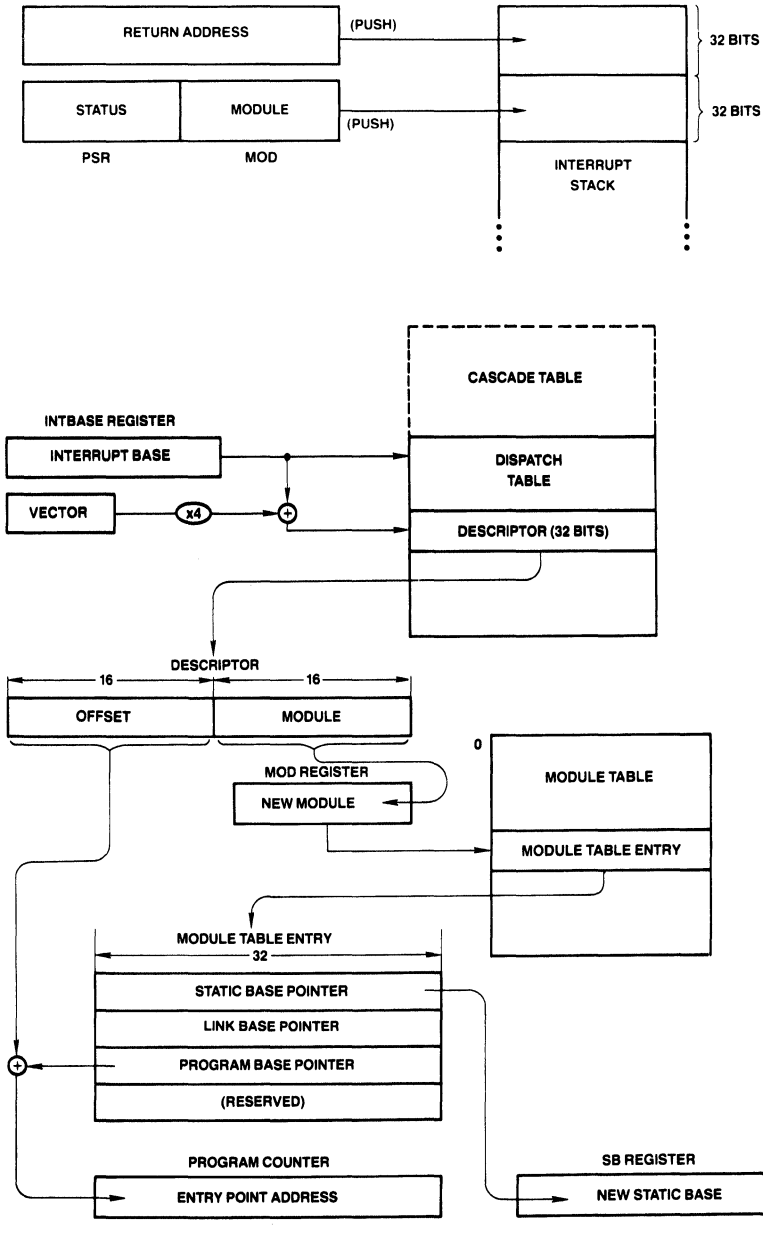


FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction (*Figure 3-24)* restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-25.*

3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$, $\overline{\text{NMI}}$ or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = 0) or Vectored (bit I = 1).

3.8.3.1 Non-Vectored Mode

In the Non-Vectored mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.

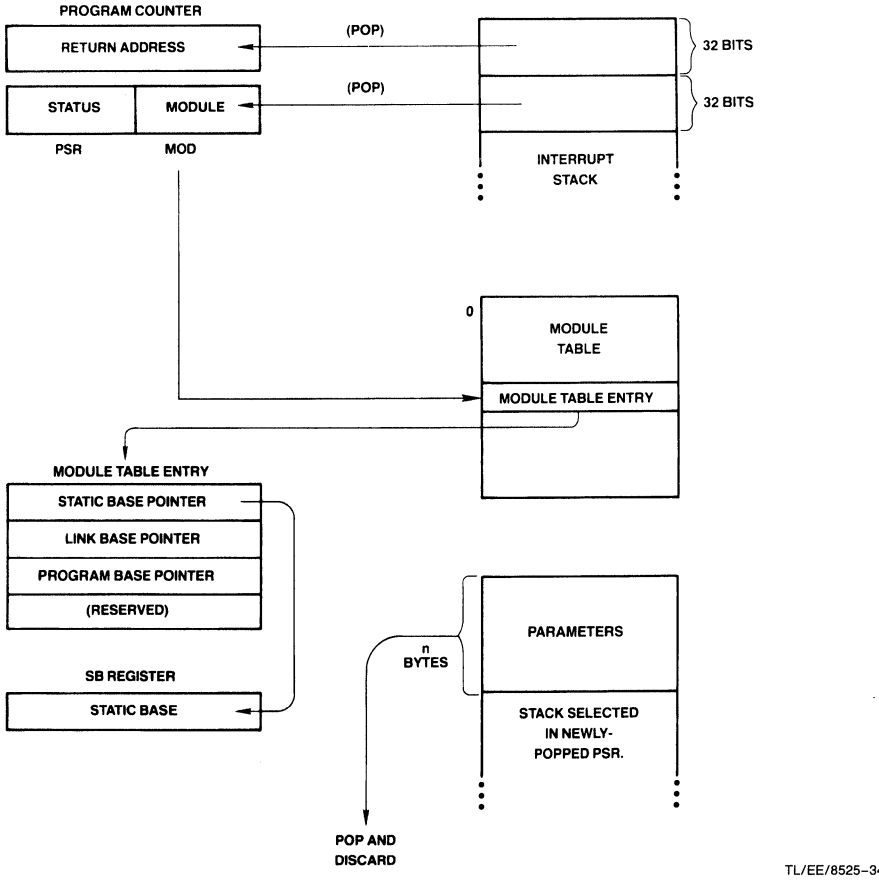
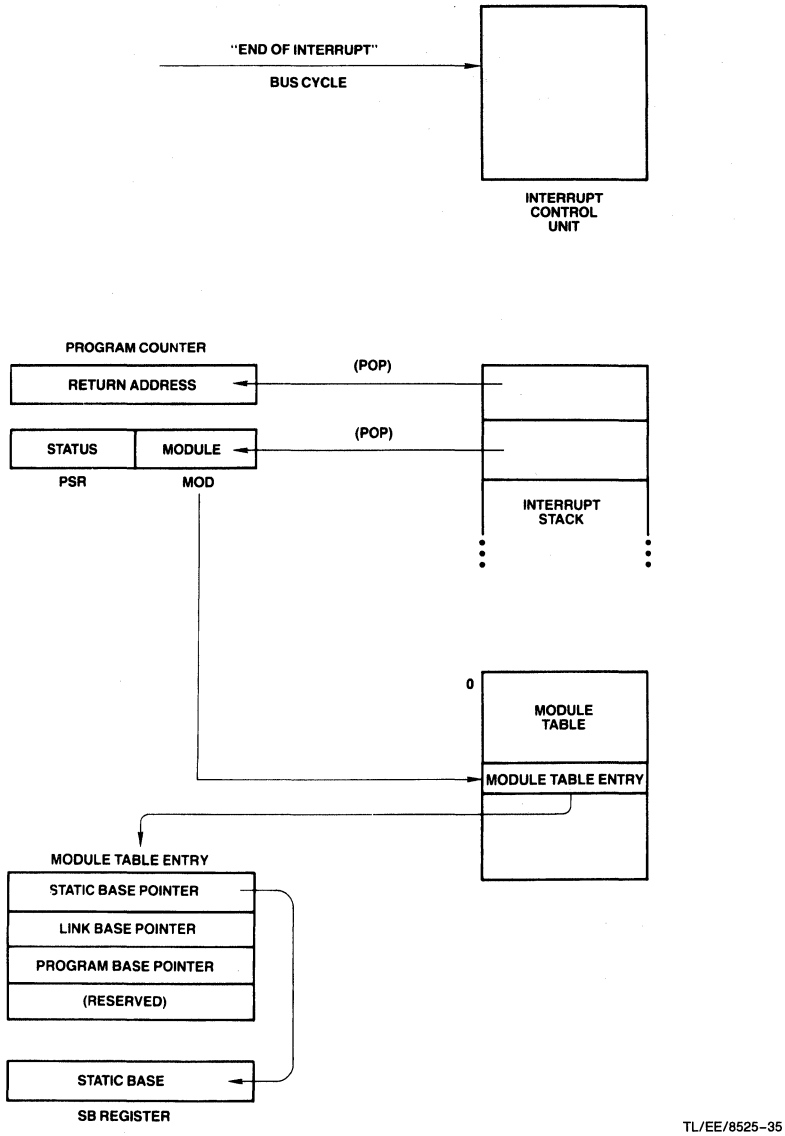


FIGURE 3-24. Return from Trap (RETT n) Instruction Flow

3.0 Functional Description (Continued)



TL/EE/8525–35

FIGURE 3-25. Return from Interrupt (RET I) Instruction Flow

3.0 Functional Description (Continued)

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.8.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-27* shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{INT}$ pin.

In a system which uses cascading, two tasks must be performed upon initialization:

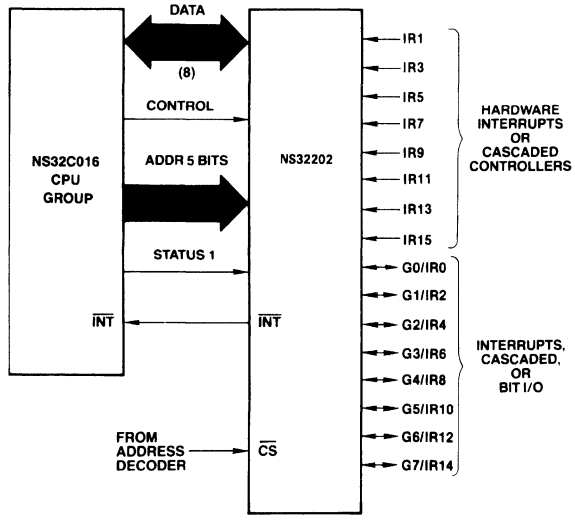
1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.
2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-22 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range −16 to −1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Section 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Section 3.4.2), whereupon the Master ICU again provides the negative Cascaded Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

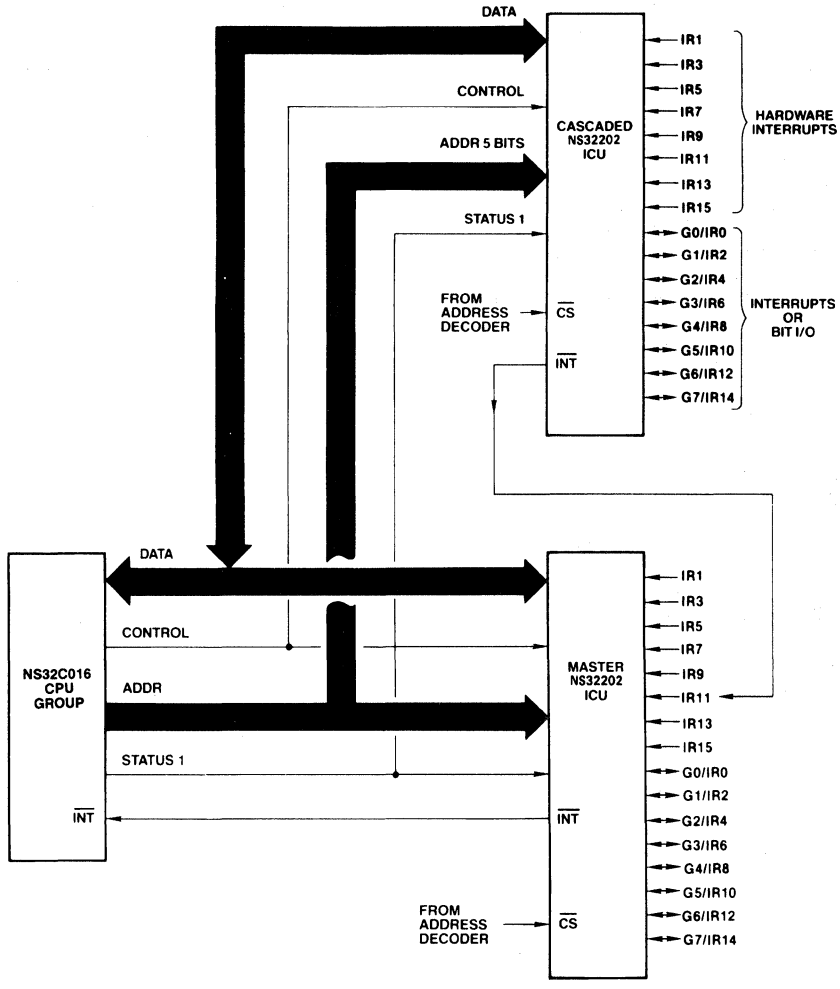
Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the Interrupt Mask Register of the Interrupt Controller. However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.



TL/EE/8525–36

FIGURE 3-26. Interrupt Control Unit Connections (16 Levels)

3.0 Functional Description (Continued)



TL/EE/8525–37

FIGURE 3-27. Cascaded Interrupt Control Unit Connections

3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the $\overline{\text{NMI}}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is $FFFF00_{16}$. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Section 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) below is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by NS32C016 CPU are:

Trap (SLAVE): An exceptional condition was detected by the Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Section 3.9.1).

3.0 Functional Description (Continued)

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

3.8.6 Prioritization

The NS32C016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1) Traps other than Trace (Highest priority)
2) Abort
3) Non-Maskable Interrupt
4) Maskable Interrupts
5) Trace Trap (Lowest priority)

3.8.7 Interrupt/Trap Sequences: Detail Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-28.* Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequenced followed in processing either Maskable or Non-Maskable Interrupts (on the $\overline{INT}$ or $\overline{NMI}$ pins, respectively), see Section 3.8.7.1. For Abort interrupts, see Section 3.8.7.4. For the Trace Trap, see Section 3.8.7.3, and for all other traps see Section 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{NMI}$ pin receives a falling edge, or the $\overline{INT}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE + 4* Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Section 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-28.*

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector*4 + INTBASE Register contents.
2) Move the Module field of the Descriptor into the MOD Register.
3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
4) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.
5) Flush Queue: Non-sequentially fetch first instruction of Interrupt Routine.
6) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-28. Service Sequence
Invoked during all interrupt/trap sequences

3.0 Functional Description (Continued)

3.8.7.2 Trap Sequence: Traps Other Than Trace

1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2) Set "Vector" to the value corresponding to the trap type.

   SLAVE: Vector = 3.
   ILL: Vector = 4.
   SVC: Vector = 5.
   DVZ: Vector = 6.
   FLG: Vector = 7.
   BPT: Vector = 8.
   UND: Vector = 10.
3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Return Address" to the address of the first byte of the trapped instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

3.8.7.3 Trace Trap Sequence

1) In the Processor Status Register (PSR), clear the P bit.
2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4) Set "Vector" to 9.
5) Set "Return Address" to the address of the next instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

3.8.7.4 Abort Sequence

1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
2) Clear the PSR P bit.
3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Vector" to 2.
6) Set "Return Address" to the address of the first byte of the aborted instruction.
7) Perform Service (Vector, Return Address), *Figure 3-28.*

3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32C016 CPU recognizes three groups of instructions as being executable by external Slave Processors:

Floating Point Instruction Set
Memory Management Instruction Set
Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Section 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

3.9.1 Slave Processor Protocol

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.
2) It specifies which Slave Processor will execute it.
3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-29.* While applying Status Code 1111 (Broadcast ID, Section 3.4.2), the CPU transfers the ID Byte on the least-significant half of the Data Bus (AD0–AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Section 3.4.2).

Status Combinations:
Send ID (ID): Code 1111
Xfer Operand (OP): Code 1101
Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operation Word.
3	OP	CPU Sends Required Operands.
4	—	Slave Starts Execution. CPU Pre-Fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-29. Slave Processor Protocol

3.0 Functional Description (Continued)

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this, and for the Address Translation strap function, $\overline{AT}/\overline{SPC}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Section 3.4.2).

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Section 3.4.2). This word has the format shown in *Figure 3-30.* If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the Slave vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.9.2 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Series 32000 Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-30).*

TABLE 3-4. Floating Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note:

D = Double Word

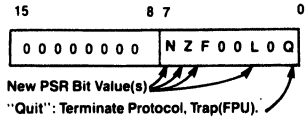
i = integer size (B,W,D) specified in mnemonic.

c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)



TL/EE/8525–38

FIGURE 3-30. Slave Processor Status Word Format

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

3.9.3 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Series 32000 Instruction Set Reference Manual and the NS32082 MMU Data Sheet.

TABLE 3-5. Memory Management Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Series 32000 Instruction Set Reference Manual and the NS32082 Memory Management Unit Data Sheet.

D = Double Word

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)

3.9.4 Custom Slave Instructions

Provided in the NS32C016 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type 'c' will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

TABLE 3-6. Custom Slave Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op. 2	none
CCMP0c	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	readi	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

Notes:

D = Double Word

i = integer size (B,W,D) specified in mnemonic.

f = Floating Point type (F,L) specified in mnemonic.

N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 NS32C016 PIN DESCRIPTIONS

The following is a brief description of all NS32C016 pins. The descriptions reference portions of the Functional Description, Section 3.

4.1.1 Supplies

Logic Power (V_{CCL}): +5V positive supply for on-chip logic. Section 3.1.

Buffer Power (V_{CCB}): +5V positive supply for on-chip output buffers. Section 3.1.

Logic Ground (GNDL): Ground reference for on-chip logic. Section 3.1.

Buffer Ground (GNDB): Ground reference for on-chip drivers connected to output pins. Section 3.1.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Section 3.2.

Ready (RDY): Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Section 3.4.1.

Hold Request ($\overline{HOLD}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.6.

Note: If the $\overline{HOLD}$ signal is generated asynchronously, its set up and hold times may be violated. In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{HLDA}$ latency. This is to avoid speed degradations in cases of heavy $\overline{HOLD}$ activity (i.e. DMA controller cycles interleaved with CPU cycles).

Interrupt ($\overline{INT}$): Active low. Maskable Interrupt request. Section 3.8.

Non-Maskable Interrupt ($\overline{NMI}$): Active low. Non-Maskable Interrupt request. Section 3.8.

Reset/Abort ($\overline{RST}/\overline{ABT}$): Active low. If held active for one clock cycle and released, this pin causes an Abort Command, Section 3.5.4. If held longer, it initiates a Reset, Section 3.3.

4.1.3 Output Signals

Address Bits 16–23 (A16–A23): These are the most significant 8 bits of the memory address bus. Section 3.4.

Address Strobe ($\overline{ADS}$): Active low. Controls address latches; indicates start of a bus cycle. Section 3.4.

Data Direction In ($\overline{DDIN}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Section 3.4.

High Byte Enable ($\overline{HBE}$): Active low. Status signal enabling transfer on the most significant byte of the Data Bus. Section 3.4; Section 3.4.3.

Note: In the current NS32C016, the $\overline{HBE}$ signal is forced low by the CPU when $\overline{FLT}$ is asserted by the MMU. However, in future revisions of the CPU, $\overline{HBE}$ will no longer be affected by $\overline{FLT}$. Therefore, in a memory managed system, an external 'AND' gate is required. This is shown in *Figure B-1* in Appendix B.

Status (ST0–ST3): Active high. Bus cycle status code, ST0 least significant. Section 3.4.2. Encodings are:

0000—Idle: CPU Inactive on Bus.
0001—Idle: WAIT Instruction.
0010—(Reserved)
0011—Idle: Waiting for Slave.
0100—Interrupt Acknowledge, Master.
0101—Interrupt Acknowledge, Cascaded.
0110—End of Interrupt, Master.
0111—End of Interrupt, Cascaded.
1000—Sequential Instruction Fetch.
1001—Non-Sequential Instruction Fetch.
1010—Data Transfer.
1011—Read Read-Modify-Write Operand.
1100—Read for Effective Address.
1101—Transfer Slave Operand.
1110—Read Slave Status Word.
1111—Broadcast Slave ID.

Hold Acknowledge ($\overline{HLDA}$): Active low. Applied by the CPU in response to $\overline{HOLD}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Section 3.6.

User/Supervisor (U/$\overline{S}$): User or Supervisor Mode status. Section 3.7. High state indicates User Mode, low indicates Supervisor Mode. Section 3.7.

Interlocked Operation ($\overline{ILO}$): Active low. Indicates that an interlocked instruction is being executed. Section 3.7.

Program Flow Status ($\overline{PFS}$): Active Low. Pulse indicates beginning of an instruction execution. Section 3.7.

4.1.4 Input-Output Signals

Address/Data 0–15 (AD0–AD15): Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Section 3.4.

Address Translation/Slave Processor Control ($\overline{AT}/\overline{SPC}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of a slave instruction. Section 3.4.6; Section 3.9. Sampled on the rising edge of Reset as Address Translation Strap. Section 3.5.1.

In non-memory-managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Data Strobe/Float ($\overline{DS}/\overline{FLT}$): Active low. Data Strobe output, Section 3.4, or Float Command input, Section 3.5.3. Pin function is selected on $\overline{AT}/\overline{SPC}$ pin, Section 3.5.1.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7V
Power Dissipation	1.5 Watt

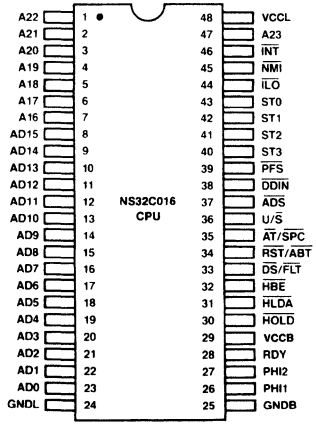
Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS: $T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	0.90 V_{CC}		$V_{CC} + 0.5$	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.10 V_{CC}	V
V_{CRT}	Clock Input Ringing Tolerance	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\ \mu A$	0.90 V_{CC}			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2$ mA			0.10 V_{CC}	V
I_{ILS}	$\overline{AT}/\overline{SPC}$ Input Current (low)	$V_{IN} = 0.4V$, $\overline{AT}/\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current Output Pins in TRI-STATE condition	$0.4 \le V_{OUT} \le V_{CC}$	−20		20	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$			70	mA

Connection Diagram

Dual-In-Line Package



TL/EE/8525-2

Top View

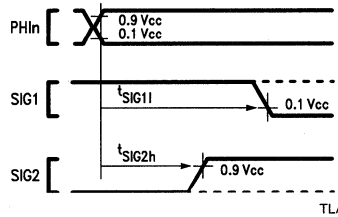
FIGURE 4-1

4.0 Device Specifications (Continued)

4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 15% or 85% of V_{CC} on the rising or falling edges of the clock phases PHI1 and PHI2, and all output signals; to 30% or 70% of V_{CC} on all the CMOS input signals, and to 0.8V or 2.0V on all the TTL input signals as illustrated in *Figures 4-2* and *4-3* unless specifically stated otherwise.



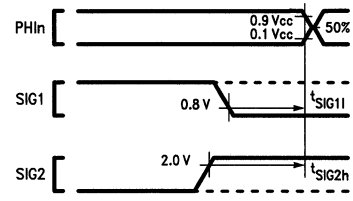
TL/EE/8525–39

FIGURE 4-2. Timing Specification Standard (CMOS Output Signals)

ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge

T.E. — trailing edge F.E. — falling edge



TL/EE/8525–40

FIGURE 4-3. Timing Specification Standard (TTL Input Signals)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32C016-6, NS32C016-10 and NS32C016-15

Maximum times assume capacitive loading of 100 pF, on the address/data bus signals and 50 pF on all other signals.

Name	Figure	Description	Reference/Conditions	NS32C016-6		NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4-4	Address bits 0–15 valid	after R.E., PHI1 T1		80		50		35	ns
t_{ALh}	4-4	Address bits 0–15 hold	after R.E., PHI1 Tmmu or T2	5		5		5		ns
t_{Dv}	4-4	Data valid (write cycle)	after R.E., PHI1 T2		80		50		38	ns
t_{Dh}	4-4	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{AHv}	4-4	Address bits 16–23 valid	after R.E., PHI1 T1		85		50		35	ns
t_{AHh}	4-4	Address bits 16–23 hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	4-5	Address bits 0–15 set up	before $\overline{ADS}$ T.E.	25		25		20		ns
t_{AHADSs}	4-5	Address bits 16–23 set up	before $\overline{ADS}$ T.E.	25		25		20		ns
t_{ALADSh}	4-9	Address bits 0–15 hold	after $\overline{ADS}$ T.E.	10		10		6		ns
t_{AHADSh}	4-9	Address bits 16–23 hold	after $\overline{ADS}$ T.E.	10		10		6		ns
t_{ALf}	4-5	Address bits 0–15 floating	after R.E., PHI1 T2 (no MMU)		25		25		18	ns
t_{ALMf}	4-9	Address bits 0–15 floating	after R.E., PHI1 Tmmu (with MMU)		25		25		23	ns
t_{AHMf}	4-9	Address bits 16–23 floating	after R.E., PHI1 Tmmu (with MMU)		25		25		23	ns
t_{HBEv}	4-4	$\overline{HBE}$ signal valid	after R.E., PHI1 T1		95		70		38	ns
t_{HBEh}	4-4	$\overline{HBE}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{STv}	4-4	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		75		45		38	ns
t_{STh}	4-4	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	4-5	$\overline{DDIN}$ signal valid	after R.E., PHI1 T1		110		65		38	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32C016-6, NS32C016-10 and NS32C016-15 (Continued)

Name	Figure	Description	Reference/Conditions	NS32C016-6		NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	Min	Max	
t_{DDINh}	4-5	$\overline{DDIN}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	4-4	$\overline{ADS}$ signal active (low)	after R.E., PHI1 T1		55		35		26	ns
t_{ADSia}	4-4	$\overline{ADS}$ signal inactive	after R.E., PHI2 T1		60		45		34	ns
t_{ADSw}	4-4	$\overline{ADS}$ pulse width	at 10% V_{CC} (both edges)	50		30		25		ns
t_{DSa}	4-4	$\overline{DS}$ signal active (low)	after R.E., PHI1 T2		70		45		30	ns
t_{DSia}	4-4	$\overline{DS}$ signal inactive	after R.E., PHI1 T4		60		40		30	ns
t_{ALf}	4-6	AD0–AD15 floating	after R.E., PHI1 T1 (caused by $\overline{HOLD}$)		40		25		18	ns
t_{AHf}	4-6	A16–A23 floating	after R.E., PHI1 T1 (caused by $\overline{HOLD}$)		40		25		18	ns
t_{DSf}	4-6	$\overline{DS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{ADSf}	4-6	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{HBEf}	4-6	$\overline{HBE}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{DDINf}	4-6	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{HLDAa}	4-6	$\overline{HLDA}$ signal active (low)	after R.E., PHI1 Ti		100		75		23	ns
t_{HLDAia}	4-8	$\overline{HLDA}$ signal inactive	after R.E., PHI1 Ti		100		75		23	ns
t_{DSr}	4-8	$\overline{DS}$ signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		90		55		40	ns
t_{ADSr}	4-8	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{HBEr}	4-8	$\overline{HBE}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{DDINr}	4-8	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		90		55		40	ns
t_{DDINf}	4-9	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		80		50		22	ns
t_{HBEl}	4-9	$\overline{HBE}$ signal low (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		100		65		22	ns
t_{DDINr}	4-10	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		75		50		22	ns
t_{HBEr}	4-10	$\overline{HBE}$ signal returns from low (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		90		75		22	ns
t_{SPCa}	4-13	$\overline{SPC}$ output active (low)	after R.E., PHI1 T1		50		35		26	ns
t_{SPCia}	4-13	$\overline{SPC}$ output inactive	after R.E., PHI1 T4		50		35		26	ns
t_{SPCnf}	4-15	$\overline{SPC}$ output nonforcing	after R.E., PHI2 T4		20		10		8	ns
t_{Dv}	4-13	Data valid (slave processor write)	after R.E., PHI1 T1		80		50		38	ns
t_{Dh}	4-13	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	4-18	$\overline{PFS}$ pulse width	at 10% V_{CC} (both edges)	70		70		45		ns
t_{PFSa}	4-18	$\overline{PFS}$ pulse active (low)	after R.E., PHI2		70		50		38	ns
t_{PFSia}	4-18	$\overline{PFS}$ pulse inactive	after R.E., PHI2		70		50		38	ns
t_{ILOs}	4-20a	$\overline{ILO}$ signal setup	before R.E., PHI1 T1 of first interlocked write cycle	30		30		30		ns
t_{ILOh}	4-20b	$\overline{ILO}$ signal hold	after R.E., PHI1 T3 of last interlocked read cycle	10		10		7		ns
t_{ILOa}	4-21	$\overline{ILO}$ signal active (low)	after R.E., PHI1		70		55		35	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32C016-6, NS32C016-10 and NS32C016-15 (Continued)

Name	Figure	Description	Reference/Conditions	NS32C016-6		NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	Min	Max	
t_{ILOia}	4-21	$\overline{ILO}$ signal inactive	after R.E., PHI1		70		55		35	ns
t_{USv}	4-22	U/$\overline{S}$ signal valid	after R.E., PHI1 T4		70		45		30	ns
t_{USh}	4-22	U/$\overline{S}$ signal hold	after R.E., PHI1 T4	10		10		6		ns
t_{NSPF}	4-19b	Nonsequential fetch to next $\overline{PFS}$ clock cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	4-19a	$\overline{PFS}$ clock cycle to next nonsequential fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	4-29	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t_{Cp}

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

4.4.2.2 Input Signal Requirements: NS32C016-6, NS32C016-10 and NS32C016-15

Name	Figure	Description	Reference/Conditions	NS32C016-6		NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	Min	Max	
t_{PWR}	4-25	Power stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		33		μs
t_{DIs}	4-5	Data in setup (read cycle)	before F.E., PHI2 T3	20		10		10		ns
t_{DIh}	4-5	Data in hold (read cycle)	after F.E., PHI1 T4	10		10		6		ns
t_{HLDa}	4-6	$\overline{HOLD}$ active (low) setup time (see note)	before F.E., PHI2 TX1	25		25		17		ns
t_{HLDia}	4-8	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25		25		17		ns
t_{HLDh}	4-6	$\overline{HOLD}$ hold time	after R.E., PHI1 TX2	0		0		0		ns
t_{FLTa}	4-9	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25		25		17		ns
t_{FLTia}	4-10	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T2	25		25		17		ns
t_{RDYs}	4-11, 4-12	RDY setup time	before F.E., PHI2 T2 or T3	25		15		10		ns
t_{RDYh}	4-11, 4-12	RDY hold time	after F.E., PHI1 T3	0		0		0		ns
t_{ABTs}	4-23	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 Tmmu	30		20		13		ns
t_{ABTs}	4-24	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 Tf	30		20		13		ns
t_{ABTh}	4-23	$\overline{ABT}$ hold time	after R.E., PHI1	0		0		0		ns
t_{RSTs}	4-25, 4-26	$\overline{RST}$ setup time	before F.E., PHI1	20		10		8		ns
t_{RSTw}	4-26	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	4-27	$\overline{INT}$ setup time	before R.E., PHI1	20		20		15		ns
t_{NMIw}	4-28	$\overline{NMI}$ pulse width	at 0.8V (both edges)	70		70		50		ns
t_{DIs}	4-14	Data setup (slave read cycle)	before F.E., PHI2 T1	20		10		10		ns
t_{DIh}	4-14	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		7		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements: NS32C016-6, NS32C016-10 and NS32C016-15 (Continued)

Name	Figure	Description	Reference/Conditions	NS32C016-6		NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	Min	Max	
t_{SPCd}	4-15	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	17		13		10		ns
t_{SPCs}	4-15	$\overline{SPC}$ setup time	before F.E., PHI1	42		32		25		ns
t_{SPCw}	4-15	$\overline{SPC}$ pulse width from slave processor (async. input)	at 0.8V (both edges)	30		20		20		ns
t_{ATs}	4-16	$\overline{AT}/\overline{SPC}$ setup for address translation strap	before R.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1		1		1		t_{Cp}
t_{ATh}	4-16	$\overline{AT}/\overline{SPC}$ hold for address translation strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		t_{Cp}

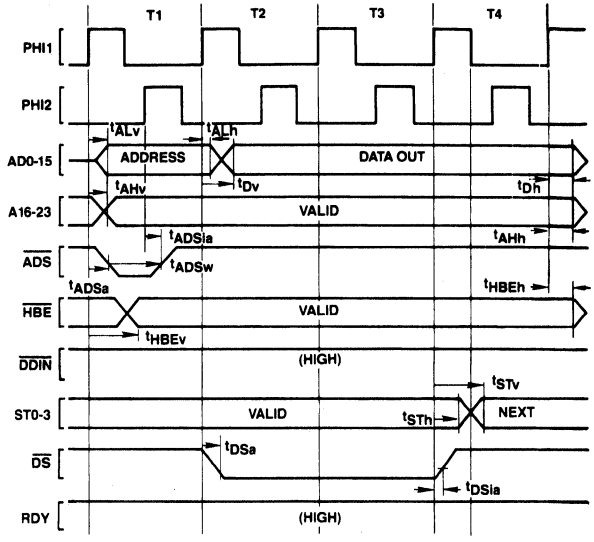
Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU cff the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

4.4.2.3 Clocking Requirements: NS32C016-6, NS32C016-10 and NS32C016-15

Name	Figure	Description	Reference/Conditions	NS32C016-6		NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4-17	PHI1, PHI2 rise time	10% to 90% V_{CC} on R.E., PHI1, PHI2		9		7		5	ns
t_{CLf}	4-17	PHI1, PHI2 fall time	90% to 10% V_{CC} on F.E., PHI1, PHI2		9		7		5	ns
t_{Cp}	4-17	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	160	2000	100	2000	66	2000	ns
$t_{CLw(1,2)}$	4-17	PHI1, PHI2 pulse width	At 2.0V on PHI1, PHI2 (both edges)	$0.5t_{Cp}$ −14 ns		$0.5t_{Cp}$ −10 ns		$0.5t_{Cp}$ −6 ns		
$t_{CLh(1,2)}$	4-17	PHI1, PHI2 High Time	At 90% V_{CC} on PHI1, PHI2	$0.5t_{Cp}$ −18 ns		$0.5t_{Cp}$ −15 ns		$0.5t_{Cp}$ −10 ns		
$t_{nOVL(1,2)}$	4-17	Non-overlap time	At 10% V_{CC} on PHI1, PHI2	0	7	0	7	0	6	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 10% V_{CC} on PHI1, PHI2	−4	4	−4	4	−3	3	ns
t_{CLwas}		PHI1, PHI2 asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−3	3	ns

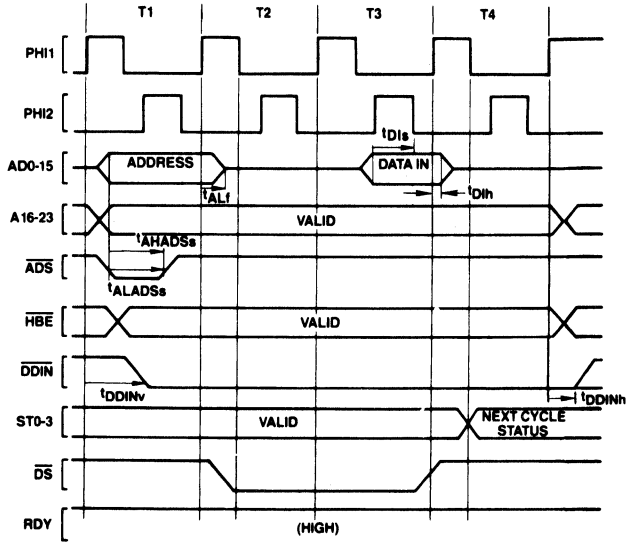
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



TL/EE/8525–41

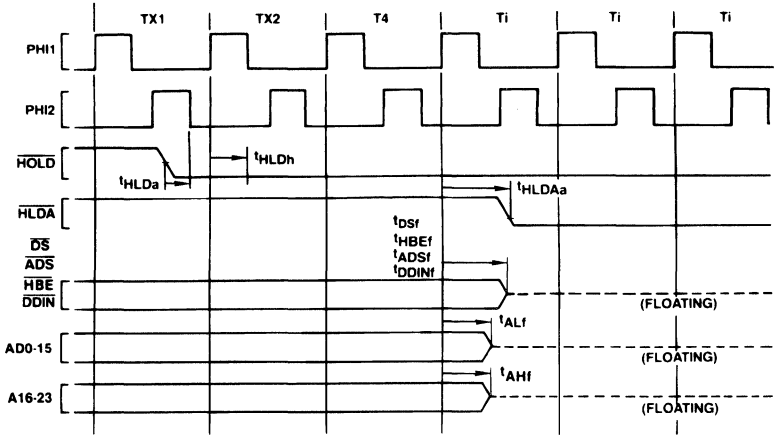
FIGURE 4-4. Write Cycle



TL/EE/8525–42

FIGURE 4-5. Read Cycle

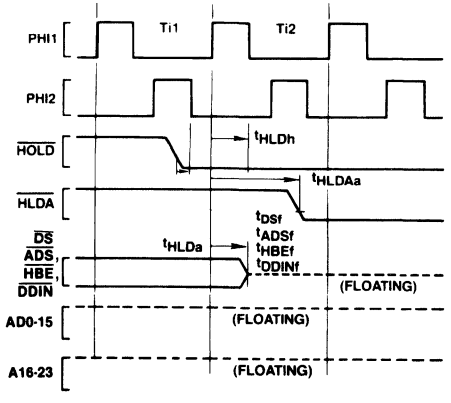
4.0 Device Specifications (Continued)



TL/EE/8525–43

FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially)

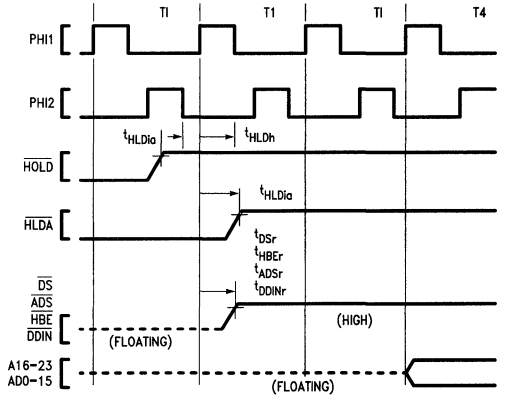
Note that whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ request ($\overline{\text{HOLD}}$ low) must be active t_{HLDa} before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until t_{HLDh} after the rising edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.



TL/EE/8525–44

FIGURE 4-7. Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle)

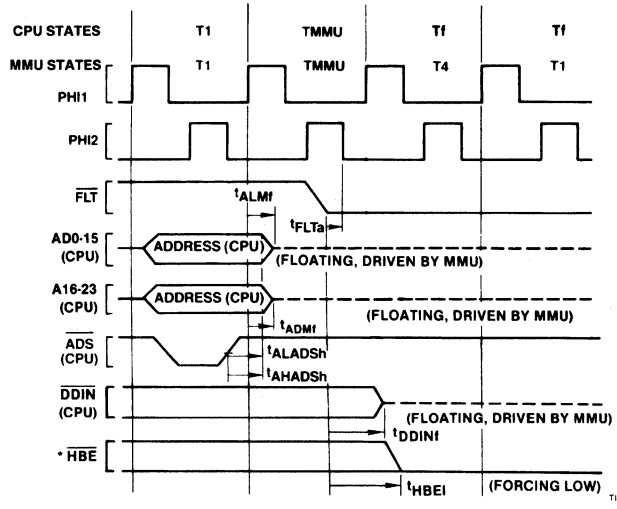
Note that during Ti1 the CPU is already idling.



TL/EE/8525–80

FIGURE 4-8. Release from $\overline{\text{HOLD}}$

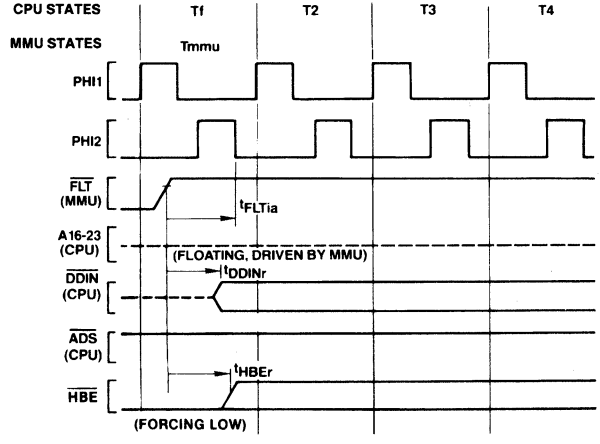
4.0 Device Specifications (Continued)



TL/EE/8525–46

***Note:** In future higher speed versions of the NS32C016, $\overline{HBE}$ will no longer be affected by $\overline{FLT}$. See *Figure B-1* in Appendix B for the required modification to the interface logic.

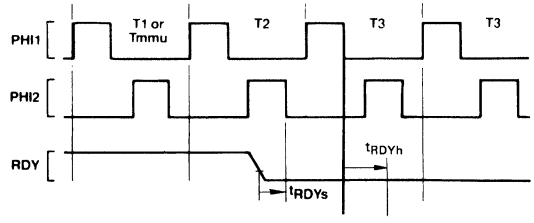
FIGURE 4-9. $\overline{FLT}$ Initiated Cycle Timing



TL/EE/8525–47

Note that when $\overline{FLT}$ is deasserted the CPU restarts driving $\overline{DDIN}$ before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force $\overline{DDIN}$ to the same logic level.

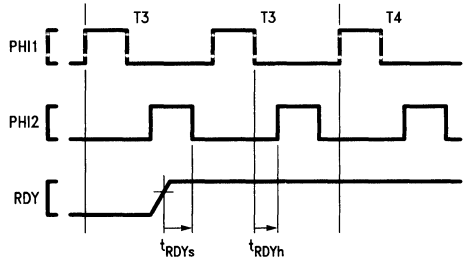
FIGURE 4-10. Release from $\overline{FLT}$ Timing



TL/EE/8525–48

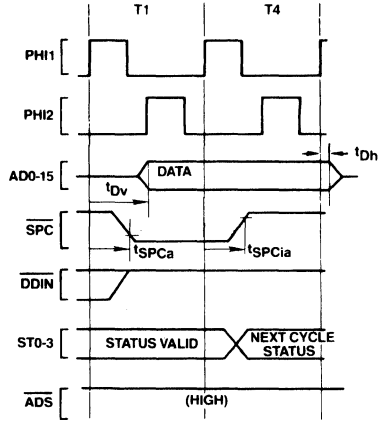
FIGURE 4-11. Ready Sampling (CPU Initially READY)

4.0 Device Specifications (Continued)



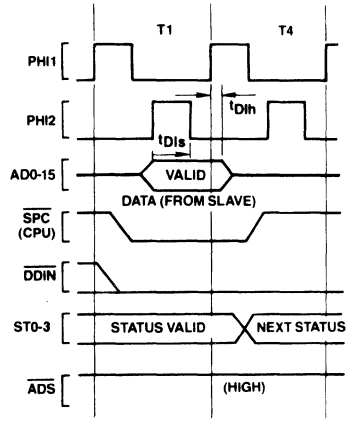
TL/EE/8525-49

FIGURE 4-12. Ready Sampling (CPU Initially NOT READY)



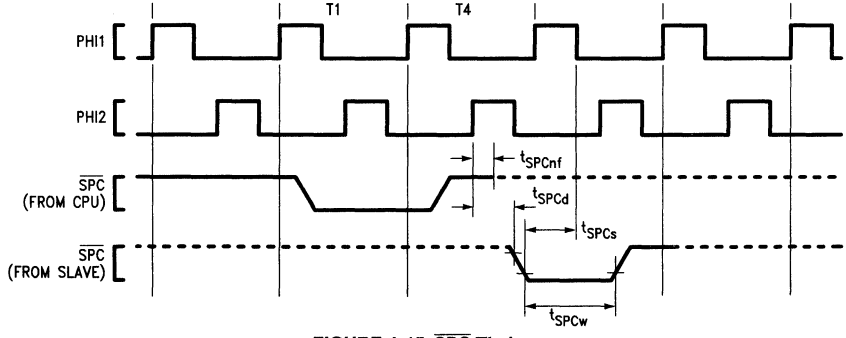
TL/EE/8525-50

FIGURE 4-13. Slave Processor Write Timing



TL/EE/8525-51

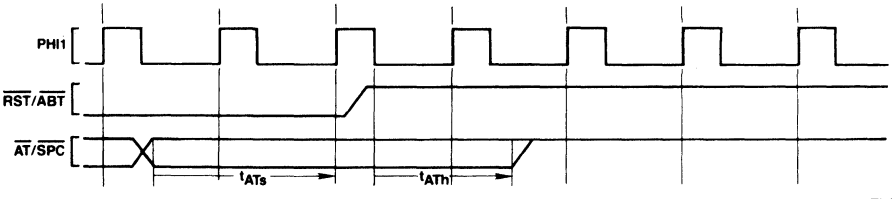
FIGURE 4-14. Slave Processor Read Timing



TL/EE/8525-81

FIGURE 4-15. $\overline{SPC}$ Timing

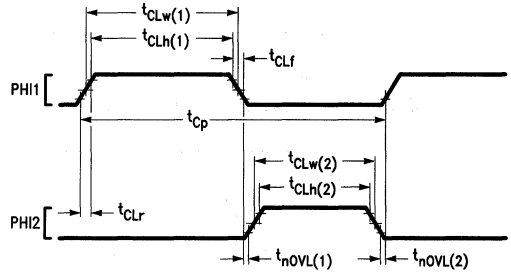
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{SPC}$ high with internal 5 kΩ pullup.



TL/EE/8525-53

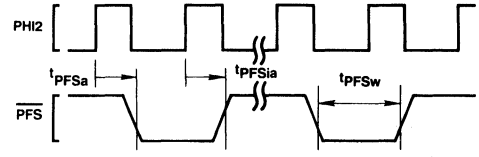
FIGURE 4-16. Reset Configuration Timing

4.0 Device Specifications (Continued)



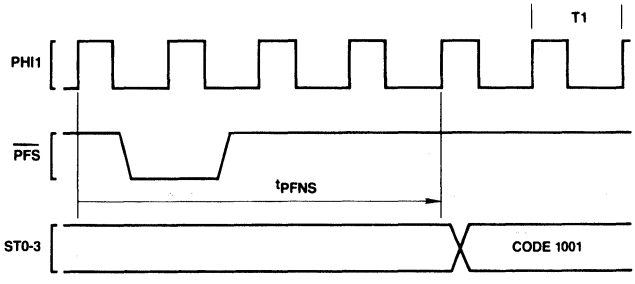
TL/EE/8525–54

FIGURE 4-17. Clock Waveforms



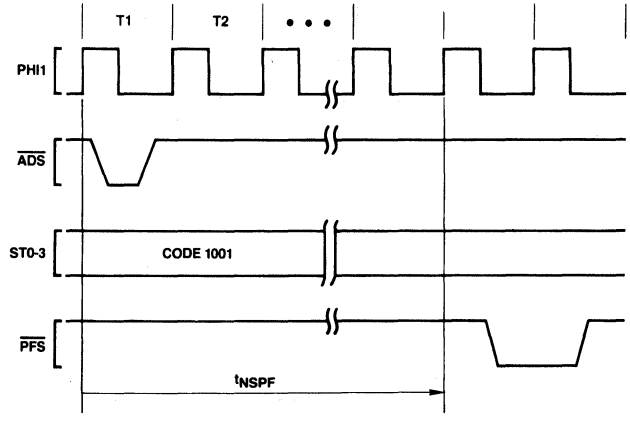
TL/EE/8525–55

FIGURE 4-18. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/8525–56

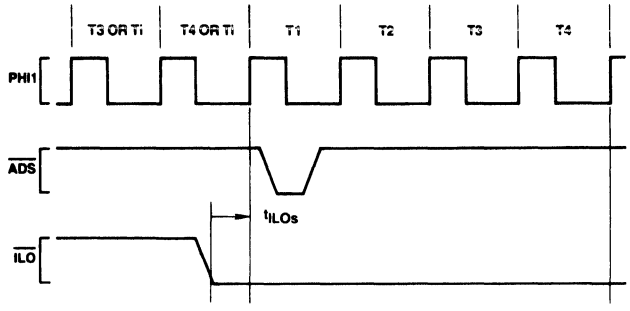
FIGURE 4-19a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/8525–57

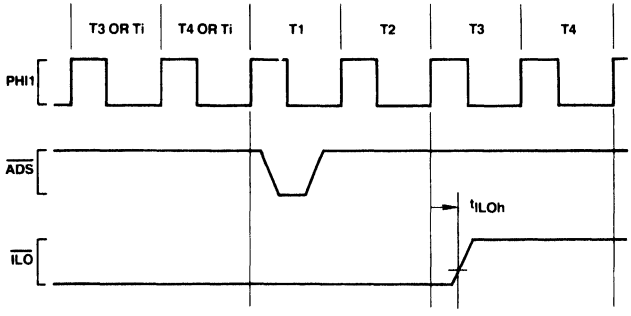
FIGURE 4-19b. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



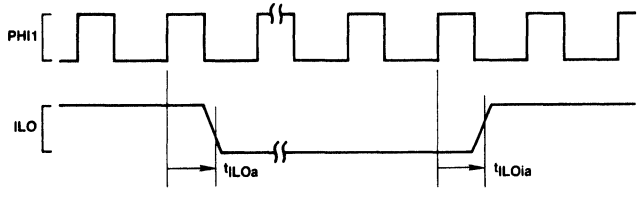
TL/EE/8525-58

FIGURE 4-20a. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction



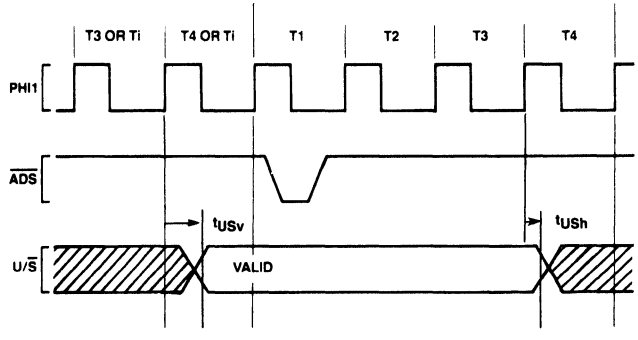
TL/EE/8525-59

FIGURE 4-20b. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction



TL/EE/8525-60

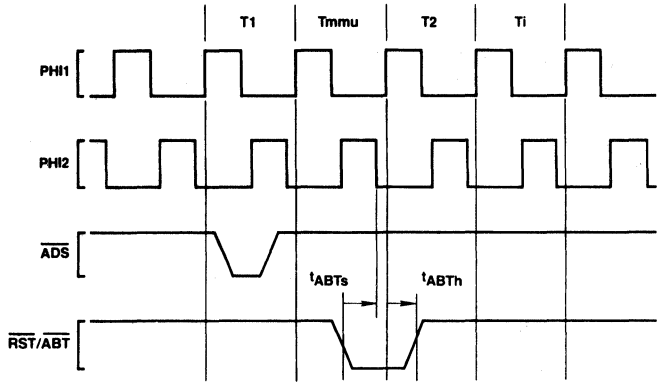
FIGURE 4-21. Relationship of $\overline{ILO}$ to Any Clock Cycle



TL/EE/8525-61

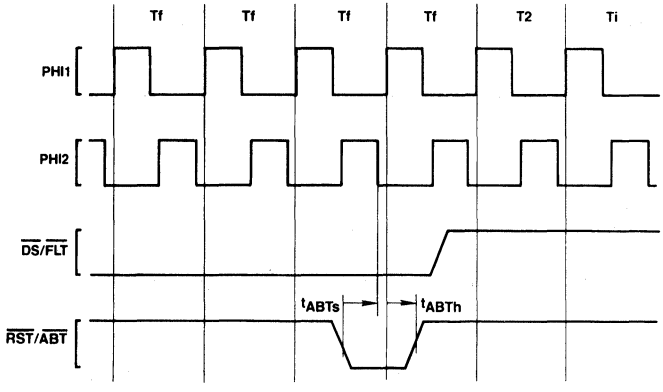
FIGURE 4-22. U/$\overline{S}$ Relationship to Any Bus Cycle— Guaranteed Valid Interval

4.0 Device Specifications (Continued)



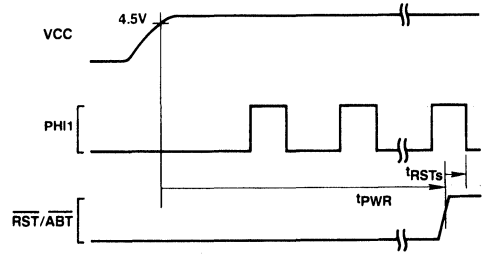
TL/EE/8525–62

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Not Applied



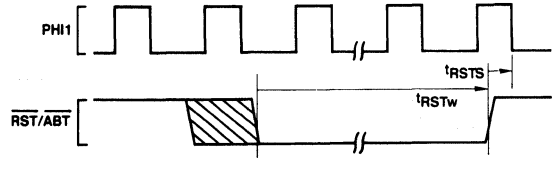
TL/EE/8525–63

FIGURE 4-24. Abort Timing, $\overline{FLT}$ Applied



TL/EE/8525–64

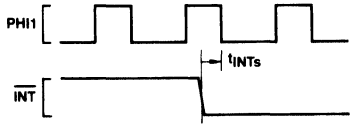
FIGURE 4-25. Power-On Reset



TL/EE/8525–65

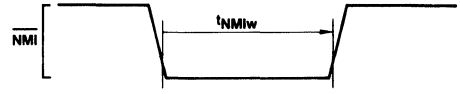
FIGURE 4-26. Non-Power-On Reset

4.0 Device Specifications (Continued)



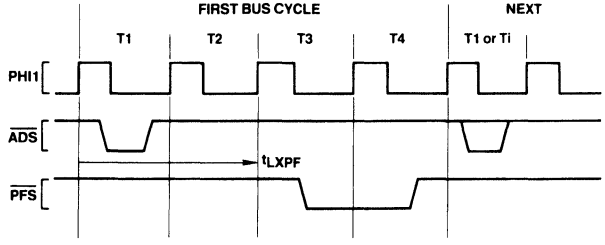
TL/EE/8525-66

FIGURE 4-27. $\overline{INT}$ Interrupt Signal Detection



TL/EE/8525-67

FIGURE 4-28. $\overline{NMI}$ Interrupt Signal Timing



TL/EE/8525-68

FIGURE 4-29. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction

NOTE:

In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

2

Appendix A: Instruction Formats

NOTATIONS:

i = Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f = Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c = Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op = Operation Code
- Valid encodings shown with each format.

gen, gen 1, gen 2 = General Addressing Mode Field
- See Sec. 2.2 for encodings.

reg = General Purpose Register Number

cond = Condition Code Field
- 0000 = EQual: Z = 1
- 0001 = Not Equal: Z = 0
- 0010 = Carry Set: C = 1
- 0011 = Carry Clear: C = 0
- 0100 = Higher: L = 1
- 0101 = Lower or Same: L = 0
- 0110 = Greater Than: N = 1
- 0111 = Less or Equal: N = 0
- 1000 = Flag Set: F = 1
- 1001 = Flag Clear: F = 0
- 1010 = LOwer: L = 0 and Z = 0
- 1011 = Higher or Same: L = 1 or Z = 1
- 1100 = Less Than: N = 0 and Z = 0
- 1101 = Greater or Equal: N = 1 or Z = 1
- 1110 = (Unconditionally True)
- 1111 = (Unconditionally False)

short = Short Immediate Value. May contain:
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.
  - 0000 = US
  - 0001 – 0111 = (Reserved)
  - 1000 = FP
  - 1001 = SP
  - 1010 = SB
  - 1011 = (Reserved)
  - 1100 = (Reserved)
  - 1101 = PSR
  - 1110 = INTBASE
  - 1111 = MOD

Options: in String Instructions

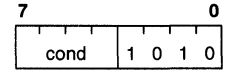
U/W	B	T

T = Translated
B = Backward
U/W = 00: None
- 01: While Match
- 11: Until Match

Configuration bits, in SETCFG:

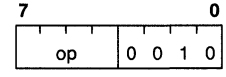
C	M	F	I

mreg: MMU Register number, in LMR, SMR.
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = PF0
- 0101 = PF1
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = SC
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA



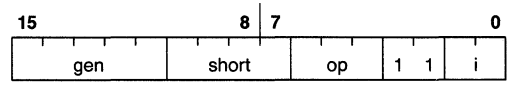
Format 0

Bcond (BR)



Format 1

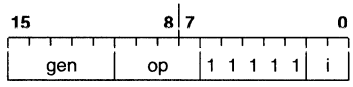
BSR	–0000	ENTER	–1000
RET	–0001	EXIT	–1001
CXP	–0010	NOP	–1010
RXP	–0011	WAIT	–1011
RETT	–0100	DIA	–1100
RETI	–0101	FLAG	–1101
SAVE	–0110	SVC	–1110
RESTORE	–0111	BPT	–1111



Format 2

ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		

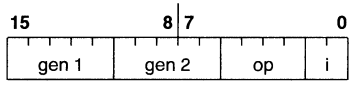
Appendix A: Instruction Formats (Continued)



Format 3

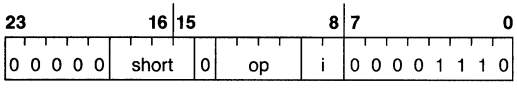
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

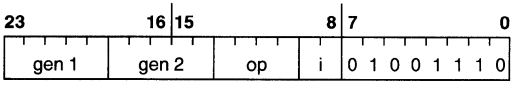
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

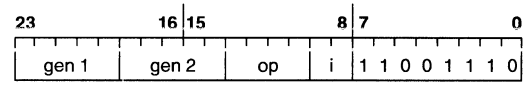
MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

Trap (UND) on 1XXX, 01XX



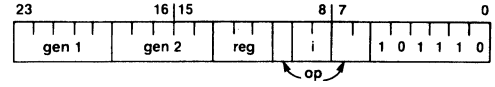
Format 6

ROT	−0000	NEG	−1000
ASH	−0001	NOT	−1001
CBIT	−0010	Trap (UND)	−1010
CBITI	−0011	SUBP	−1011
Trap (UND)	−0100	ABS	−1100
LSH	−0101	COM	−1101
SBIT	−0110	IBIT	−1110
SBITI	−0111	ADDP	−1111



Format 7

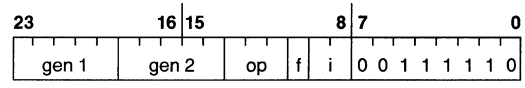
MOVM	−0000	MUL	−1000
CMPM	−0001	MEI	−1001
INSS	−0010	Trap (UND)	−1010
EXTS	−0011	DEI	−1011
MOVXBW	−0100	QUO	−1100
MOVZBW	−0101	REM	−1101
MOVZiD	−0110	MOD	−1110
MOVXiD	−0111	DIV	−1111



TL/EE/8525-69

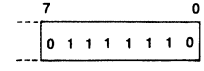
Format 8

EXT	−000	INDEX	−100
CVTP	−001	FFS	−101
INS	−010		
CHECK	−011		
MOVSU	−110, reg=001		
MOVUS	−110, reg=011		



Format 9

MOVif	−000	ROUND	−100
LFSR	−001	TRUNC	−101
MOVLF	−010	SFSR	−110
MOVFL	−011	FLOOR	−111

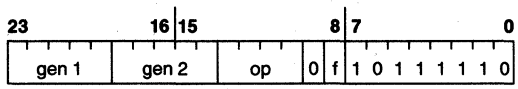


TL/EE/8525-70

Format 10

Trap (UND) Always

Appendix A: Instruction Formats (Continued)



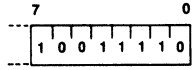
Format 11

ADDf	−0000	DIVf	−1000
MOVf	−0001	Trap (SLAVE)	−1001
CMPf	−0010	Trap (UND)	−1010
Trap (SLAVE)	−0011	Trap (UND)	−1011
SUBf	−0100	MULf	−1100
NEGf	−0101	ABSf	−1101
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111

TL/EE/8525–71

Format 12

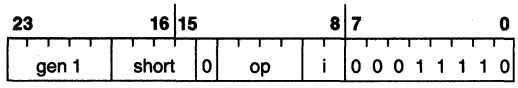
Trap (UND) Always



TL/EE/8525–72

Format 13

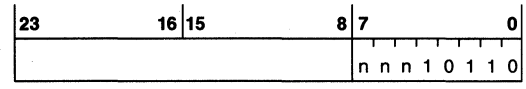
Trap (UND) Always



Format 14

RDVAL	−0000	LMR	−0010
WRVAL	−0001	SMR	−0011

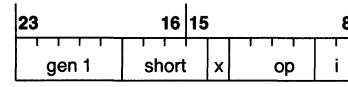
Trap (UND) on 01XX, 1XXX



Format 15
(Custom Slave)

nnn **Operation Word Format**

000

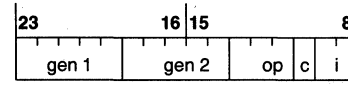


Format 15.0

CATST0	−0000	LCR	−0010
CATST1	−0001	SCR	−0011

Trap (UND) on all others

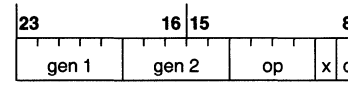
001



Format 15.1

CCV3	−000	CCV2	−100
LCSR	−001	CCV1	−101
CCV5	−010	SCSR	−110
CCV4	−011	CCV0	−111

101

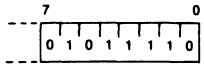


Format 15.5

CCAL0	−0000	CCAL3	−1000
CMOV0	−0001	CMOV3	−1001
CCMP0	−0010	Trap (UND)	−1010
CCMP1	−0011	Trap (UND)	−1011
CCAL1	−0100	CCAL2	−1100
CMOV2	−0101	CMOV1	−1101
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111

If nnn = 010, 011, 100, 110, 111
then Trap (UND) Always

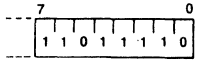
Appendix A: Instruction Formats (Continued)



TL/EE/8525–73

Format 16

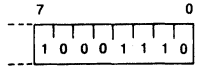
Trap (UND) Always



TL/EE/8525–75

Format 17

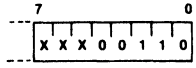
Trap (UND) Always



TL/EE/8525–76

Format 18

Trap (UND) Always

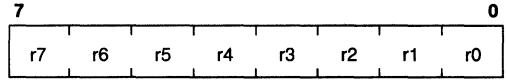


TL/EE/8525–74

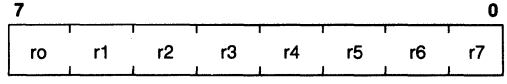
Format 19

Trap (UND) Always

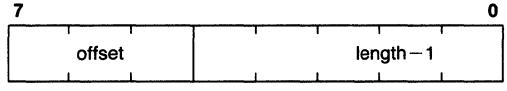
Implied Immediate Encodings:



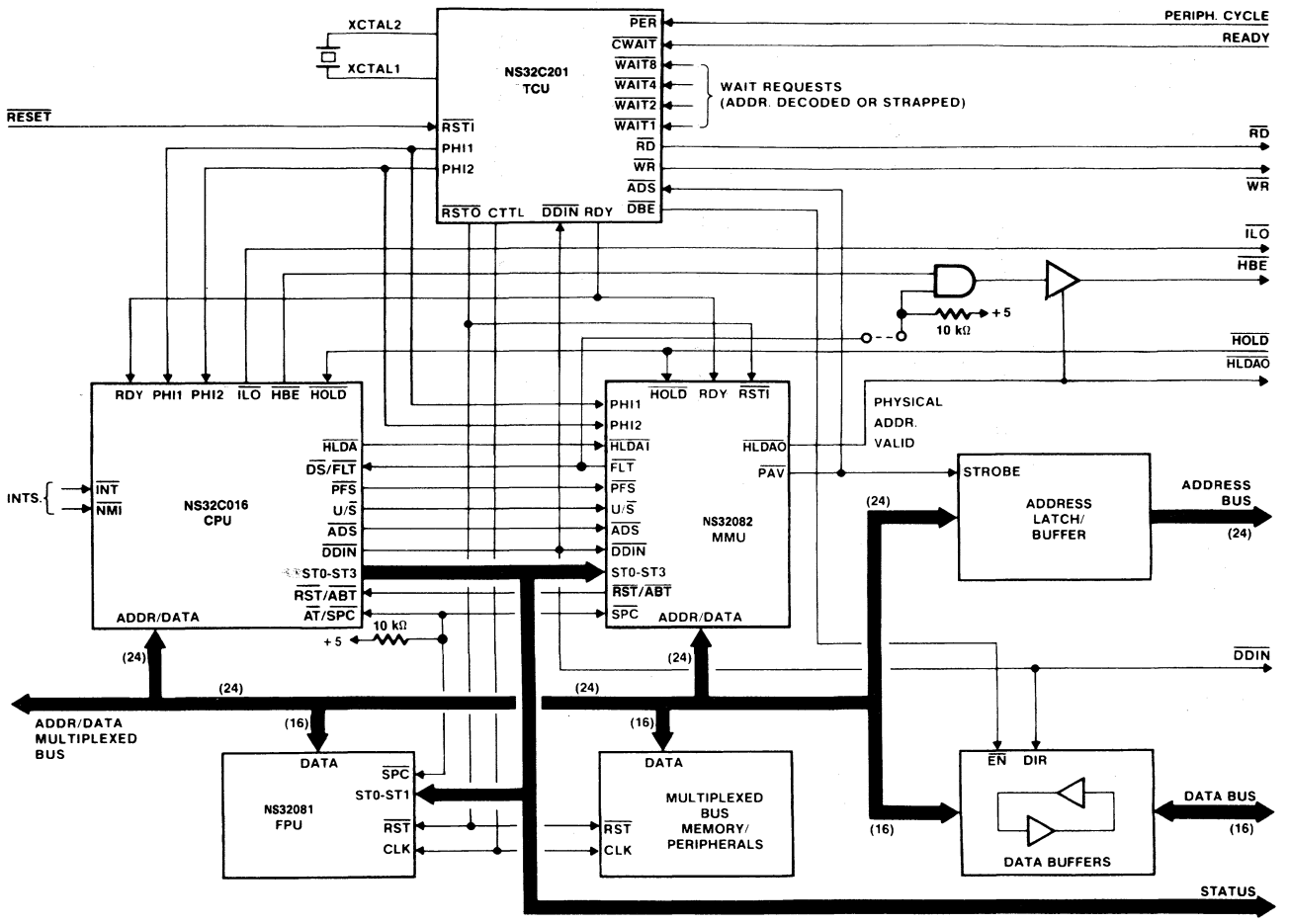
Register Mask, appended to SAVE, ENTER



Register Mask, appended to RESTORE, EXIT



Offset/Length Modifier appended to INSS, EXTS



TL/EE/8525–77

FIGURE B-1. System Connection Diagram

National Semiconductor Corporation

PRELIMINARY

NS32016-6/NS32016-8/NS32016-10 High-Performance Microprocessors

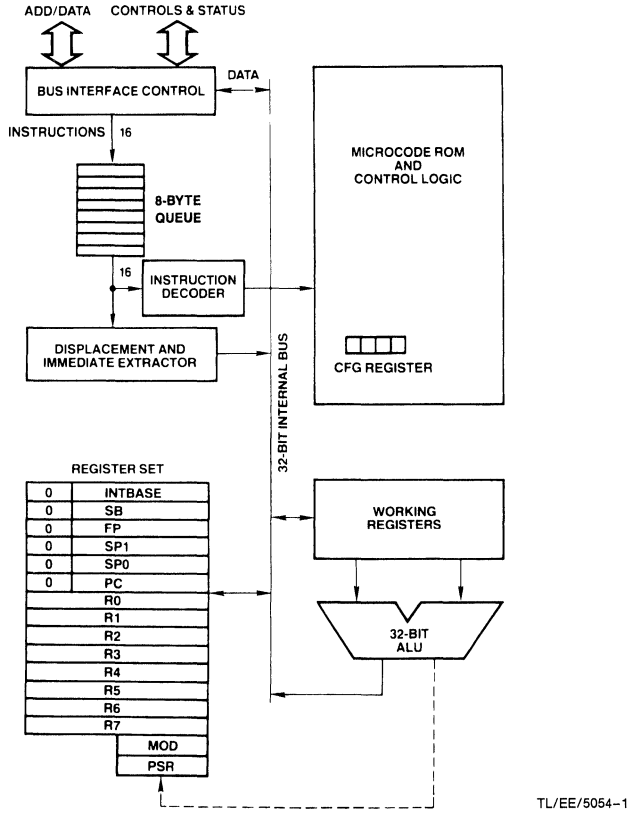
General Description

The NS32016 is a 32-bit, virtual memory microprocessor with a 16-MByte linear address space and a 16-bit external data bus. It has a 32-bit ALU, eight 32-bit general purpose registers, an eight-byte prefetch queue, and a slave processor interface. The NS32016 is fabricated with National Semiconductor's advanced XMOS process, and is fully object code compatible with other Series 32000 processors. The Series 32000 instructions set is optimized for modular high-level languages (HLL). The set is very symmetric, it has a two address format, and it incorporates HLL oriented addressing modes. The capabilities of the NS32016 can be expanded with the use of the NS32081 floating point unit (FPU), and the NS32082 demand-paged virtual memory management unit (MMU). Both devices interface to the NS32016 as slave processors. The NS32016 is a general purpose microprocessor that is ideal for a wide range of computational intensive applications.

Features

- 32-bit architecture and implementation
- Virtual memory support
- 16-MByte linear address space
- 16-bit external data bus
- Powerful instruction set
  - — General 2-address capability
  - — High degree of symmetry
  - — Addressing modes optimized for high-level languages
- Series 32000 slave processor support
- High-speed XMOS™ technology
- 48-pin dual-in-line (DIP) package

Block Diagram



TL/EE/5054–1

Table of Contents

1.0 PRODUCT INTRODUCTION
2.0 ARCHITECTURAL DESCRIPTION
2.1 Programming Model
2.1.1 General Purpose Registers
2.1.2 Dedicated Register
2.1.3 The Configuration Register (CFG)
2.1.4 Memory Organization
2.1.5 Dedicated Tables
2.2 Instruction Set
2.2.1 General Instruction Format
2.2.2 Addressing Modes
2.2.3 Instruction Set Summary

3.0 FUNCTIONAL DESCRIPTION
3.1 Power and Grounding
3.2 Clocking
3.3 Resetting
3.4 Bus Cycles
3.4.1 Cycle Extension
3.4.2 Bus Status
3.4.3 Data Access Sequences
3.4.3.1 Bit Access
3.4.3.2 Bit Field Accesses
3.4.3.3 Extending Multiply Accesses
3.4.4 Instruction Fetches
3.4.5 Interrupt Control Cycles
3.4.6 Slave Processor Communication
3.4.6.1 Slave Processor Bus Cycles
3.4.6.2 Slave Operand Transfer Sequences
3.5 Memory Management Option
3.5.1 Address Translation Strap
3.5.2 Translated Bus Timing
3.5.3 The $\overline{FLT}$ (Float) Pin
3.5.4 Aborting Bus Cycles
3.5.4.1 The Abort Interrupt
3.5.4.2 Hardware Considerations
3.6 Bus Access Control
3.7 Dual Processing
3.7.1 Bus Arbitration
3.7.2 Processor Assignment

3.0 FUNCTIONAL DESCRIPTION (Continued)
3.8 Instruction Status
3.8.1 General Interrupt/Trap Sequence
3.8.2 Interrupt/Trap Return
3.8.3 Maskable Interrupts (The $\overline{INT}$ Plan)
3.8.3.1 Non-Vectored Mode
3.8.3.2 Vectored Mode: Non-Cascaded Case
3.8.3.3 Vectored Mode: Cascaded Case
3.8.4 Non-Maskable Interrupt (The $\overline{NMI}$ Pin)
3.8.5 Traps
3.8.6 Prioritization
3.8.7 Interrupt/Trap Sequence: Detail Flow
3.8.7.1 Maskable/Non-Maskable Interrupt Sequence
3.8.7.2 Trap Sequence: Traps Other Than Trace
3.9 Slave Processor Instructions
3.9.1 Slave Processor Protocol
3.9.2 Floating Point Instructions
3.9.3 Memory Management Instructions
3.9.4 Custom Slave Instructions

4.0 DEVICE SPECIFICATIONS
4.1 Pin Descriptions
4.1.1 Supplies
4.1.2 Input Signals
4.1.3 Output Signals
4.1.4 Input/Output Signals
4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics
4.4.1 Definitions
4.4.2 Timing Tables
4.4.2.1 Output Signals: Internal Propagation Delays
4.4.2.2 Input Signals Requirements
4.4.2.3 Clocking Requirements
4.4.3 Timing Requirements

Appendix A: Instruction Formats
B: Interfacing Suggestions

List of Illustrations

The General and Dedicated Registers 2-1
Processor Status Register 2-2
CFG Register 2-3
Module Descriptor Format 2-4
A Sample Link Table 2-5
General Instruction Format 2-6
Index Byte Format 2-7
Displacement Encodings 2-8
Recommended Supply Connections 3-1
Clock Timing Relationships 3-2
Power-On Reset Requirements 3-3
General Reset Timing 3-4
Recommended Reset Connections, Non-Memory-Managed System 3-5a
Recommended Reset Connections, Memory-Managed System 3-5b
Bus Connections 3-6
Read Cycle Timing 3-7
Write Cycle Timing 3-8
RDY Pin Timing 3-9
Extended Cycle Example 3-10
Memory Interface 3-11
Slave Processor Connections 3-12
CPU Read from Slave Processor 3-13
CPU Write to Slave Processor 3-14
Read Cycle with Address Translation (CPU Action) 3-15
Write Cycle with Address Translation (CPU Action) 3-16
Memory-Managed Read Cycle 3-17
Memory-Managed Write Cycle 3-18
$\overline{\text{FLT}}$ Timing 3-19
$\overline{\text{HOLD}}$ Timing, Bus Initially Idle 3-20
$\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle 3-21
Interrupt Dispatch and Cascade Tables 3-22
Interrupt/Trap Service Routine Calling Sequence 3-23
Return from Trap (RETT n) Instruction Flow 3-24
Return from Interrupt (RET) Instruction Flow 3-25
Interrupt Control Connections (16 levels) 3-26
Cascaded Interrupt Control Unit Connections 3-27
Service Sequence 3-28
Slave Processor Protocol 3-29
Slave Processor Status Word Format 3-30
NS32016 Connection Diagram 4-1
Timing Specification Standard (Signal Valid after Clock Edge) 4-2
Timing Specification Standard (Signal Valid before Clock Edge) 4-3
Write Cycle 4-4
Read Cycle 4-5
Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially) 4-6
Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle) 4-7

List of Illustrations (Continued)

Release from $\overline{HOLD}$ 4-8
$\overline{FLT}$ Initiated Float Cycle Timing 4-9
Release from $\overline{FLT}$ Timing 4-10
Ready Sampling (CPU Initially READY) 4-11
Ready Sampling (CPU Initially NOT READY) 4-12
Slave Processor Write Timing 4-13
Slave Processor Read Timing 4-14
$\overline{SPC}$ Timing 4-15
Reset Configuration Timing 4-16
Clock Waveforms 4-17
Relationship of PFS to Clock Cycles 4-18
Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch 4-19a
Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$ 4-19b
Relationship of $\overline{ILO}$ to First Operand of an Interlocked Instruction 4-20a
Relationship of $\overline{ILO}$ to Last Operand of an Interlocked Instruction 4-20b
Relationship of $\overline{ILO}$ to Any Clock Cycle 4-21
U/$\overline{S}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval 4-22
Abort Timing, $\overline{FLT}$ Not Applied 4-23
Abort Timing, $\overline{FLT}$ Applied 4-24
Power-On Reset 4-25
Non-Power-On Reset 4-26
$\overline{INT}$ Interrupt Signal Detection 4-27
$\overline{NMI}$ Interrupt Signal Timing 4-28
Relationship between Last Data Transfer of an Instruction and PFS Pulse of Next Instruction 4-29
System Connection Diagram B-1

List of Tables

NS32016 Addressing Modes 2-1
NS32016 Instruction Set Summary 2-2
Bus Cycle Categories 3-1
Access Sequences 3-2
Interrupt Sequences 3-3
Floating Point Instruction Protocols 3-4
Memory Management Instruction Protocols 3-5
Custom Slave Instruction Protocols 3-6

1.0 Product Introduction

The Series 32000 Microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from a minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operations. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes: Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types: The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set: While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations: The Series 32000 CPUs represent two-operand machines with each operand addressable by all addressing modes. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management: Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing: The NS32016 has 24-bit address pointers that can address up to 16 megabytes without requiring any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support: Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept: The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32016 CPU.

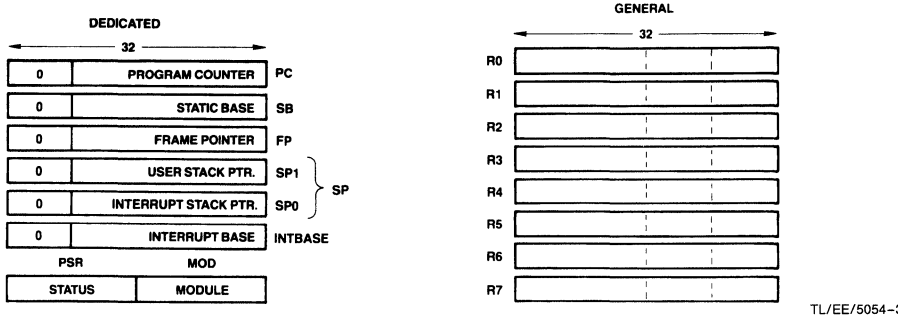


FIGURE 2-1. The General and Dedicated Registers

2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

2.0 Architectural Description (Continued)

2.1.2 Dedicated Registers

The eight dedicated registers of the NS32016 are assigned specific functions.

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32016 the upper eight bits of this register are always zero.)

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 then SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32016 the upper eight bits of these registers are always zero.)

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.

FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32016 the upper eight bits of this register are always zero.)

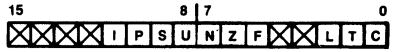
SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32016 the upper eight bits of this register are always zero.)

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32016 the upper eight bits of this register are always zero.)

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

PSR: The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the NS32016 microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/5054–81

FIGURE 2-2. Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Section 3.8.5).

L: The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U=0 the NS32016 is said to be in Supervisor Mode; when U=1 the NS32016 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Section 3.8.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I=1, then all interrupts will be accepted (Section 3.8). If I=0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32016 CPU is the four-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3.*

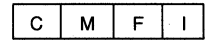


FIGURE 2-3. CFG Register

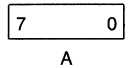
The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{\text{INT}}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Section 3.8.

The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

2.0 Architectural Description (Continued)

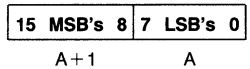
2.1.4 Memory Organization

The main memory of the NS32016 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at $2^{24} - 1$. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.



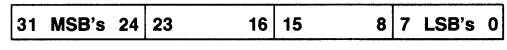
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Section 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.



Word at Address A

Two contiguous words are called a double word. Except where noted (Section 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



Double Word at Address A

Although memory is addressed as bytes, it is actually organized as words. Therefore, words and double words that are aligned to start at even addresses (multiples of two) are accessed more quickly than words and double words that are not so aligned.

2.1.5 Dedicated Tables

Two of the NS32016 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

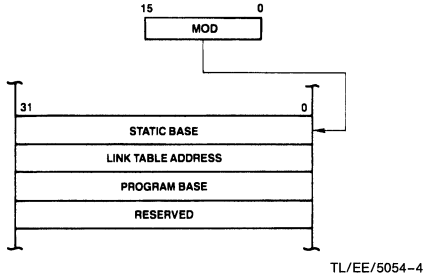
The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Section 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by the NS32016. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically updated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in *Figure 2-4*. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.

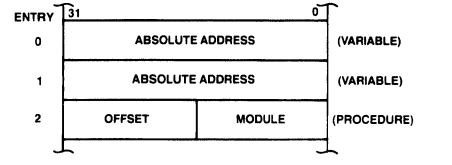
The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.



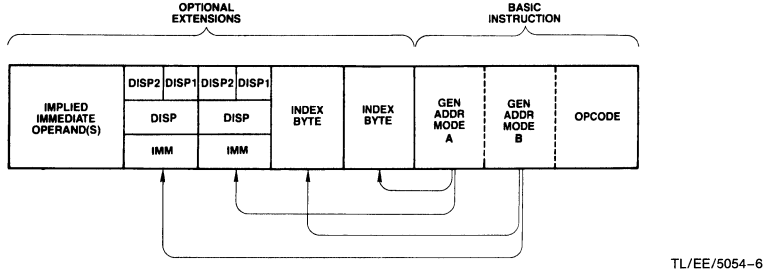
TL/EE/5054-4

FIGURE 2-4. Module Descriptor Format



TL/EE/5054-5

FIGURE 2-5. A Sample Link Table



TL/EE/5054-6

FIGURE 2-6. General Instruction Format

2.0 Architectural Description (Continued)

2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-5.* A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

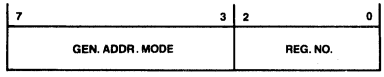
For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-7.*



TL/EE/5054–7

FIGURE 2-7. Index Byte Format

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one of two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in *Figure 2-8,* with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most-significant byte first. Note that this is different from the memory representation of data (Section 2.1.4).

Some instructions require additional "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Section 2.2.3).

2.2.2 Addressing Modes

The NS32016 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the NS32016 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32016 Addressing Modes fall into nine basic types:

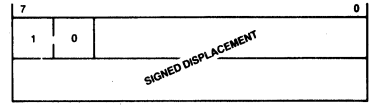
Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

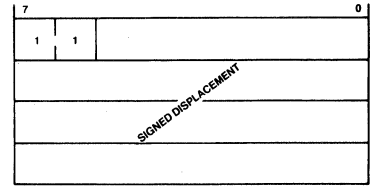
Memory Space: Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

Byte Displacement: Range −64 to +63

Word Displacement: Range −8192 to +8191



Double Word Displacement: Range (Entire Addressing Space)



TL/EE/5054–10

FIGURE 2-8. Displacement Encodings

Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Series 32000 Instruction Set Reference Manual.

2.0 Architectural Description (Continued)

TABLE 2-1
NS32016 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R6 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1 (FP))	Disp2 + Pointer; Pointer found at address Disp 1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1 (SP))	
10010	Static memory relative	disp2(disp1 (SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top Of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	*+ disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2×Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4×Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8×Rn. "Mode" and "n" are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32016 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Series 32000 Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0-R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

TABLE 2-2
NS32016 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move effective address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDi	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply.
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to extended integer.
7	DEIi	gen,gen	Divide extended integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add packed.
6	SUBPi	gen,gen	Subtract packed.

2.0 Architectural Description (Continued)

TABLE 2-2
NS32016 Instruction Set Summary (Continued)

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical shift, left or right.
6	ASHi	gen,gen	Arithmetic shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked.
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit.

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to bit field pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

2.0 Architectural Description (Continued)

TABLE 2-2
NS32016 Instruction Set Summary (Continued)

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 — Comparison Value
R3 — Translation Table Pointer
R2 — String 2 Pointer
R1 — String 1 Pointer
R0 — Limit Count

Options on all string instructions are:

B (Backward): Decrement strong pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Description
5	MOVSi	options	Move string 1 to string 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare string 1 to string 2.
	CMPST	options	Compare, translating string 1 bytes.
5	SKPSi	options	Skip over string 1 entries.
	SKPST	options	Skip, translating bytes for until/while.

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor call.
1	FLAG		Flag trap.
1	BPT		Breakpoint trap.
1	ENTER	[reg list], disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save general purpose registers.
1	RESTORE	[reg list]	Restore general purpose registers.
2	LPRi	areg,gen	Load dedicated register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store dedicated register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust stack pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set configuration register. (Privileged)

2.0 Architectural Description (Continued)

TABLE 2-2
NS32016 Instruction Set Summary (Continued)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a floating point value.
9	MOVLF	gen,gen	Move and shorten a long value to standard.
9	MOVFL	gen,gen	Move and lengthen a standard value to long.
9	MOVif	gen,gen	Convert any integer to standard or long floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load memory management register. (Privileged)
14	SMR	mreg,gen	Store memory management register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUi	gen,gen	Move a value from supervisor space to user space. (Privileged)
8	MOVUSi	gen,gen	Move a value from user space to supervisor space. (Privileged)

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
	CMOV3c	gen,gen	
15.5	CCMPc	gen,gen	Custom compare.
	CCMP1c	gen,gen	
15.1	CCV0ci	gen,gen	Custom convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load custom status register.
15.1	SCSR	gen	Store custom status register.
15.0	CATST0	gen	Custom address/test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load custom register. (Privileged)
15.0	SCR	creg,gen	Store custom register. (Privileged)

3.0 Functional Description

3.1 POWER AND GROUNDING

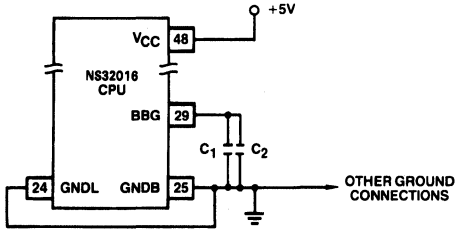
The NS32016 requires a single 5-volt power supply, applied on pin 48 (V_{CC}).

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (*Figure 3-1*).

In addition to V_{CC} and Ground, the NS32016 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Figure 3-1*) from the BBG pin to ground. Recommended values of these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.



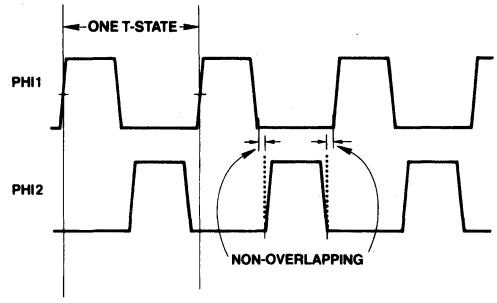
TL/EE/5054–11

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32016 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in *Figure 3-2.*

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See Section 4 for complete specifications of PHI1 and PHI2.



TL/EE/5054–12

FIGURE 3-2. Clock Timing Relationships

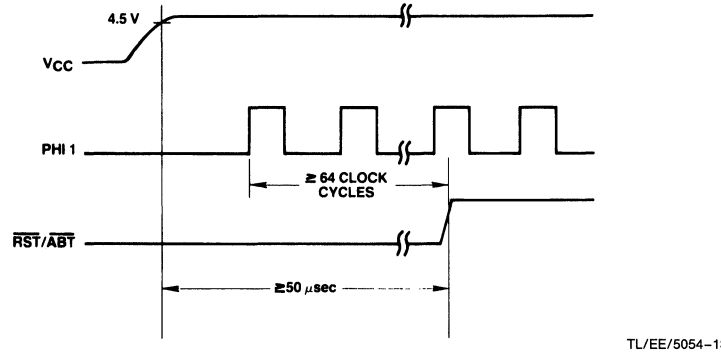
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 RESETTING

The $\overline{RST}/\overline{ABT}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Section 3.5.4.

The CPU may be reset at any time by pulling the $\overline{RST}/\overline{ABT}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, $\overline{RST}/\overline{ABT}$ must be held low for at least 50 μs after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active



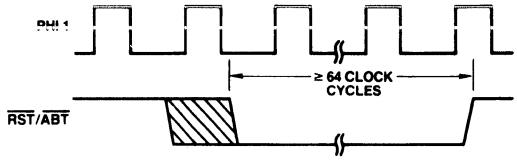
TL/EE/5054–13

FIGURE 3-3. Power-On Reset Requirements

3.0 Functional Description (Continued)

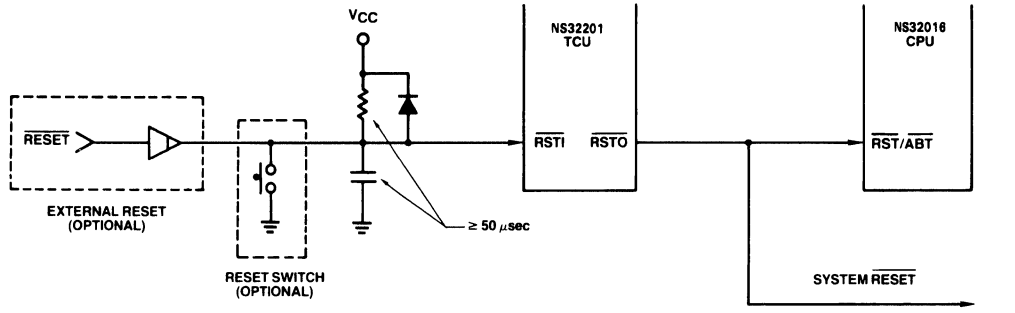
for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4*.

The NS32201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32016 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



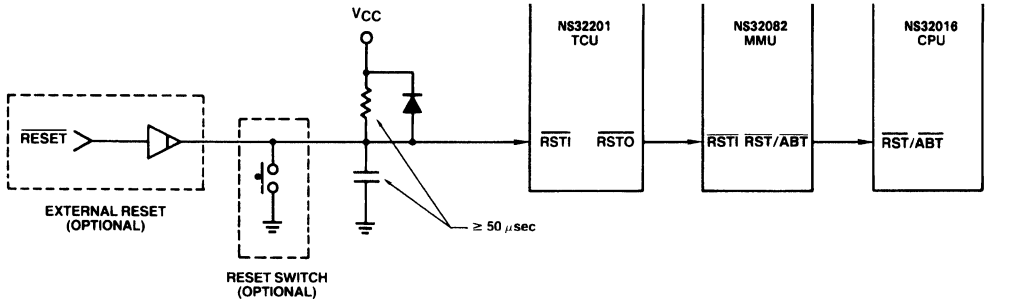
TL/EE/5054–14

FIGURE 3-4. General Reset Timing



TL/EE/5054–15

FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System



TL/EE/5054–16

FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

3.4 BUS CYCLES

The NS32016 CPU has a strap option which defines the Bus Timing Mode as either With or Without Address Translation. This section describes only bus cycles under the No Address Translation option. For details of the use of the strap and of bus cycles with address translation, see Section 3.5.

The CPU will perform a bus cycle for one of the following reasons:

1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
2) To fetch instructions into the eight-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4) To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Section 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0-ST3). Slave Processor cycles differ in that separate control signals are applied (Section 3.4.6).

The sequence of events in a non-Slave bus cycle is shown in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Section 3.4.1).

3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

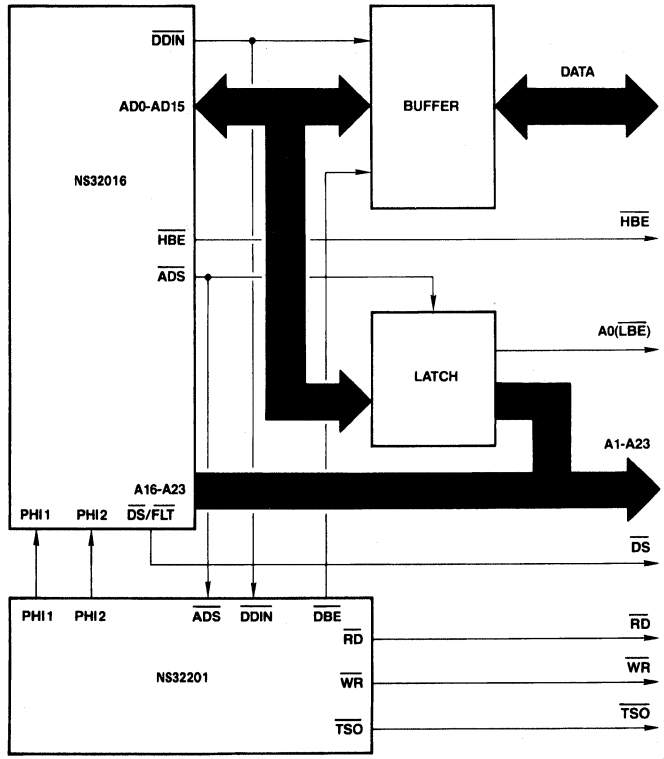
During T1, the CPU applies an address on pins AD0–AD15 and A16–A23. It also provides a low-going pulse on the $\overline{ADS}$ pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–15 from the AD0–AD15 pins. See *Figure 3-6*. During this time also the status signals $\overline{DDIN}$, indicating the direction of the transfer, and $\overline{HBE}$, indicating whether the high byte (AD8–AD15) is to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD15, to either accept or present data. Note that the signals A16–A23 remain valid, and need not be latched. It also starts the data strobe ($\overline{DS}$), signaling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time: $\overline{RD}$ (Read Strobe) or $\overline{WR}$ (Write Strobe), $\overline{TSO}$ (Timing State Output, indicating that T2 has been reached) and $\overline{DBE}$ (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the end of T2 or T3, on the falling edge of the PHI2 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Section 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD15) is sampled at the falling edge of PHI2 of the last T3 state. See Section 4. Data must, however, be held at least until the beginning of T4. $\overline{DS}$ and $\overline{RD}$ are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

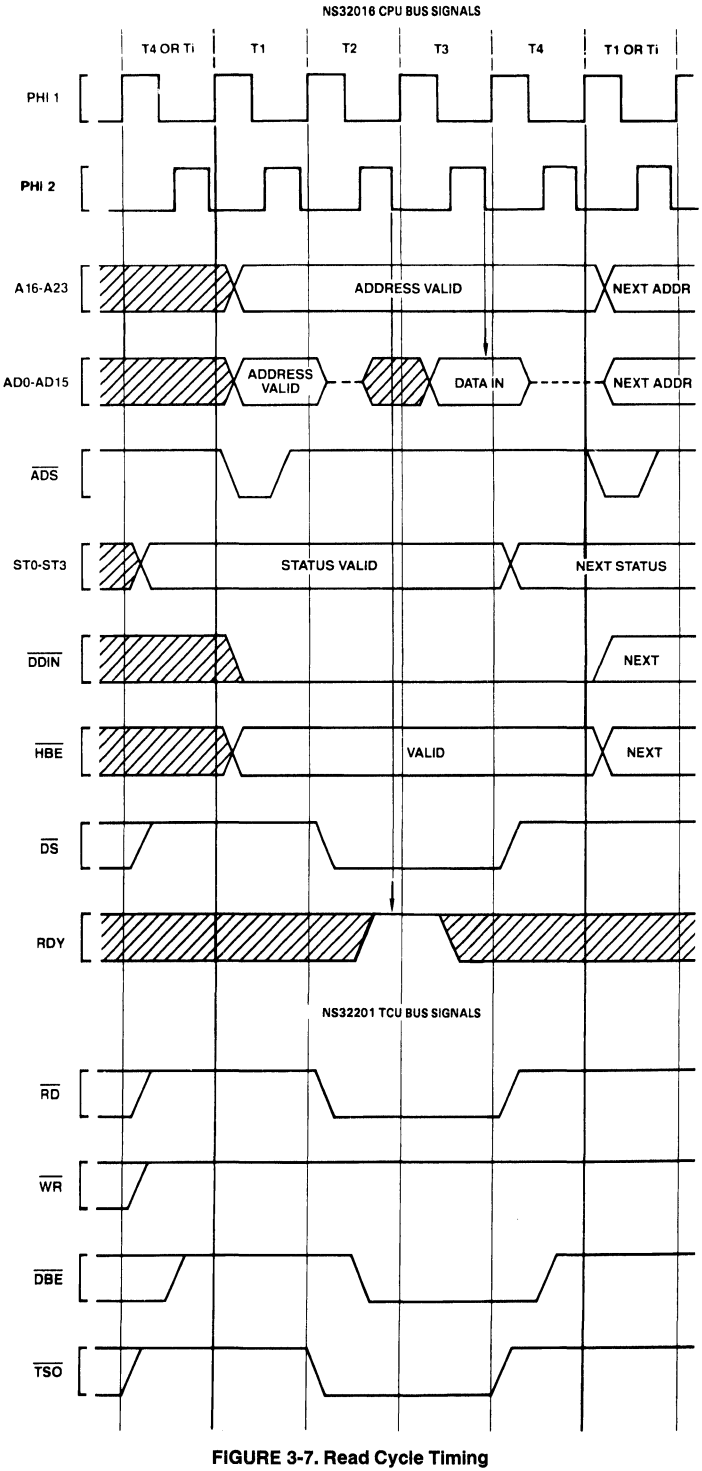
The T4 state finishes the bus cycle. At the beginning of T4, the $\overline{DS}$, $\overline{RD}$, or $\overline{WR}$, and $\overline{TSO}$ signals go inactive, and at the rising edge of PHI2, $\overline{DBE}$ goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).



TL/EE/5054–17

FIGURE 3-6. Bus Connections

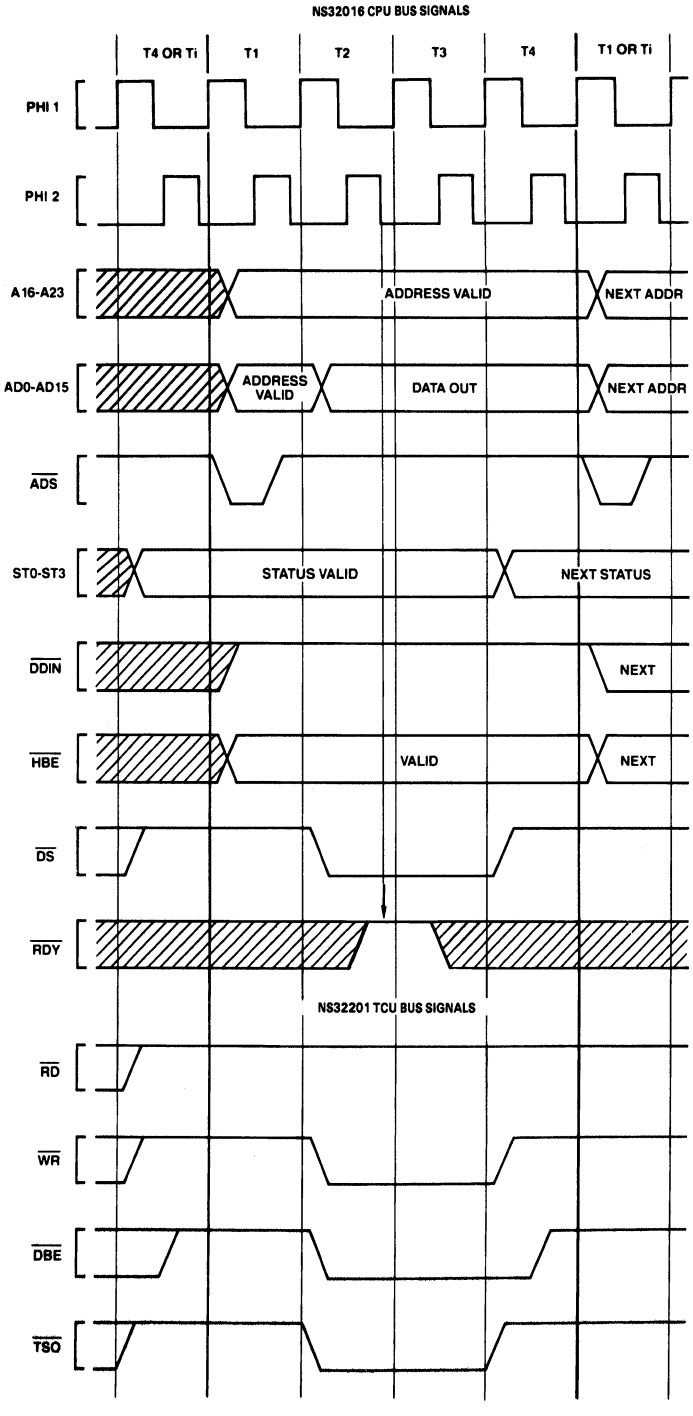
3.0 Functional Description (Continued)



TL/EE/5054–18

FIGURE 3-7. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/5054–19

FIGURE 3-8. Write Cycle Timing

3.0 Functional Description (Continued)

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32016 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

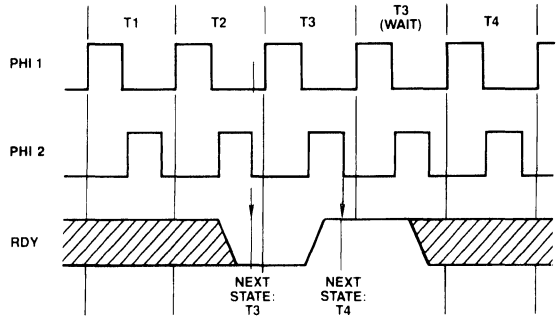
At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If it is sampled low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "wait state." See *Figure 3-9*.

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1) $\overline{\text{CWAIT}}$ (Continues WAIT), which holds the CPU in WAIT states until removed.
2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
3) $\overline{\text{PER}}$ (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 TCU Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU $\overline{\text{WAITn}}$ pins.



TL/EE/5054–20

FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS32016 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why it is idle.

Referring to *Figures 3-7* and *3-8*, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before $\overline{\text{ADS}}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 — The bus is idle because the CPU does not need to perform a bus access.

0001 — The bus is idle because the CPU is executing the WAIT instruction.

0010 — (Reserved for future use.)

0011 — The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 — Interrupt Acknowledge, Master.
The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.
To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Section 3.4.5.

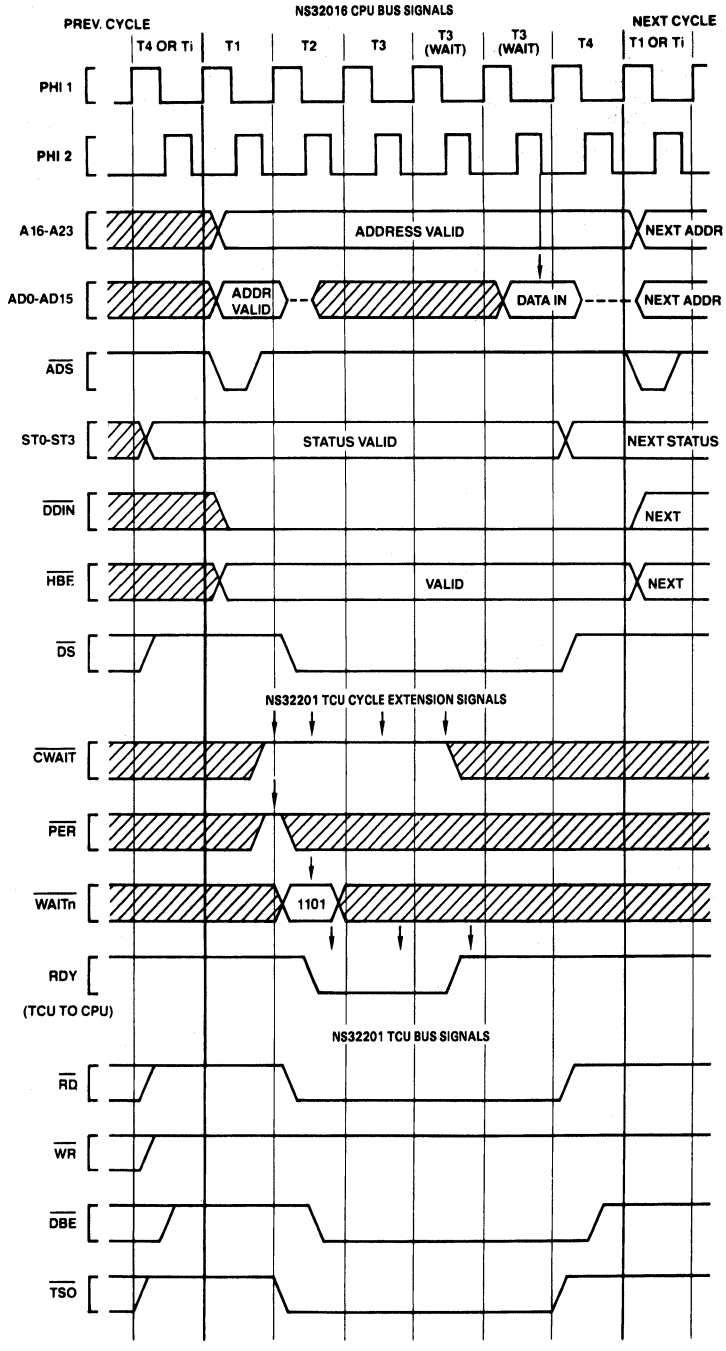
0101 — Interrupt Acknowledge, Cascaded.
The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Section 3.4.5.

0110 — End of Interrupt, Master.
The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Section 3.4.5.

0111 — End of Interrupt, Cascaded.
The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Section 3.4.5.

1000 — Sequential Instruction Fetch.
The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

3.0 Functional Description (Continued)



TL/EE/5054–21

FIGURE 3-10. Extended Cycle Example

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAITn}}$ indicate points at which the TCU samples. Arrows on AD0–AD15 and RDY indicate points at which the CPU samples.

3.0 Functional Description (Continued)

1001 — Non-Sequential Instruction Fetch.
The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 — Data Transfer.
The CPU is reading or writing an operand of an instruction.

1011 — Read RMW Operand.
The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 — Read for Effective Address Calculation.
The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 — Transfer Slave Processor Operand.
The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Section 3.9.1.

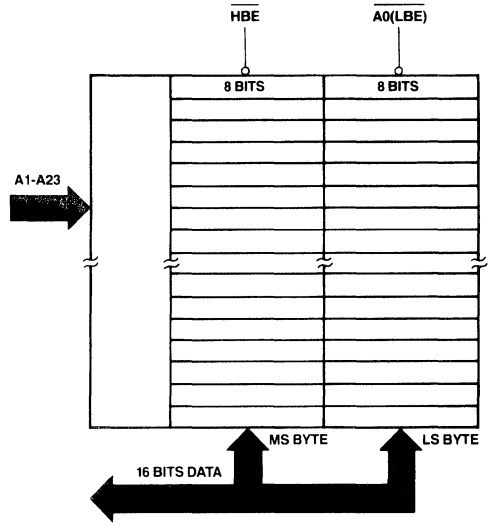
1110 — Read Slave Processor Status.
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Section 3.9.1.

1111 — Broadcast Slave ID.
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Section 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS32016 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32016 is that the presence of a 16-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32016 provides a special control signal, High Byte Enable ($\overline{HBE}$), which facilitates individual byte addressing on a 16-bit bus.

Memory is intended to be organized as two eight-bit banks, each bank receiving the word address (A1–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7, is enabled to respond to even byte addresses; i.e., when the least significant address bit (A0) is low. The other bank, connected to Data Bus pins AD8–AD15, is enabled when $\overline{HBE}$ is low. See *Figure 3-11.*



TL/EE/5054–22

FIGURE 3-11. Memory Interface

Any bus cycle falls into one of three categories: Even Byte Access, Odd Byte Access, and Even Word Access. All accesses to any data type are made up of sequences of these cycles. Table 3-1 gives the state of A0 and $\overline{HBE}$ for each category.

TABLE 3-1
Bus Cycle Categories

Category	$\overline{HBE}$	A0
Even Byte	1	0
Odd Byte	0	1
Even Word	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment (i.e., whether it starts on an even byte address or an odd byte address). Table 3-2 lists the bus cycle performed for each situation. For the timing of A0 and $\overline{HBE}$, see Section 3.4.

3.0 Functional Description (Continued)

TABLE 3.2
Access Sequences

Cycle	Type	Address	$\overline{HBE}$	A0	High Bus	Low Bus	
					BYTE 1	**BYTE 0**	← A
A. Odd Word Access Sequence							
1	Odd Byte	A	0	1	Byte 0	Don't Care	
2	Even Byte	A+1	1	0	Don't Care	Byte 1	
					BYTE 3	**BYTE 2**	**BYTE 1**	**BYTE 0**	← A
B. Even Double-Word Access Sequence									
1	Even Word	A	0	0	Byte 1	Byte 0			
2	Even Word	A+2	0	0	Byte 3	Byte 2			
					BYTE 3	**BYTE 2**	**BYTE 1**	**BYTE 0**	← A
C. Odd Double-Word Access Sequence									
1	Odd Byte	A	0	1	Byte 0	Don't Care			
2	Even Word	A+1	0	0	Byte 2	Byte 1			
3	Even Byte	A+3	1	0	Don't Care	Byte 3			
					BYTE 7	**BYTE 6**	**BYTE 5**	**BYTE 4**	**BYTE 3**	**BYTE 2**	**BYTE 1**	**BYTE 0**	← A
D. Even Quad-Word Access Sequence													
1	Even Word	A	0	0	Byte 1	Byte 0							
2	Even Word	A+2	0	0	Byte 3	Byte 2							
Other bus cycles (instruction prefetch or slave) can occur here.													
3	Even Word	A+4	0	0	Byte 5	Byte 4							
4	Even Word	A+6	0	0	Byte 7	Byte 6							
					BYTE 7	**BYTE 6**	**BYTE 5**	**BYTE 4**	**BYTE 3**	**BYTE 2**	**BYTE 1**	**BYTE 0**	← A
E. Odd Quad-Word Access Sequence													
1	Odd Byte	A	0	1	Byte 0	Don't Care							
2	Even Word	A+1	0	0	Byte 2	Byte 1							
3	Even Byte	A+3	1	0	Don't Care	Byte 3							
Other bus cycles (instruction prefetch or slave) can occur here.													
4	Odd Byte	A+4	0	1	Byte 4	Don't Care							
5	Even Word	A+5	0	0	Byte 6	Byte 5							
6	Even Byte	A+7	1	0	Don't Care	Byte 7							

3.0 Functional Description (Continued)

3.4.3.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.3.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

3.4.4 Instruction Fetches

Instructions for the NS32016 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the eight-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Section 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always Even Word Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle is either an Even Word Read or an Odd Byte Read, depending on whether the destination address is even or odd.

3.4.5 Interrupt Control Cycles

Activating the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pin on the CPU will initiate one or more bus cycles whose purpose is inerrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0–ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32016 interrupt structure, see Section 3.8.

3.0 Functional Description (Continued)

TABLE 3-3
Interrupt Sequences

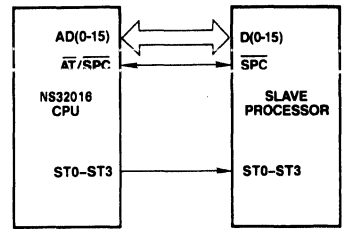
Cycle	Status	Address	$\overline{DDIN}$	$\overline{HBE}$	A0	High Bus	Low Bus
			A. Non-Maskable Interrupt Control Sequences.				
Interrupt Acknowledge							
1	0100	$FFFF00_{16}$	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
			B. Non-Vectored Interrupt Control Sequences.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
			C. Vectored Interrupt Sequences: Non-Cascaded.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Vector: Range: 0–127
Interrupt Return							
1	0110	$FFFE00_{16}$	0	1	0	Don't Care	Vector: Same as in Previous Int. Ack. Cycle
			D. Vectored Interrupt Sequences: Cascaded.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Cascade Index: range −16 to −1
(The CPU here uses the Cascade Indx to find the Cascade Address.)							
2	0101	Cascade Address	0	1 or 0*	0 or 1*	Vector, range 0–255; on appropriate half of Data Bus for even/odd address	
Interrupt Return							
1	0110	$FFFE00_{16}$	0	1	0	Don't Care	Cascade Index: same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address.)							
2	0111	Cascade Address	0	1 or 0*	0 or 1*	Don't Care	Don't Care

* If the Cascaded ICU Address is Even (A0 is low), then the CPU applies $\overline{HBE}$ high and reads the vector number from bits 0–7 of the Data Bus.
If the address is Odd (A0 is high), then the CPU applies $\overline{HBE}$ low and reads the vector number from bits 8–15 of the Data Bus. The vector number may be in the range 0–255.

3.0 Functional Description (Continued)

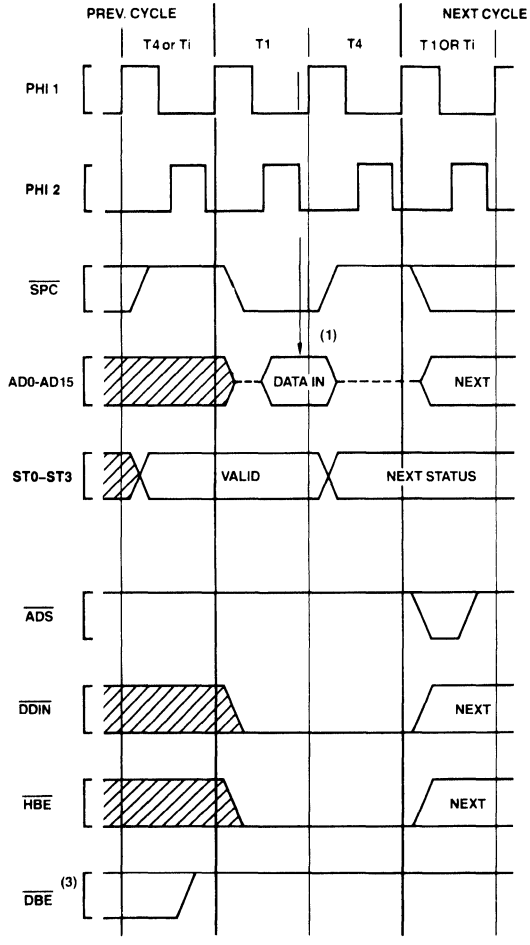
3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Section 3.5.1), the $\overline{AT}/\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{SPC}$). In a slave processor bus cycle, data is transferred on the Data Bus (AD0–AD15), and the Status Lines ST0–ST3 are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Section 3.9 for full protocol sequences.



TL/EE/5054–23

FIGURE 3-12. Slave Processor Connections



TL/EE/5054–24

Note:

(1) CPU samples Data Bus here.

(2) $\overline{DBE}$ and all other NS32201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor

3.0 Functional Description (Continued)

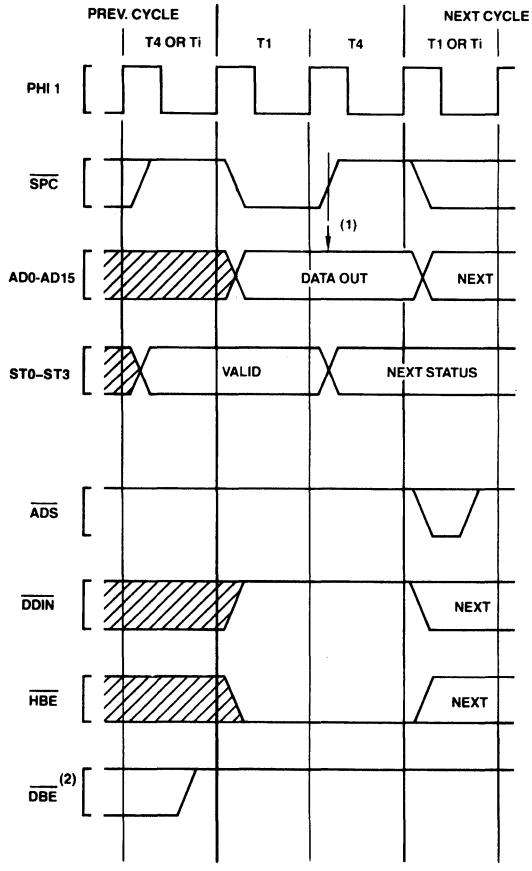
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-13* and *3-14*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ($\overline{ADS}$), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.6.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0–AD7), and a Word operand is transferred on the entire bus. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.



TL/EE/5054–25

Note:

(1) Slave Processor samples data bus here.

(2) $\overline{DBE}$, being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. $\overline{TCU}$ signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32016 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

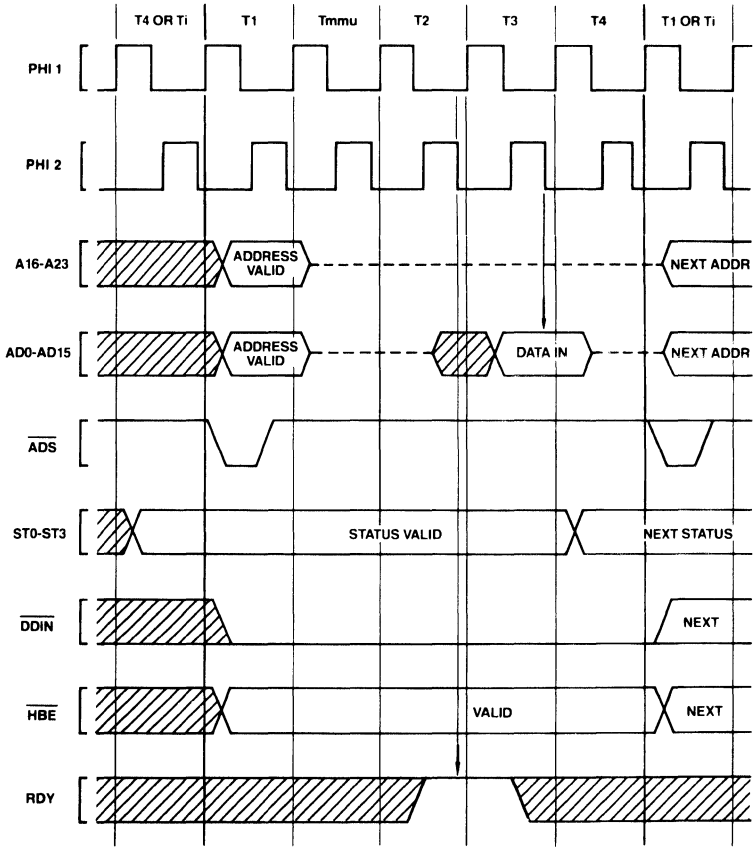
3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32016 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the $\overline{AT}/\overline{SPC}$ (Address Translation/Slave Processor Control) pin on the rising edge of the $\overline{RST}$ (Reset) pulse. If $\overline{AT}/\overline{SPC}$ is sampled as high, the bus timing is as previously described in Section 3.4. If it is sampled as low, two changes occur:

1) An extra clock cycle, Tmmu, is inserted into all bus cycles except Slave Processor transfers.
2) The $\overline{DS}/\overline{FLT}$ pin changes in function from a Data Strobe output ($\overline{DS}$) to a Float Command input ($\overline{FLT}$).

The NS32082 MMU will itself pull the CPU $\overline{AT}/\overline{SPC}$ pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the



TL/EE/5054–26

FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

3.0 Functional Description (Continued)

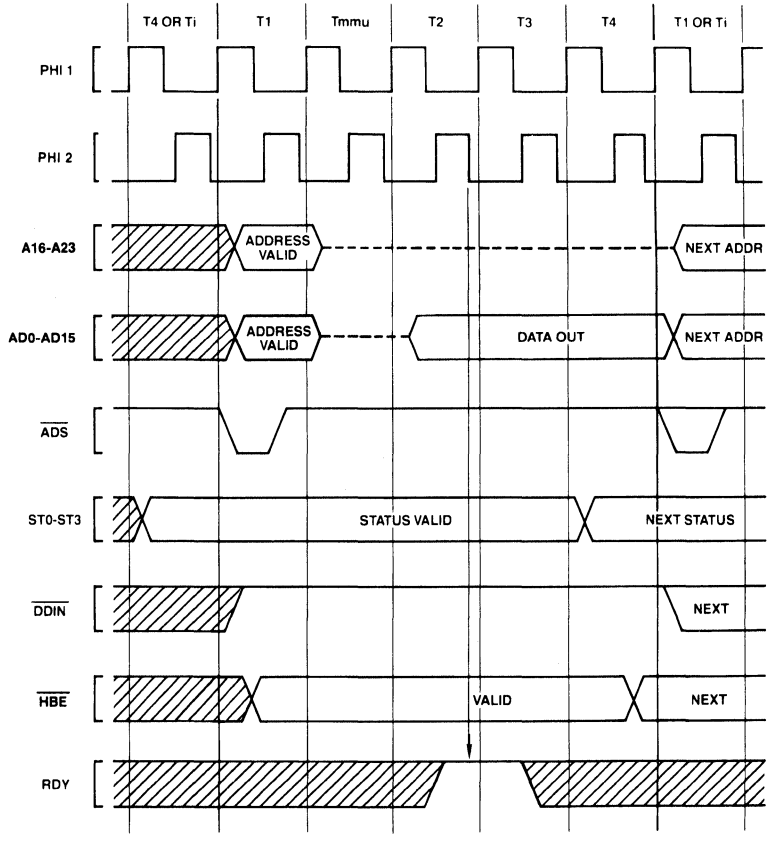
presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Section 2.1.3.

3.5.2 Translated Bus Timing

Figures 3-15 and *3-16* illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0–AD15 and A16–A23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{PAV}$. T2 through T4 of the cycle are identical to their counter-parts without Address Translation, with the exception that the CPU Address lines A16–A23 remain in the TRI-STATE condition. This allows the MMU to continue asserting the translated address on those pins.

Note that in order for the NS32082 MMU to operate correctly it must be set to the 32016 mode by forcing A24/$\overline{HBF}$ high during reset.

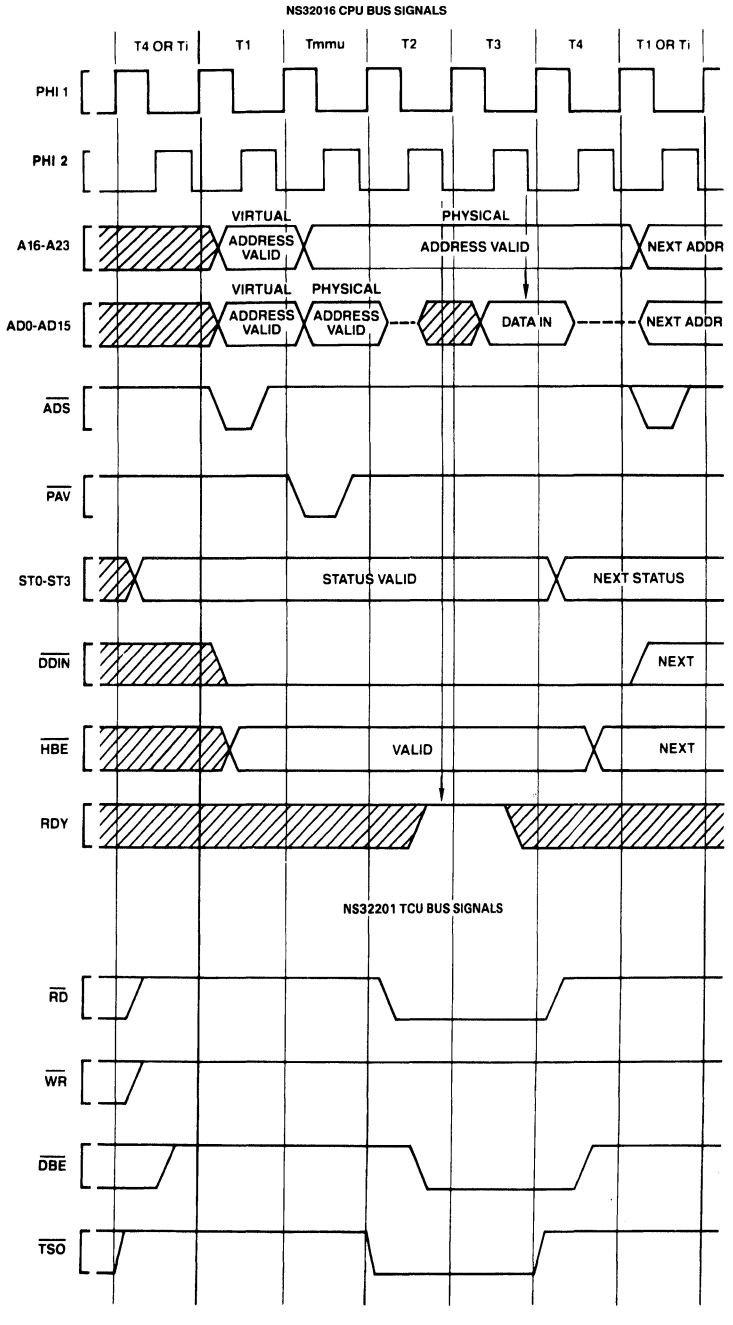
Figures 3-17 and *3-18* show a Read cycle and a Write cycle as generated by the 32016/32082/32201 group. Note that with the CPU $\overline{ADS}$ signal going only to the MMU, and with the MMU $\overline{PAV}$ signal substituting for $\overline{ADS}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.



TL/EE/5054–27

FIGURE 3-16. Write Cycle with Address Translation (CPU Action)

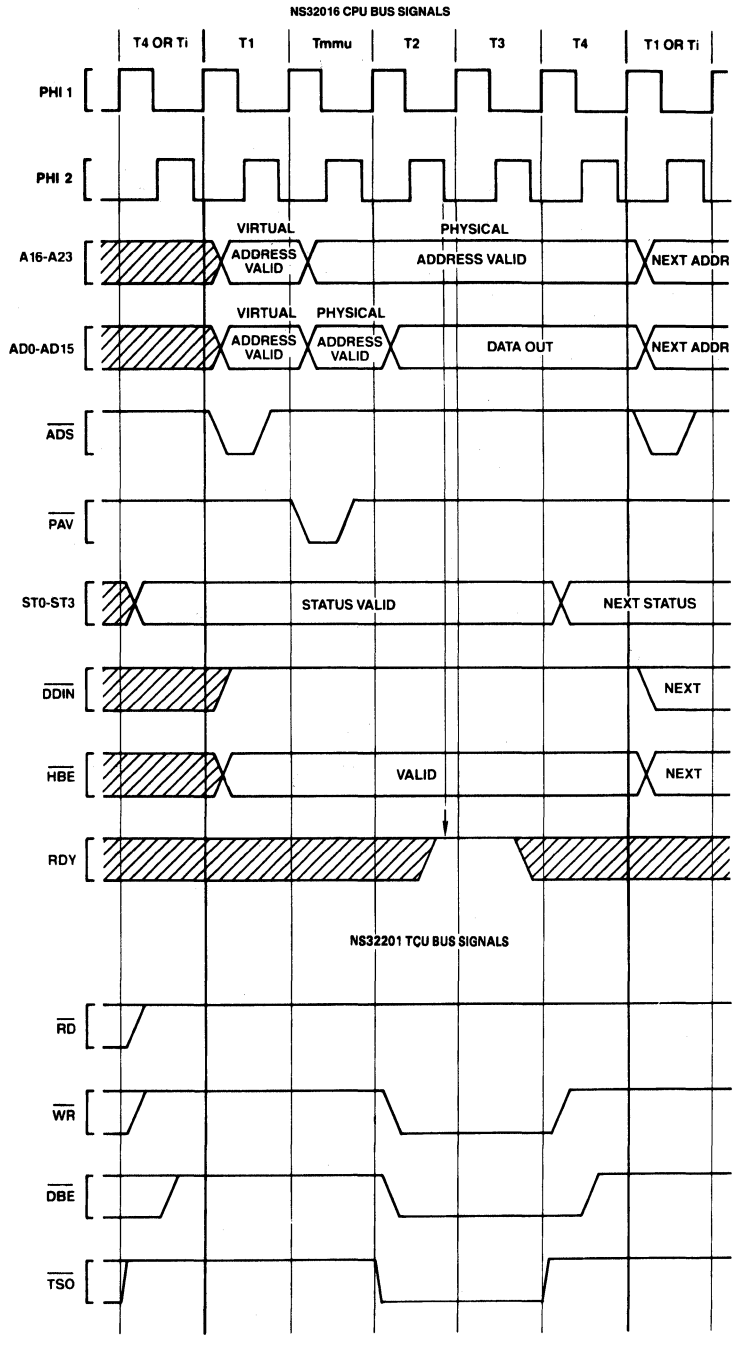
3.0 Functional Description (Continued)



TL/EE/5054-28

FIGURE 3-17. Memory-Managed Read Cycle

3.0 Functional Description (Continued)



TL/EE/5054–29

FIGURE 3-18. Memory-Managed Write Cycle

3.0 Functional Description (Continued)

3.5.3 The $\overline{FLT}$ (Float) Pin

The $\overline{FLT}$ pin is used by the CPU for address translation support. Activating $\overline{FLT}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its internal translation look.aside buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effects of $\overline{FLT}$. Upon sampling $\overline{FLT}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD15, A16–A23 and $\overline{DDIN}$ to the TRI-STATE condition ("floating").
2) Sets $\overline{HBE}$ low.
3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{RST}/\overline{ABT}$ description, Section 3.5.4.)

Note that the AD0–AD15 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until $\overline{FLT}$ again goes high. See the Timing Specifications, Section 4.

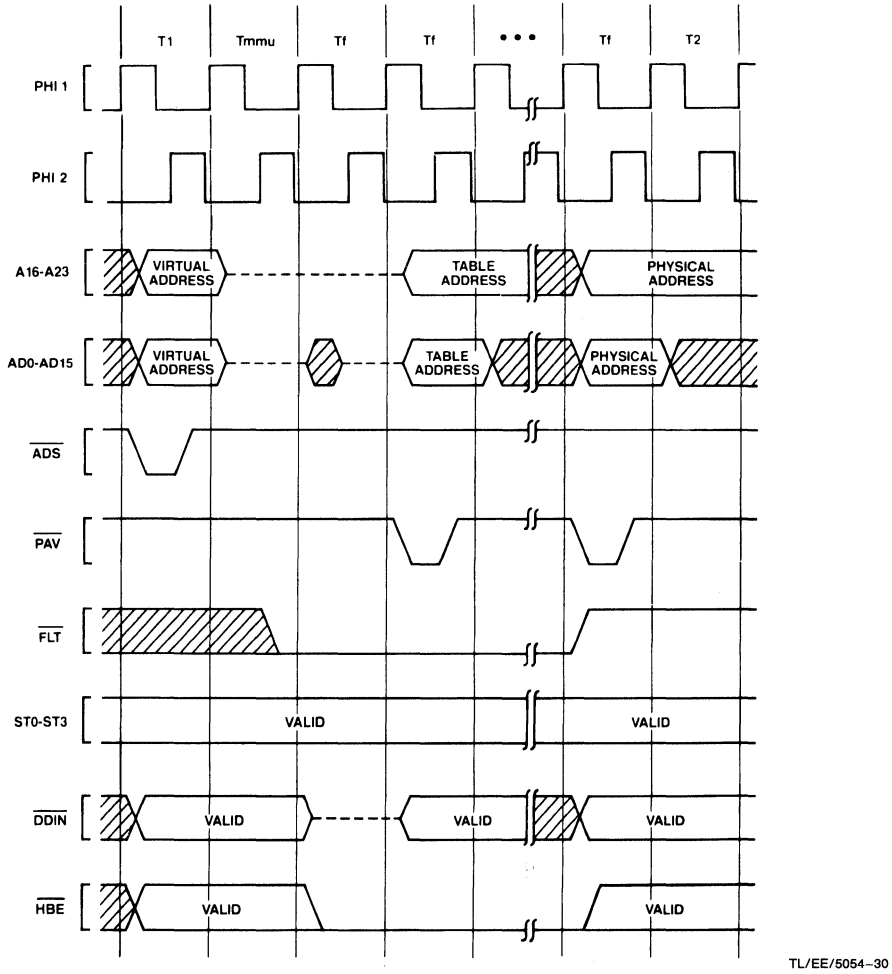


FIGURE 3-19. $\overline{FLT}$ Timing

3.0 Functional Description (Continued)

3.5.4 Aborting Bus Cycles

The $\overline{RST}/\overline{ABT}$ pin, apart from its Reset function (Section 3.3), also serves as the means to "abort," or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the $\overline{RST}/\overline{ABT}$ pin is held active for only one clock cycle.

If $\overline{RST}/\overline{ABT}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then Ti, thereby terminating the cycle. Since it is the MMU $\overline{PAV}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The reference page must be brought into physical memory from mass storage to make it accessible to the CPU.
2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Section 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

3.5.4.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the NS32082 Memory Management Unit.

1) If $\overline{FLT}$ has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, *Figure 4-22.*
2) If $\overline{FLT}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{FLT}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{FLT}$ is removed. See *Figure 4-23.*
3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management funtions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If $\overline{RST}/\overline{ABT}$ is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

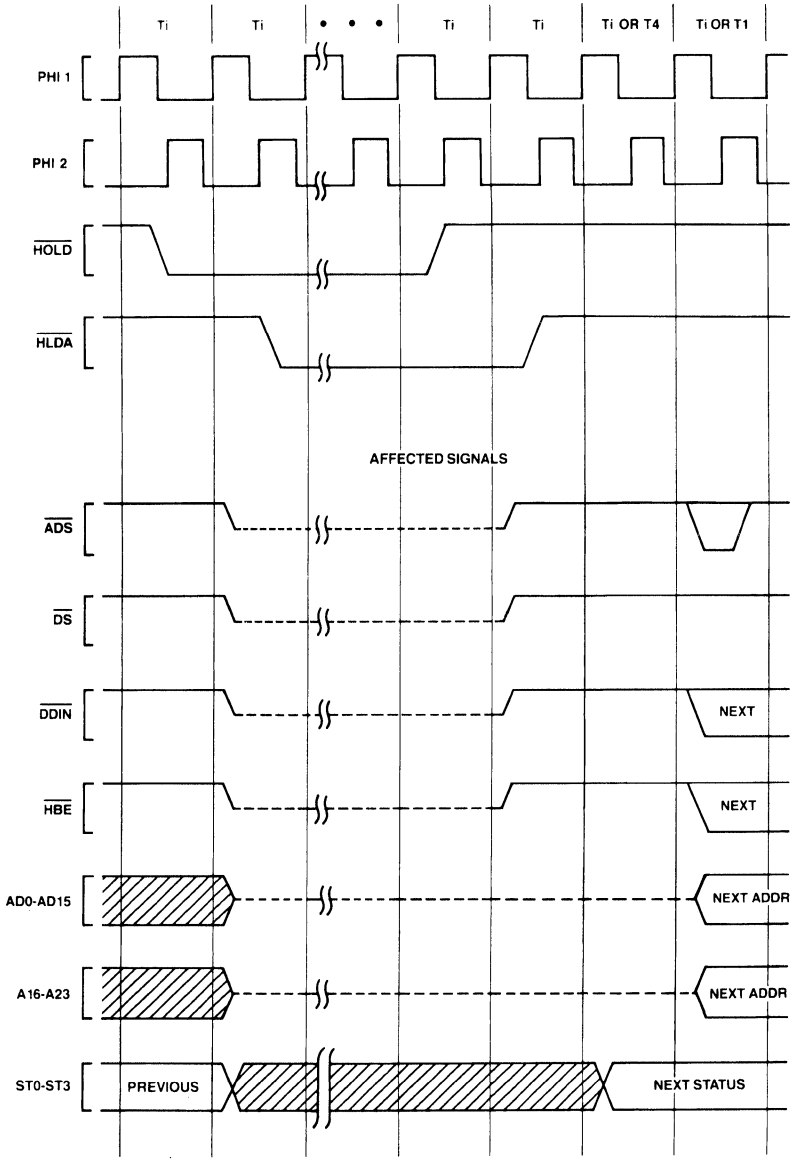
3.6 BUS ACCESS CONTROL

The NS32016 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{HOLD}$ (Hold Request) and $\overline{HLDA}$ (Hold Acknowledge) pins. By asserting $\overline{HOLD}$ low, an external device requests access to the bus. On receipt of $\overline{HLDA}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD15, A16–A23, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{HBE}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{HOLD}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{HLDA}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{HOLD}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-20* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-21* shows the sequence if the CPU is using the bus at the time that the $\overline{HOLD}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{HLDA}$ signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.

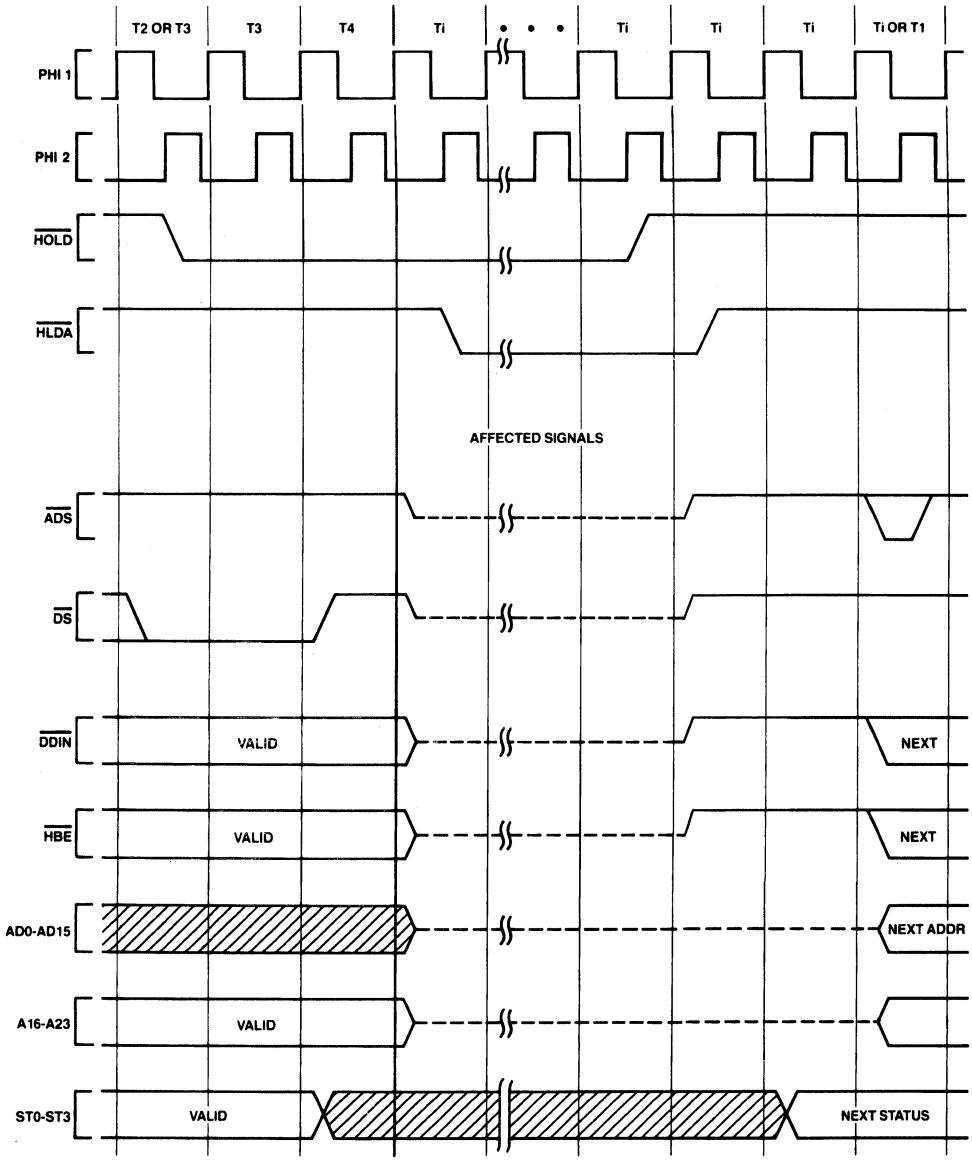
3.0 Functional Description (Continued)



TL/EE/5054–31

FIGURE 3-20. HOLD Timing, Bus Initially Idle

3.0 Functional Description (Continued)



TL/EE/5054–32

FIGURE 3-21. $\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32016 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/$\overline{S}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-21.*

$\overline{ILO}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing. As with the U/$\overline{S}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, *Figure 4-19.*

3.8 NS32016 INTERRUPT STRUCTURE

$\overline{INT}$, on which maskable interrupts may be requested,

$\overline{NMI}$, on which non-maskable interrupts may be requested, and

$\overline{RST}$/$\overline{ABT}$, which may be used to abort a bus cycle and any associated instruction. See Section 3.5.4.

In addition, there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

1) Adjustment of Registers.

   Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2) Saving Processor Status.

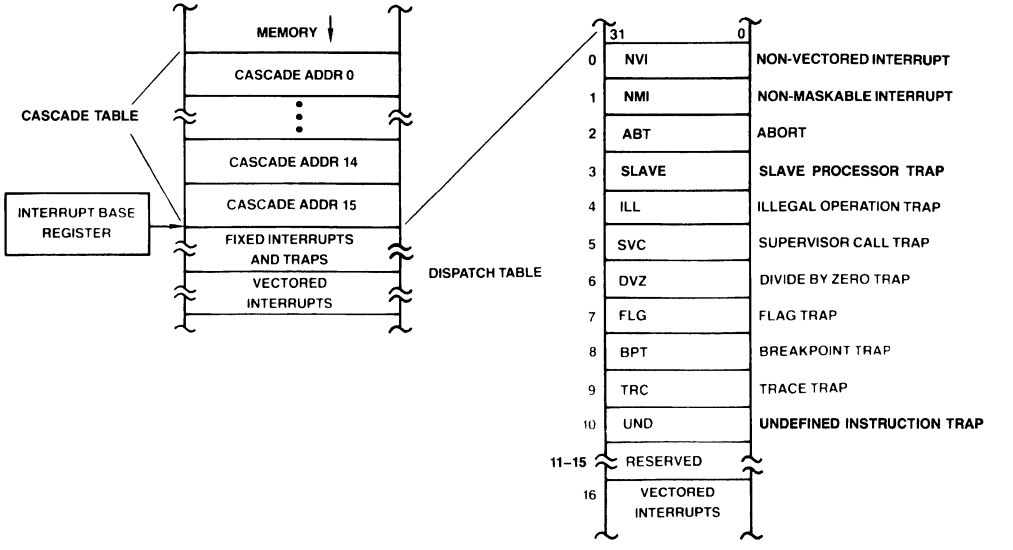
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

3) Vector Acquisition.

   A Vector is either obtained from the Data Bus or is supplied by default.

4) Service Call.

   The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-22.* A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.



TL/EE/5054–33

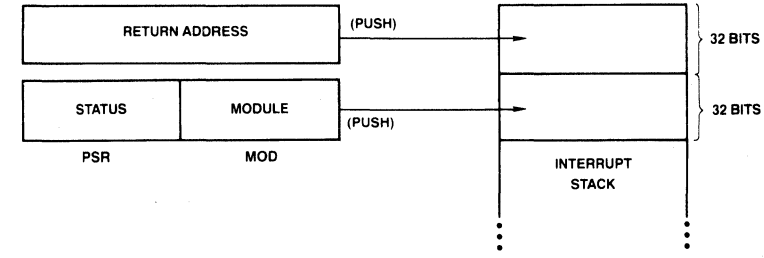
FIGURE 3-22. Interrupt Dispatch and Cascade Tables

3.0 Functional Description (Continued)

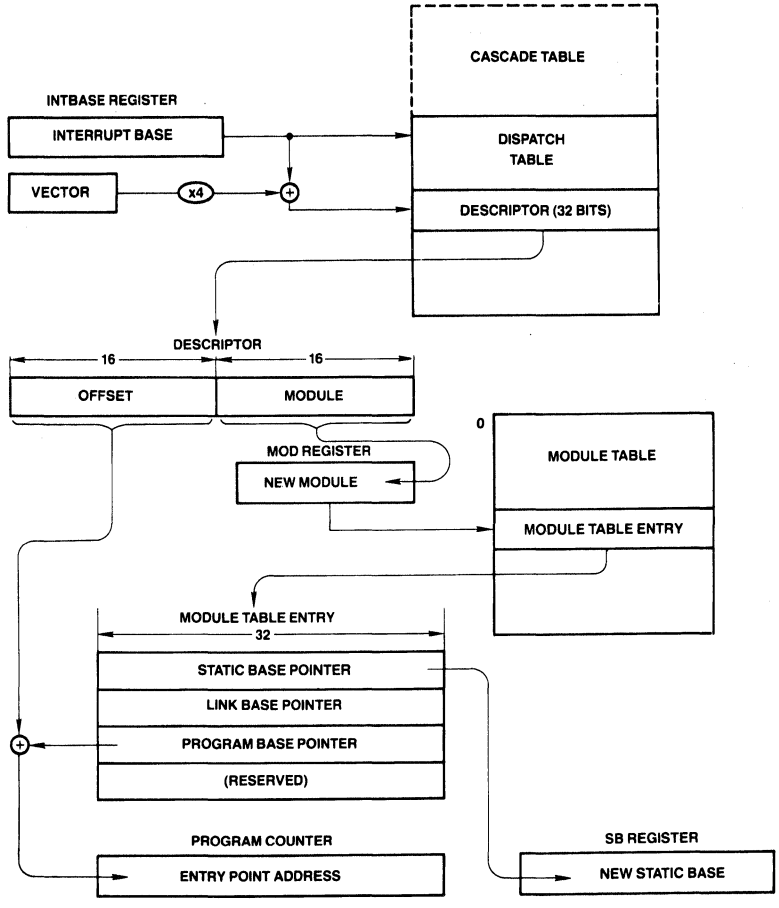
This process is illustrated in *Figure 3-23,* from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Section 3.8.7.1.
Abort Interrupt:	Section 3.8.7.4.
Traps (except Trace):	Section 3.8.7.2.
Trace Trap:	Section 3.8.7.3.



TL/EE/5054–34



TL/EE/5054–35

FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction (*Figure 3-24*) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-25.*

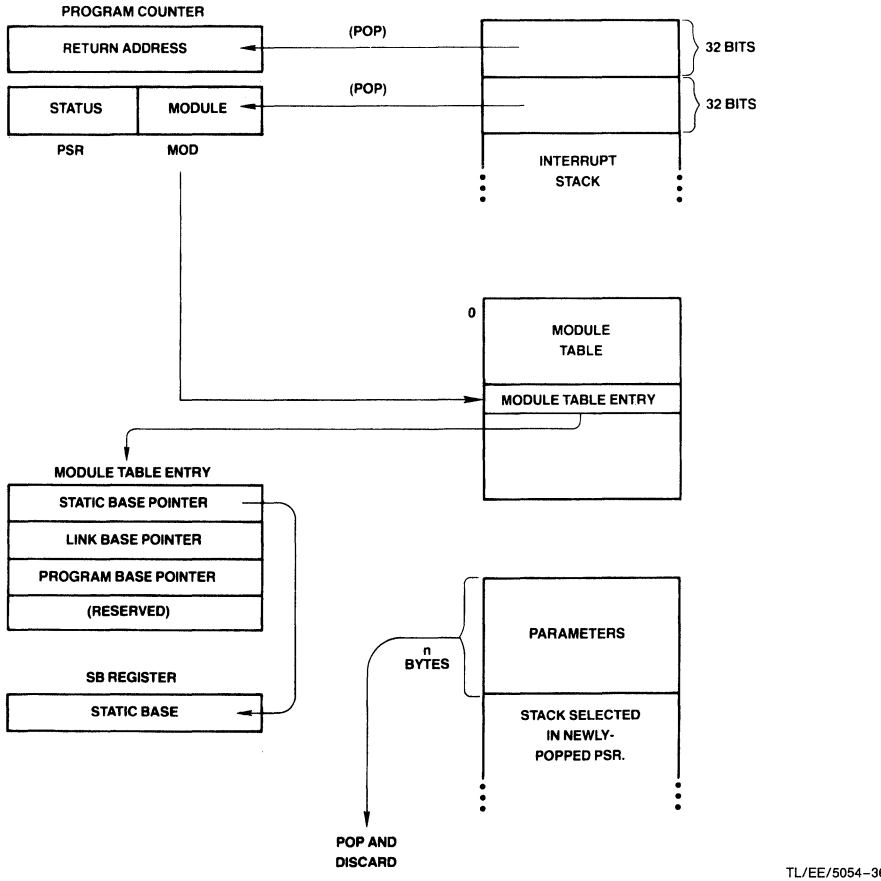
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$, $\overline{\text{NMI}}$ or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = 0) or Vectored (bit I = 1).

3.8.3.1 Non-Vectored Mode

In the Non-Vectored mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.



TL/EE/5054–36

FIGURE 3-24. Return from Trap (RETT n) Instruction Flow

3.0 Functional Description (Continued)

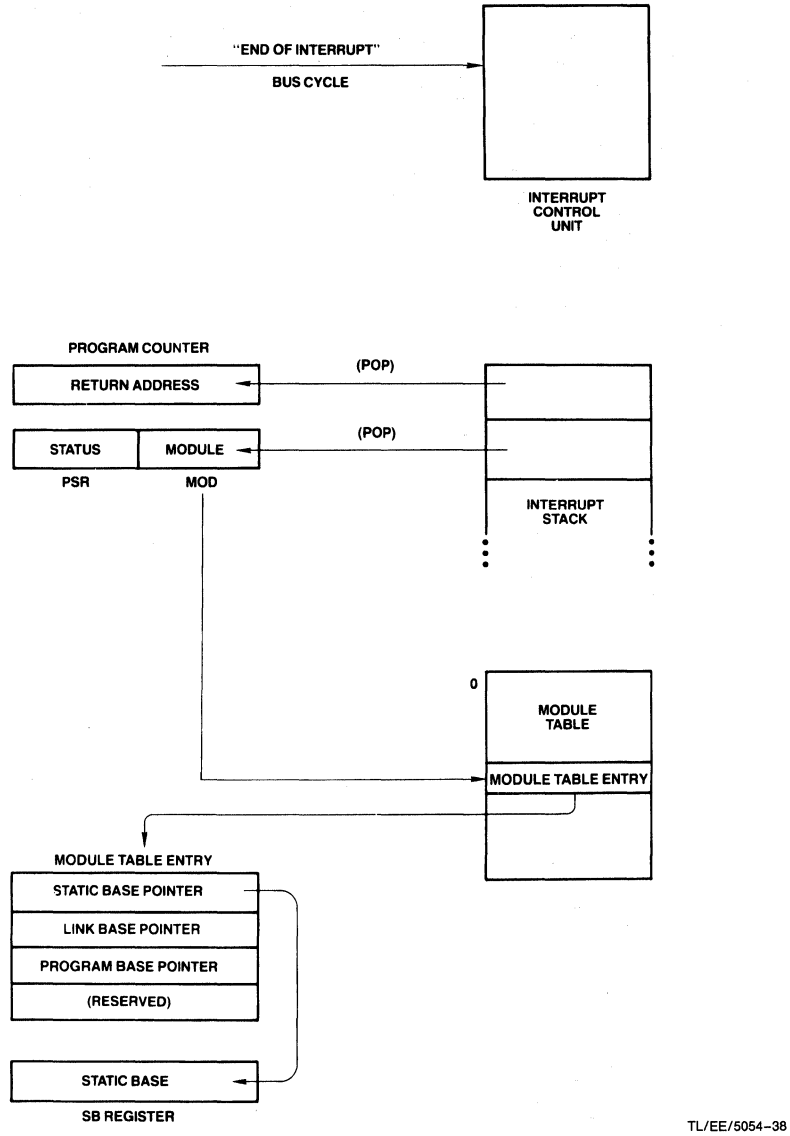


FIGURE 3-25. Return from Interrupt (RET) Instruction Flow

3.0 Functional Description (Continued)

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.8.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-27* shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{INT}$ pin.

In a system which uses cascading, two tasks must be performed upon initialization:

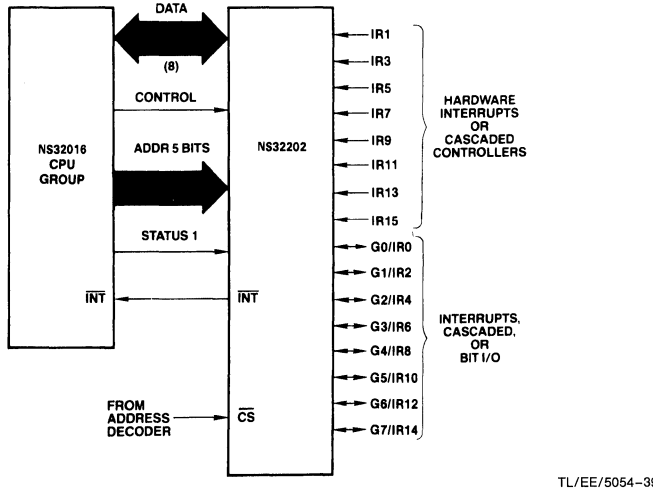
1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.
2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-22 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range −16 to −1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Section 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Section 3.4.2), whereupon the Master ICU again provides the negative Cascaded Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

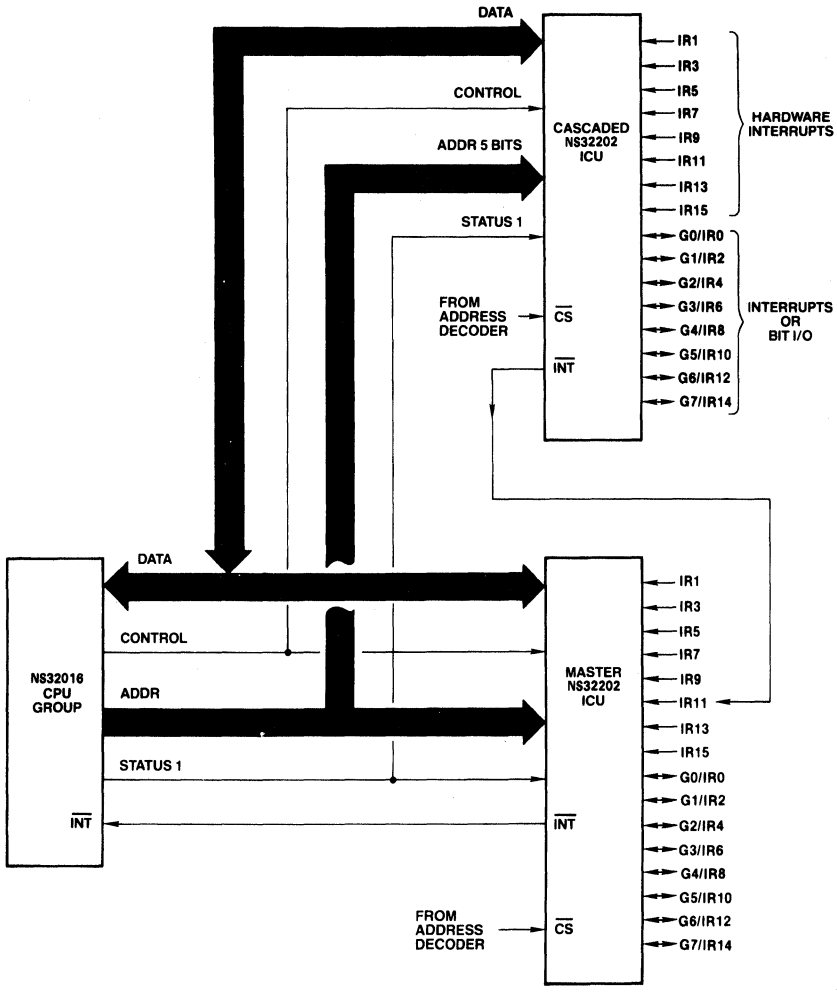
Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the interrupt mask register of the interrupt controller. However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.



TL/EE/5054-39

FIGURE 3-26. Interrupt Control Unit Connections (16 Levels)

3.0 Functional Description (Continued)



TL/EE/5054–40

FIGURE 3–27. Cascaded Interrupt Control Unit Connections

3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the $\overline{\text{NMI}}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is FFFF00_{16}. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Section 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) below is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by NS32016 CPU are:

Trap (Slave): An exceptional condition was detected by the Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Section 3.9.1).

3.0 Functional Description (Continued)

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

3.8.6 Prioritization

The NS32016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1) Traps other than Trace (Highest priority)
2) Abort
3) Non-Maskable Interrupt
4) Maskable Interrupts
5) Trace Trap (Lowest priority)

3.8.7 Interrupt/Trap Sequences: Detail Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-28.* Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequenced followed in processing either Maskable or Non-Maskable Interrupts (on the $\overline{INT}$ or $\overline{NMI}$ pins, respectively), see Section 3.8.7.1. For Abort interrupts, see Section 3.8.7.4. For the Trace Trap, see Section 3.8.7.3, and for all other traps see Section 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{NMI}$ pin receives a falling edge, or the $\overline{INT}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE + 4* Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Section 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-28.*

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector*4 + INTBASE Register contents.
2) Move the Module field of the Descriptor into the MOD Register.
3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
4) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.
5) Flush Queue: Non-sequentially fetch first instruction of Interrupt Routine.
6) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-28. Service Sequence
Invoked during all interrupt/trap sequences

3.0 Functional Description (Continued)

3.8.7.2 Trap Sequence: Traps Other Than Trace

1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2) Set "Vector" to the value corresponding to the trap type.

SLAVE:	Vector = 3.
ILL:	Vector = 4.
SVC:	Vector = 5.
DVZ:	Vector = 6.
FLG:	Vector = 7.
BPT:	Vector = 8.
UND:	Vector = 10.

3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Return Address" to the address of the first byte of the trapped instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

3.8.7.3 Trace Trap Sequence

1) In the Processor Status Register (PSR), clear the P bit.
2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4) Set "Vector" to 9.
5) Set "Return Address" to the address of the next instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

3.8.7.4 Abort Sequence

1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
2) Clear the PSR P bit.
3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set "Vector" to 2.
6) Set "Return Address" to the address of the first byte of the aborted instruction.
7) Perform Service (Vector, Return Address), *Figure 3-28.*

3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32016 CPU recognizes three groups of instructions as being executable by external Slave Processors:

Floating Point Instruction Set

Memory Management Instruction Set

Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Section 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

3.9.1 Slave Processor Protocol

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.
2) It specifies which Slave Processor will execute it.
3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-29.* While applying Status Code 1111 (Broadcast ID, Section 3.4.2), the CPU transfers the ID Byte on the least-significant half of the Data Bus (AD0–AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Section 3.4.2).

Status Combinations:
Send ID (ID): Code 1111
Xfer Operand (OP): Code 1101
Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operation Word.
3	OP	CPU Sends Required Operands.
4	—	Slave Starts Execution. CPU Pre-Fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-29. Slave Processor Protocol

3.0 Functional Description (Continued)

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this, and for the Address Translation strap function, $\overline{AT}/\overline{SPC}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Section 3.4.2).

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Section 3.4.2). This word has the format shown in *Figure 3-30.* If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the Slave vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.9.2 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Series 32000 Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-30).*

TABLE 3-4
Floating Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note:

D = Double Word

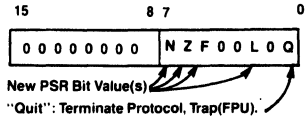
i = integer size (B,W,D) specified in mnemonic.

c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)



TL/EE/5054-41

FIGURE 3-30. Slave Processor Status Word Format

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

3.9.3 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Series 32000 Instruction Set Reference Manual and the NS32082 MMU Data Sheet.

TABLE 3-5.
Memory Management Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Series 32000 Instruction Set Reference Manual and the NS32082 Memory Management Unit Data Sheet.

D = Double Word

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.0 Functional Description (Continued)

3.9.4 Custom Slave Instructions

Provided in the NS32016 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type 'c' will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

TABLE 3-6.
Custom Slave Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op. 2	none
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	readi	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

NOTE:

D = Double Word

i = integer size (B,W,D) specified in mnemonic.

f = Floating Point type (F,L) specified in mnemonic.

N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 PIN DESCRIPTIONS

The following is a brief description of all NS32016 pins. The descriptions reference portions of the Functional Description, Section 3.

4.1.1 Supplies

Power (V_{CC}): +5V positive supply. Section 3.1

Logic Ground (GNDL): Ground reference for on-chip logic. Section 3-1.

Buffer Ground (GNDB): Ground reference for on-chip drivers connected to output pins. Section 3.1.

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Section 3.1.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Section 3.2.

Ready (RDY): Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Section 3.4.1.

Hold Request ($\overline{\text{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.6.

Note: If the $\overline{\text{HOLD}}$ signal is generated asynchronously, it's set up and hold times may be violated.

In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{\text{HLDA}}$ latency. This is to avoid speed degradations in cases of heavy $\overline{\text{HOLD}}$ activity (i.e. DMA controller cycles interleaved with CPU cycles.)

Interrupt ($\overline{\text{INT}}$): Active low, Maskable interrupt request. Section 3.8.

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low, Non-Maskable interrupt request. Section 3.8.

Reset/Abort ($\overline{\text{RST}}$/$\overline{\text{ABT}}$): Active low. If held active for one clock cycle and released, this pin causes an Abort Command, Section 3.5.4. If held longer, it initiates a Reset, Section 3.3.

4.1.3 Output Signals

Address Bits 16–23 (A16–A23): These are the most significant 8 bits of the memory address bus. Section 3.4.

Address Strobe ($\overline{\text{ADS}}$): Address low. Controls address latches; indicates start of a bus cycle. Section 3.4.

Data Direction In ($\overline{\text{DDIN}}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Section 3.4.

High Byte Enable ($\overline{\text{HBE}}$): Active low. Status signal enabling transfer on the most significant byte of the Data Bus. Section 3.4; Section 3.4.3.

Note: The $\overline{\text{HBE}}$ signal is normally floated when the CPU grants the bus in response to a DMA request on the $\overline{\text{HOLD}}$ pin.

However, when an MMU is used and the bus is granted during an MMU page table look-up, $\overline{\text{HBE}}$ is not floated since the CPU does not have sufficient information to synchronize the release of $\overline{\text{HBE}}$ to the MMU's bus cycles.

Therefore, in a memory managed system, an external TRI-STATE buffer is required. This is shown in Figure B-1 in Appendix B.

Status (ST0–ST3): Active high. Bus cycle status code, ST0 least significant. Section 3.4.2. Encodings are:

0000 — Idle: CPU Inactive on Bus.
0001 — Idle: WAIT Instruction.
0010 — (Reserved)
0011 — Idle: Waiting for Slave.
0100 — Interrupt Acknowledge, Master.
0101 — Interrupt Acknowledge, Cascaded.
0110 — End of Interrupt, Master.
0111 — End of Interrupt, Cascaded.
1000 — Sequential Instruction Fetch.
1001 — Non-Sequential Instruction Fetch.
1010 — Data Transfer.
1011 — Read Read-Modify-Write Operand.
1100 — Read for Effective Address.
1101 — Transfer Slave Operand.
1110 — Read Slave Status Word.
1111 — Broadcast Slave ID.

Hold Acknowledge ($\overline{\text{HLDA}}$): Active low. Applied by the CPU isn response to $\overline{\text{HOLD}}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Section 3.7.

User/Superior (U/$\overline{\text{S}}$): User or Supervisor Mode status. Section 3.7. High state indicates User Mode, low indicates Supervisor Mode. Section 3.7.

Interlocked Operation ($\overline{\text{ILO}}$): Active low. Indicates that an interlocked instruction is being executed. Section 3.7.

Program Flow Status ($\overline{\text{PFS}}$): Active low. Pulse indicates beginning of an instruction execution. Section 3.7.

4.1.4 Input/Output Signals

Address/Data 0–15 (AD0–AD15): Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Section 3.4.

Address Translation/Slave Processor Control ($\overline{\text{AT}}$/$\overline{\text{SPC}}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of a slave instruction. Section 3.4.6. Section 3.9. Sampled on the rising edge of Reset as Address Translation Strap. Section 3.5.1.

In Non-Memory-Managed systems this pin should be pulled-up to V_{CC} through a 10 kΩ resistor.

Data Strobe/Float ($\overline{\text{DS}}$/$\overline{\text{FLT}}$): Active low. Data Strobe output, Section 3.4, or Float Command input, Section 3.5.3. Pin function is selected on $\overline{\text{AT}}$/$\overline{\text{SPC}}$ pin, Section 3.5.1.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages With Respect to GND	−0.5V to +7V
Power Dissipation	1.5 Watt

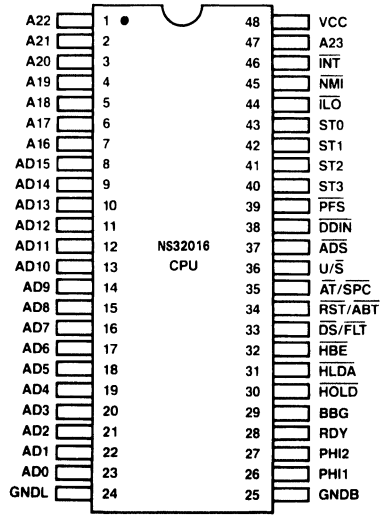
Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS T_A = 0 to +70°C, V_{CC} = 5V ±5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC}+0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC}-0.35$		$V_{CC}+0.5$	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	I_{OH} = −400 μA	2.4			V
V_{OL}	Logical 0 Output Voltage	I_{OL} = 2 mA			0.45	V
I_{ILS}	$\overline{AT}/\overline{SPC}$ Input Current (low)	V_{IN} = 0.4V, $\overline{AT}/\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Leakage Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current Output Pins in TRI-STATE condition	$0.4 \le V_{OUT} \le V_{CC}$	−20		30	μA
I_{CC}	Active Supply Current	I_{OUT} = 0, T_A = 25°C		200	300	mA

Connection Diagram

Dual-In-Line Package



TL/EE/5054–2

Top View

FIGURE 4-1

Order Number NS32016D or NS32016N
See NS Package Number D48A or N48A

4.0 Device Specifications (Continued)

4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 and PHI2 and 0.8V or 2.0V on all other signals as illustrated in *Figures 4-2* and *4-3*, unless specifically stated otherwise.

ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge

T.E. — trailing edge F.E. — falling edge

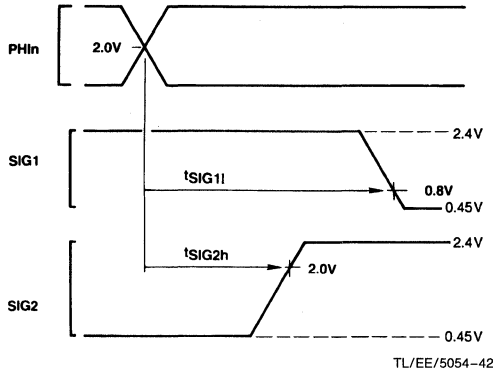


FIGURE 4-2. Timing Specification Standard (Signal Valid After Clock Edge)

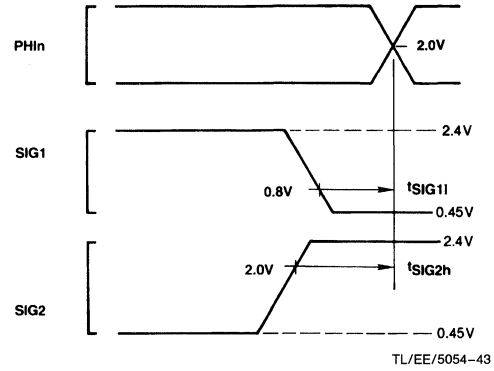


FIGURE 4-3. Timing Specification Standard (Signal Valid Before Clock Edge)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32016-6, NS32016-8 and NS32016-10

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4-4	Address bits 0–15 valid	after R.E., PHI1 T1		65		55		40	ns
t_{ALh}	4-4	Address bits 0–15 hold	after R.E., PHI1 Tmmu or T2	5		5		5		ns
t_{Dv}	4-4	Data valid (write cycle)	after R.E., PHI1 T2		70		60		50	ns
t_{Dh}	4-4	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{AHv}	4-4	Address bits 16–23 valid	after R.E., PHI1 T1		65		55		40	ns
t_{AHh}	4-4	Address bits 16–23 hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	4-5	Address bits 0–15 setup	before $\overline{ADS}$ T.E.	25		25		25		ns
t_{AHADSs}	4-5	Address bits 16–23 set up	before $\overline{ADS}$ T.E.	25		25		25		ns
t_{ALADSh}	4-9	Address bits 0–15 hold	after $\overline{ADS}$ T.E.	20		20		15		ns
t_{AHADSh}	4-9	Address bits 16–23 hold	after $\overline{ADS}$ T.E.	20		20		15		ns
t_{ALf}	4-5	Address bits 0–15 floating	after R.E., PHI1 T2 (no MMU)		25		25		25	ns
t_{ALMf}	4-9	Address bits 0–15 floating	after R.E., PHI1 Tmmu (with MMU)		25		25		25	ns
t_{AHMf}	4-9	Address bits 16–23 floating	after R.E., PHI1 Tmmu (with MMU)		25		25		25	ns
t_{HBEv}	4-4	$\overline{HBE}$ signal valid	after R.E., PHI1 T1		70		60		50	ns
t_{HBEh}	4-4	$\overline{HBE}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{STv}	4-4	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		65		55		45	ns
t_{STh}	4-4	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	4-5	$\overline{DDIN}$ signal valid	after R.E., PHI1 T1		83		62		50	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32016-6, NS32016-8 and NS32016-10 (Continued)

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t_{DDINh}	4-5	$\overline{DDIN}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	4-4	$\overline{ADS}$ signal active (low)	after R.E., PHI1 T1		55		45		35	ns
t_{ADSia}	4-4	$\overline{ADS}$ signal inactive	after R.E., PHI2 T1	15	60	15	55	15	45	ns
t_{ADSw}	4-4	$\overline{ADS}$ pulse width	at 0.8V (both edges)	60		48		35		ns
t_{DSa}	4-4	$\overline{DS}$ signal active (low)	after R.E., PHI1 T2		70		60		45	ns
t_{DSia}	4-4	$\overline{DS}$ signal inactive	after R.E., PHI1 T4	10	60	10	50	10	40	ns
t_{ALf}	4-6	AD0–AD15 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{AHf}	4-6	A16–A23 floating (caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{DSf}	4-6	$\overline{DS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{ADSf}	4-6	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HBEf}	4-6	$\overline{HBE}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-6	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HLDAa}	4-6	$\overline{HLDA}$ signal active (low)	after R.E., PHI1 Ti		100		90		75	ns
t_{HLDAia}	4-8	$\overline{HLDA}$ signal inactive	after R.E., PHI1 Ti		100		90		75	ns
t_{DSr}	4-8	$\overline{DS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{ADSr}	4-8	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HBEr}	4-8	$\overline{HBE}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINr}	4-8	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-9	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		80		65		50	ns
t_{HBEl}	4-9	$\overline{HBE}$ signal low (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		100		85		65	ns
t_{DDINr}	4-10	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		75		65		50	ns
t_{HBEr}	4-10	$\overline{HBE}$ signal returns from LOW (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		90		85		75	ns
t_{SPCa}	4-13	$\overline{SPC}$ output active (low)	after R.E., PHI1 T1		50		45		35	ns
t_{SPCia}	4-13	$\overline{SPC}$ output inactive	after R.E., PHI1 T4		50		45		35	ns
t_{SPCnf}	4-15	$\overline{SPC}$ output nonforcing	after R.E., PHI2 T4		40		25		10	ns
t_{Dv}	4-13	Data valid (slave processor write)	after R.E., PHI1 T1		80		65		50	ns
t_{Dh}	4-13	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	4-18	$\overline{PFS}$ pulse width	at 0.8V (both edges)	70		70		70		ns
t_{PFSa}	4-18	$\overline{PFS}$ pulse active (low)	after R.E., PHI2		70		60		50	ns
t_{PFSia}	4-18	$\overline{PFS}$ pulse inactive	after R.E., PHI2		70		60		50	ns
t_{ILOs}	4-20a	$\overline{ILO}$ signal setup	before R.E., PHI1 T1 of first interlocked write cycle	30		30		30		ns
t_{ILOh}	4-20b	$\overline{ILO}$ signal hold	after R.E., PHI1 T3 of last interlocked read cycle	10		10		10		ns
t_{ILOa}	4-21	$\overline{ILO}$ signal active (low)	after R.E., PHI1		70		65		55	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32016-6, NS32016-8 and NS32016-10 (Continued)

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ILOia}	4-21	$\overline{ILO}$ signal inactive	after R.E., PHI1		55		45		35	ns
t_{USv}	4-22	U/$\overline{S}$ signal valid	after R.E., PHI1 T4		55		45		35	ns
t_{USh}	4-22	U/$\overline{S}$ signal hold	after R.E., PHI1 T4	10		10		8		ns
t_{NSPF}	4-19b	Nonsequential fetch to next $\overline{PFS}$ clock cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	4-19a	$\overline{PFS}$ clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	4-29	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t_{Cp}

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

4.4.2.2 Input Signal Requirements: NS32016-6, NS32016-8 and NS32016-10

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t_{PWR}	4-25	Power stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	4-5	Data in setup (read cycle)	before F.E., PHI2 T3	25		20		15		ns
t_{DIh}	4-5	Data in hold (read cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	4-6	$\overline{HOLD}$ active (low) setup time (see note)	before F.E., PHI2 TX1	25		25		25		ns
t_{HLDia}	4-8	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	4-6	$\overline{HOLD}$ hold time	after R.E., PHI1 TX2	0		0		0		ns
t_{FLTa}	4-9	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25		25		25		ns
t_{FLTia}	4-11	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T2	25		25		25		ns
t_{RDYs}	4-11, 4-12	RDY setup time	before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	4-11, 4-12	RDY hold time	after F.E., PHI1 T3	0		0		0		ns
t_{ABTs}	4-23	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t_{ABTs}	4-24	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 T_f	30		25		20		ns
t_{ABTh}	4-23	$\overline{ABT}$ hold time	after R.E., PHI1	0		0		0		ns
t_{RSTs}	4-25, 4-26	$\overline{RST}$ setup time	before F.E., PHI1	20		20		15		ns
t_{RSTw}	4-26	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	4-27	$\overline{INT}$ setup time	before F.E., PHI1	25		25		25		ns
t_{NMIw}	4-28	$\overline{NMI}$ pulse width	at 0.8V (both edges)	80		75		70		ns
t_{DIs}	4-14	Data setup (slave read cycle)	before F.E., PHI2 T1	30		25		20		ns
t_{DIh}	4-14	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	4-15	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	40		35		25		ns
t_{SPCs}	4-15	$\overline{SPC}$ setup time	before F.E., PHI1	35		30		25		ns
t_{SPCw}	4-15	$\overline{SPC}$ pulse width from slave processor (async.input)	at 0.8V (both edges)	30		25		20		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements: NS32016-6, NS32016-8 and NS32016-10 (Continued)

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ATh}	4-16	$\overline{AT}/\overline{SPC}$ hold for address translation strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		t_{Cp}
t_{ATs}	4-16	$\overline{AT}/\overline{SPC}$ setup for address translation strap	before R.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1		1		1		t_{Cp}

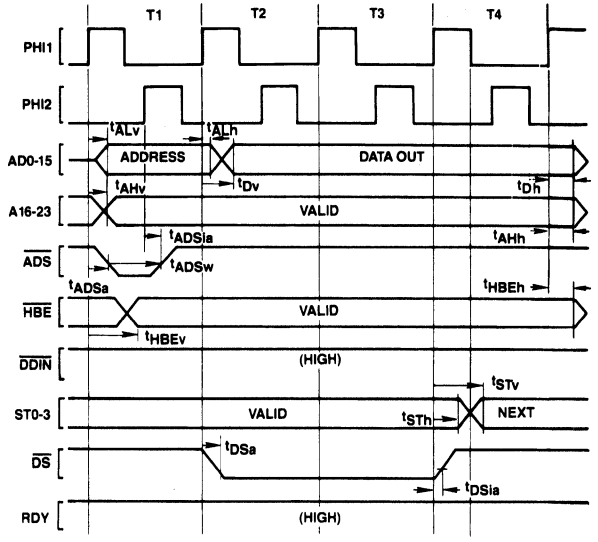
Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

4.2.3 Clocking Requirements: NS32016-6, NS32016-8 and NS32016-10

Name	Figure	Description	Reference/ Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4-17	PHI1, PHI2 rise time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	4-17	PHI1, PHI2 fall time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	4-17	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4-17	PHI1, PHI2 pulse width	At 2.0V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 14		0.5 t_{Cp} − 12		0.5 t_{Cp} − 10		ns
$t_{CLh(1,2)}$	4-17	PHI1, PHI2 high time	at V_{CC} − 0.9V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 18		0.5 t_{Cp} − 17		0.5 t_{Cp} − 15		ns
$t_{nOVL(1,2)}$	4-17	Non-overlap time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}		PHI1, PHI2 asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

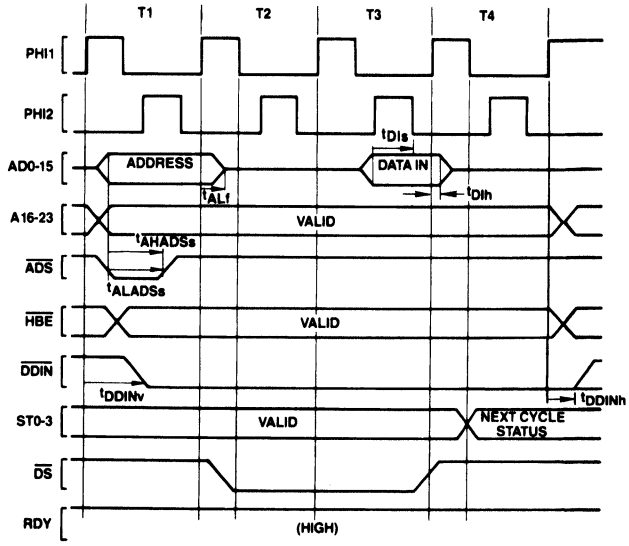
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



TL/EE/5054–44

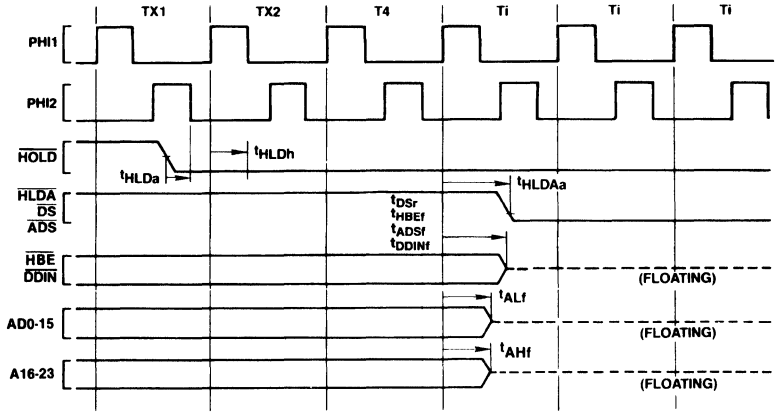
FIGURE 4-4. Write Cycle



TL/EE/5054–45

FIGURE 4-5. Read Cycle

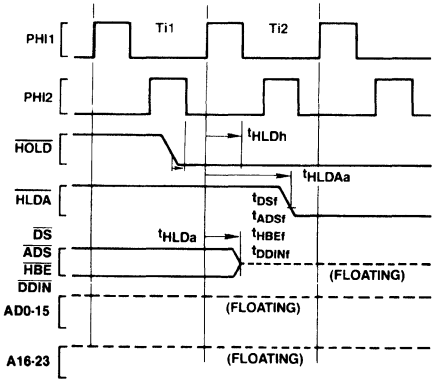
4.0 Device Specifications (Continued)



TL/EE/5054-46

FIGURE 4-6. Floating by $\overline{HOLD}$ Timing (CPU Not Idle Initially)

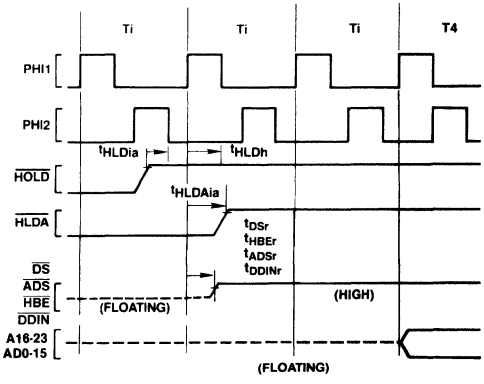
Note that whenever the CPU is not idling (not in Ti), the $\overline{HOLD}$ request ($\overline{HOLD}$ low) must be active t_{HLDa} before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until t_{HLDh} after the rising edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.



TL/EE/5054-47

FIGURE 4-7. Floating by $\overline{HOLD}$ Timing (CPU Initially Idle)

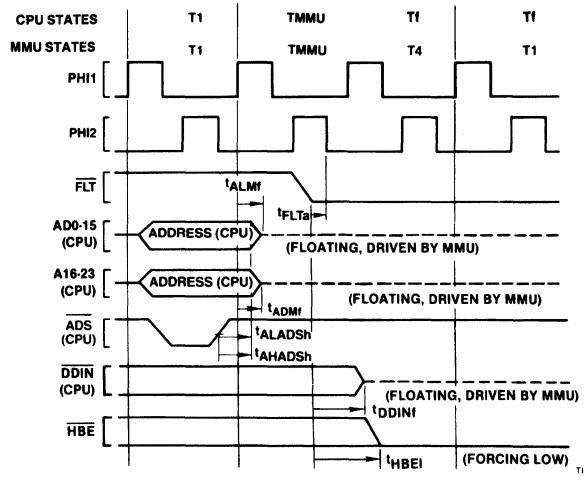
Note that during Ti1 the CPU is already idling.



TL/EE/5054-48

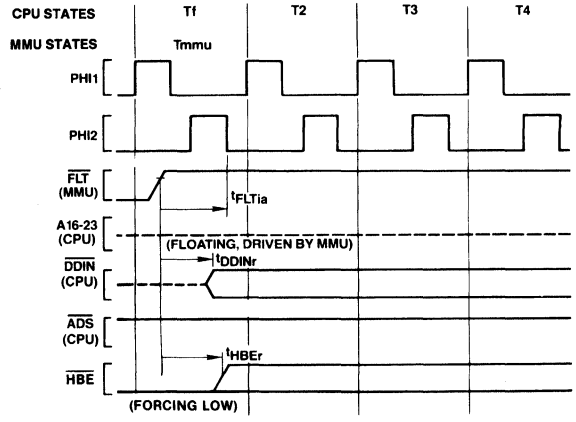
FIGURE 4-8. Release from $\overline{HOLD}$

4.0 Device Specifications (Continued)



TL/EE/5054-49

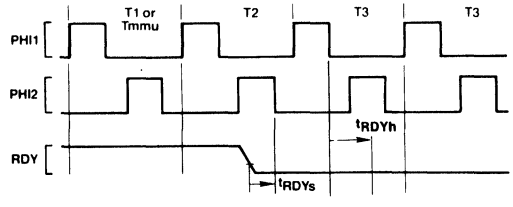
FIGURE 4-9. FLT Initiated Cycle Timing



TL/EE/5054-50

Note that when FLT is deasserted the CPU restarts driving DDIN before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force DDIN to the same logic level.

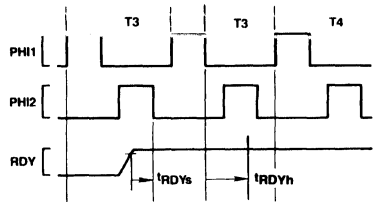
FIGURE 4-10. Release from FLT Timing



TL/EE/5054-51

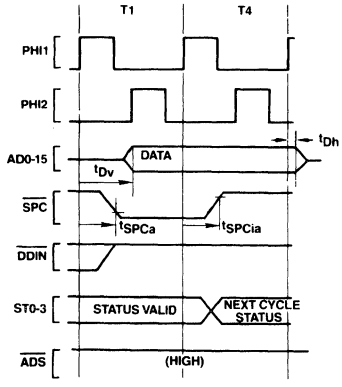
FIGURE 4-11. Ready Sampling (CPU Initially READY)

4.0 Device Specifications (Continued)



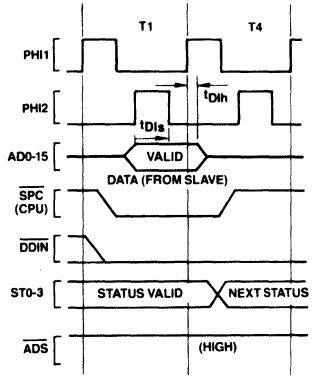
TL/EE/5054-52

FIGURE 4-12. Ready Sampling (CPU Initially NOT READY)



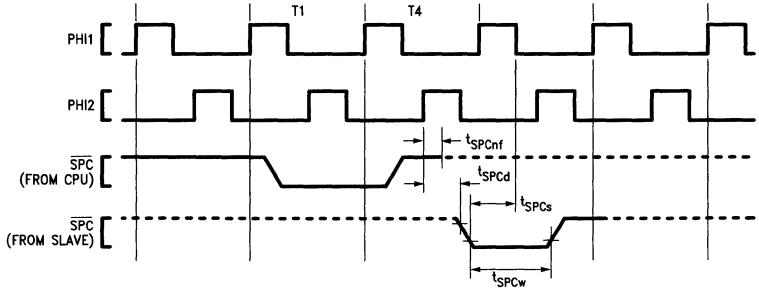
TL/EE/5054-53

FIGURE 4-13. Slave Processor Write Timing



TL/EE/5054-54

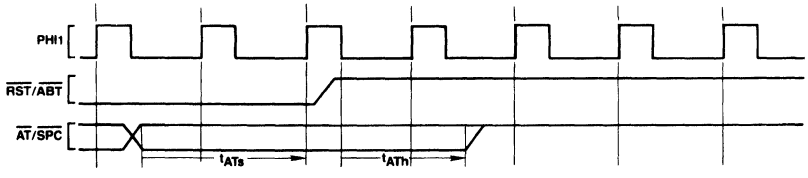
FIGURE 4-14. Slave Processor Read Timing



TL/EE/5054-82

FIGURE 4-15. $\overline{SPC}$ Timing

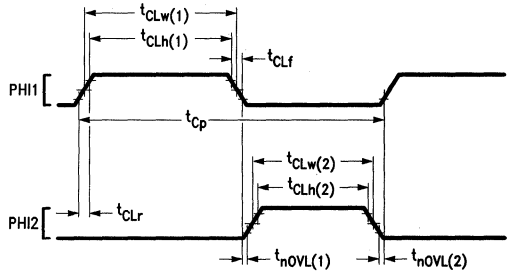
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{SPC}$ high with internal 5 kΩ pullup.



TL/EE/5054-56

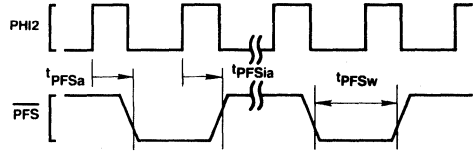
FIGURE 4-16. Reset Configuration Timing

4.0 Device Specifications (Continued)



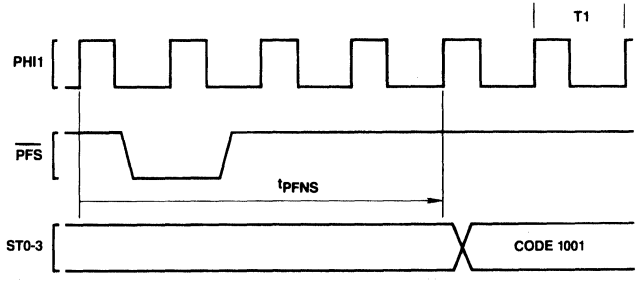
TL/EE/5054–57

FIGURE 4-17. Clock Waveforms



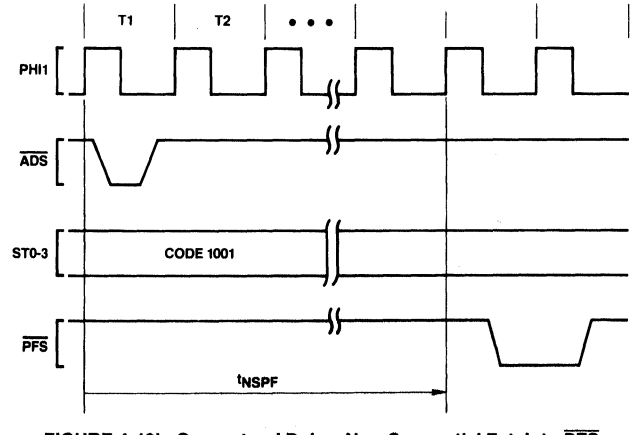
TL/EE/5054–58

FIGURE 4-18. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/5054–59

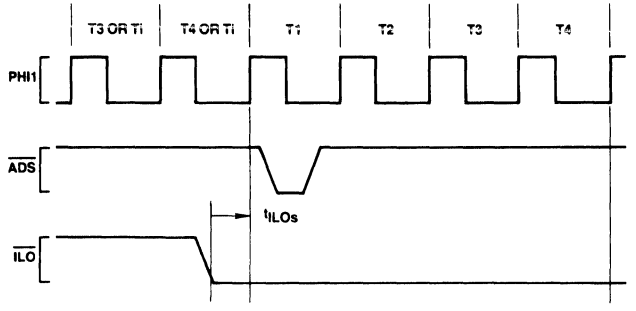
FIGURE 4-19a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/5054–60

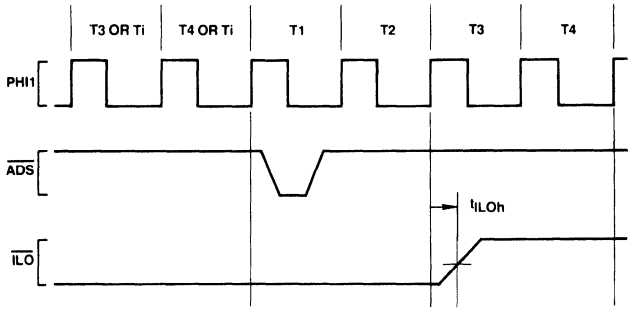
FIGURE 4-19b. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



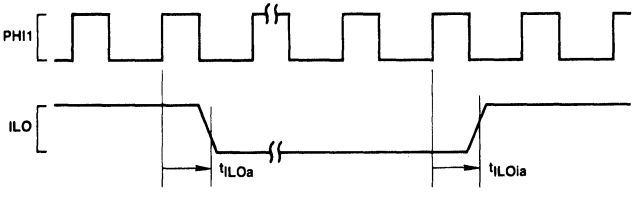
TL/EE/5054-61

FIGURE 4-20a. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction



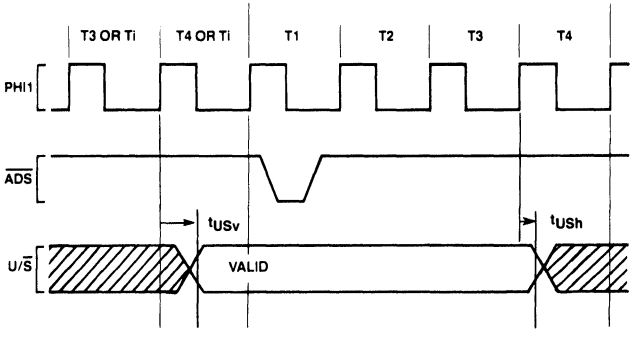
TL/EE/5054-62

FIGURE 4-20b. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction



TL/EE/5054-63

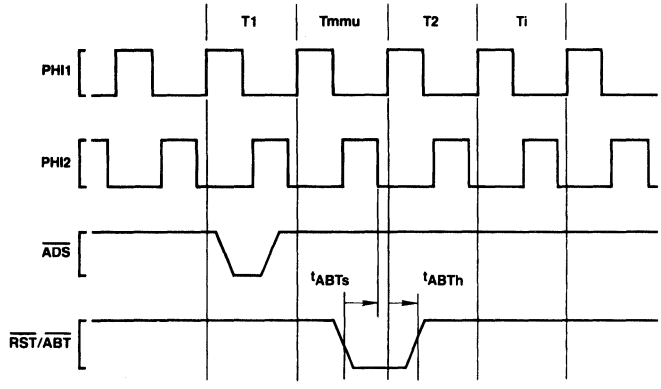
FIGURE 4-21. Relationship of $\overline{ILO}$ to Any Clock Cycle



TL/EE/5054-64

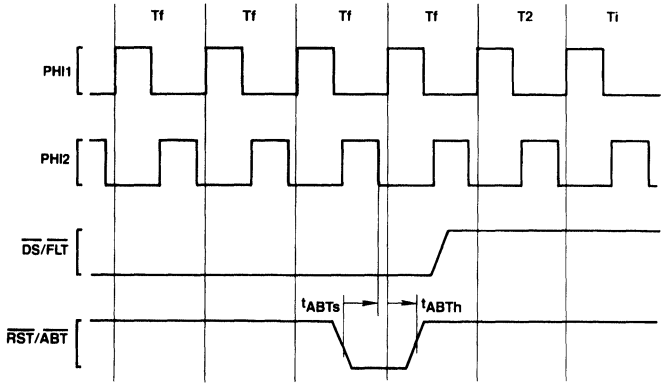
FIGURE 4-22. U/$\overline{S}$ Relationship to Any Bus Cycle— Guaranteed Valid Interval

4.0 Device Specifications (Continued)



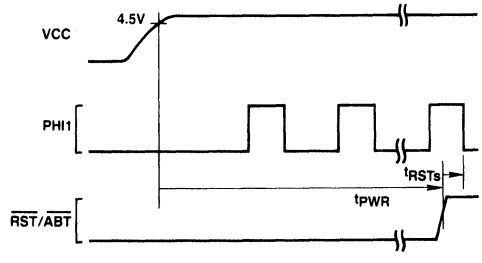
TL/EE/5054–65

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Not Applied



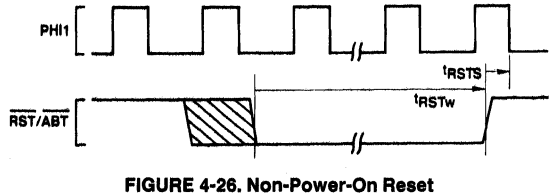
TL/EE/5054–66

FIGURE 4-24. Abort Timing, $\overline{FLT}$ Applied



TL/EE/5054–67

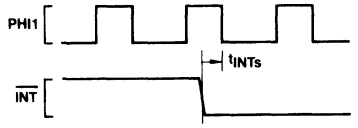
FIGURE 4-25. Power-On Reset



TL/EE/5054–68

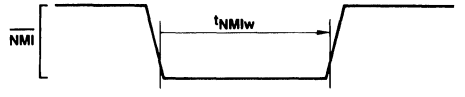
FIGURE 4-26. Non-Power-On Reset

4.0 Device Specifications (Continued)



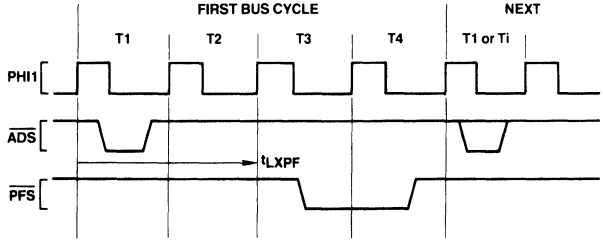
TL/EE/5054–69

FIGURE 4-27. $\overline{INT}$ Interrupt Signal Detection



TL/EE/5054–70

FIGURE 4-28. $\overline{NMI}$ Interrupt Signal Timing



TL/EE/5054–71

FIGURE 4-29. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction

NOTE:

In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

2

Appendix A: Instruction Formats

NOTATIONS:

i = Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f = Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c = Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op = Operation Code
Valid encodings shown with each format.

gen, gen 1, gen 2 = General Addressing Mode Field
See Sec. 2.2 for encodings.

reg = General Purpose Register Number

cond = Condition Code Field
- 0000 = EQual: Z = 1
- 0001 = Not Equal: Z = 0
- 0010 = Carry Set: C = 1
- 0011 = Carry Clear: C = 0
- 0100 = Higher: L = 1
- 0101 = Lower or Same: L = 0
- 0110 = Greater Than: N = 1
- 0111 = Less or Equal: N = 0
- 1000 = Flag Set: F = 1
- 1001 = Flag Clear: F = 0
- 1010 = LOwer: L = 0 and Z = 0
- 1011 = Higher or Same: L = 1 or Z = 1
- 1100 = Less Than: N = 0 and Z = 0
- 1101 = Greater or Equal: N = 1 or Z = 1
- 1110 = (Unconditionally True)
- 1111 = (Unconditionally False)

short = Short Immediate Value. May contain:
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.
  - 0000 = US
  - 0001 – 0111 = (Reserved)
  - 1000 = FP
  - 1001 = SP
  - 1010 = SB
  - 1011 = (Reserved)
  - 1100 = (Reserved)
  - 1101 = PSR
  - 1110 = INTBASE
  - 1111 = MOD

Options: in String Instructions

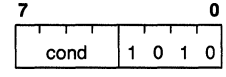
U/W	B	T

T = Translated
B = Backward
U/W = 00: None
01: While Match
11: Until Match

Configuration bits, in SETCFG:

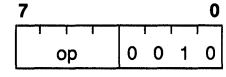
C	M	F	I

mreg: MMU Register number, in LMR, SMR.
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = PF0
- 0101 = PF1
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = SC
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA



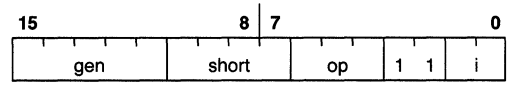
Format 0

Bcond (BR)



Format 1

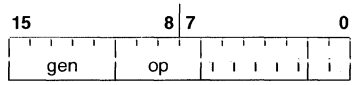
BSR	–0000	ENTER	–1000
RET	–0001	EXIT	–1001
CXP	–0010	NOP	–1010
RXP	–0011	WAIT	–1011
RETT	–0100	DIA	–1100
RETI	–0101	FLAG	–1101
SAVE	–0110	SVC	–1110
RESTORE	–0111	BPT	–1111



Format 2

ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		

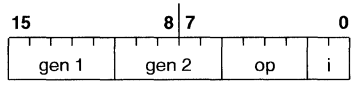
Appendix A: Instruction Formats (Continued)



Format 3

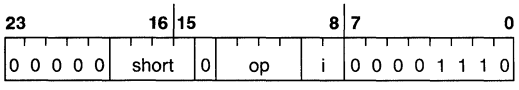
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

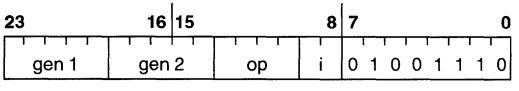
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

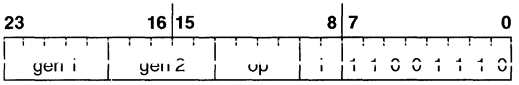
MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

Trap (UND) on 1XXX, 01XX



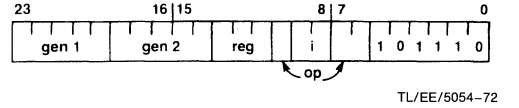
Format 6

ROT	−0000	NEG	−1000
ASH	−0001	NOT	−1001
CBIT	−0010	Trap (UND)	−1010
CBITI	−0011	SUBP	−1011
Trap (UND)	−0100	ABS	−1100
LSH	−0101	COM	−1101
SBIT	−0110	IBIT	−1110
SBITI	−0111	ADDP	−1111



Format 7

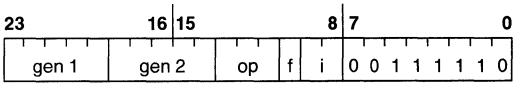
MOVM	−0000	MUL	−1000
CMPM	−0001	MEI	−1001
INSS	−0010	Trap (UND)	−1010
EXTS	−0011	DEI	−1011
MOVXBW	−0100	QUO	−1100
MOVZBW	−0101	REM	−1101
MOVZiD	−0110	MOD	−1110
MOVXiD	−0111	DIV	−1111



TL/EE/5054–72

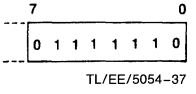
Format 8

EXT	−000	INDEX	−100
CVTP	−001	FFS	−101
INS	−010		
CHECK	−011		
MOVSU	−110, reg=001		
MOVUS	−110, reg=011		



Format 9

MOVif	−000	ROUND	−100
LFSR	−001	TRUNC	−101
MOVLF	−010	SFSR	−110
MOVFL	−011	FLOOR	−111

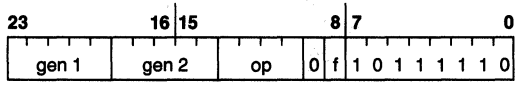


TL/EE/5054–37

Format 10

Trap (UND) Always

Appendix A: Instruction Formats (Continued)



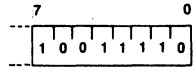
Format 11

ADDf	−0000	DIVf	−1000
MOVf	−0001	Trap (SLAVE)	−1001
CMPf	−0010	Trap (UND)	−1010
Trap (SLAVE)	−0011	Trap (UND)	−1011
SUBf	−0100	MULf	−1100
NEGf	−0101	ABSf	−1101
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111

TL/EE/5054–75

Format 12

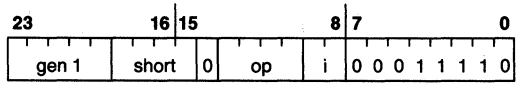
Trap (UND) Always



TL/EE/5054–76

Format 13

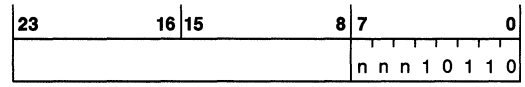
Trap (UND) Always



Format 14

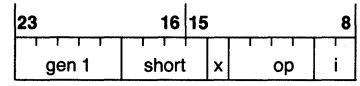
RDVAL	−0000	LMR	−0010
WRVAL	−0001	SMR	−0011

Trap (UND) on 01XX, 1XXX



Format 15 (Custom Slave)

nnn **Operation Word Format**

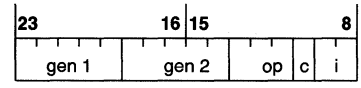


000

Format 15.0

CATST0	−0000	LCR	−0010
CATST1	−0001	SCR	−0011

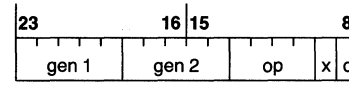
Trap (UND) on all others



001

Format 15.1

CCV3	−000	CCV2	−100
LCSR	−001	CCV1	−101
CCV5	−010	SCSR	−110
CCV4	−011	CCV0	−111



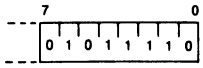
101

Format 15.5

CCAL0	−0000	CCAL3	−1000
CMOV0	−0001	CMOV3	−1001
CCMP0	−0010	Trap (UND)	−1010
CCMP1	−0011	Trap (UND)	−1011
CCAL1	−0100	CCAL2	−1100
CMOV2	−0101	CMOV1	−1101
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111

If nnn = 010, 011, 100, 110, 111
then Trap (UND) Always

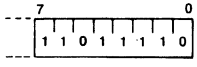
Appendix A: Instruction Formats (Continued)



TL/EE/5054–77

Format 16

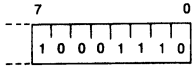
Trap (UND) Always



TL/EE/5054–78

Format 17

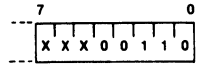
Trap (UND) Always



TL/EE/5054–79

Format 18

Trap (UND) Always

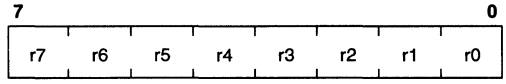


TL/EE/5054–80

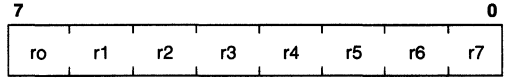
Format 19

Trap (UND) Always

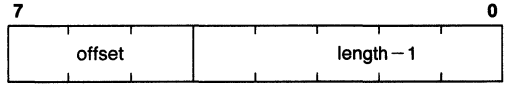
Implied Immediate Encodings:



Register Mask, appended to SAVE, ENTER



Register Mask, appended to RESTORE, EXIT



Offset/Length Modifier appended to INSS, EXTS

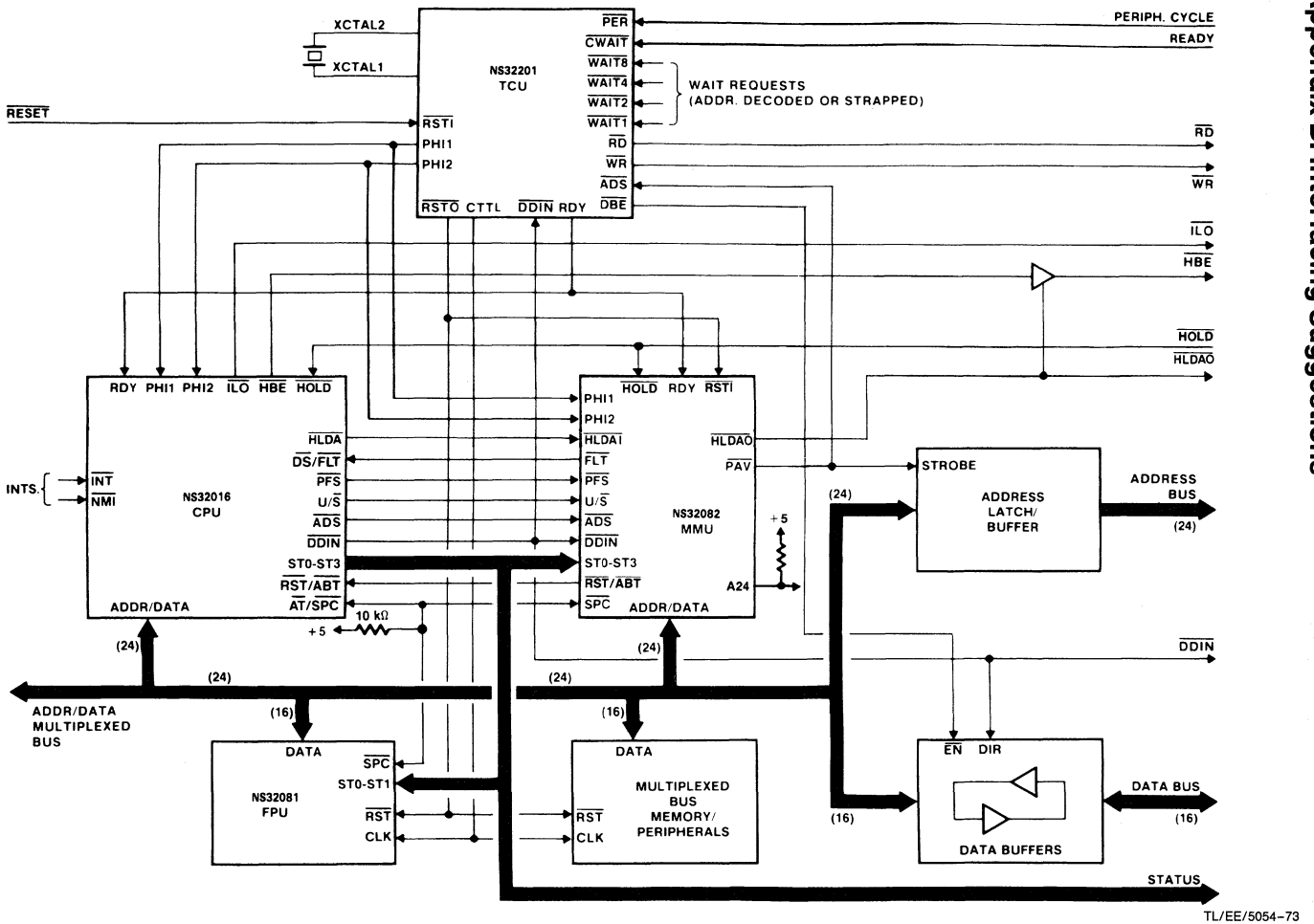


FIGURE B-1. System Connection Diagram

National Semiconductor Corporation

NS32008-6/NS32008-8/NS32008-10 High-Performance 8-Bit Microprocessors

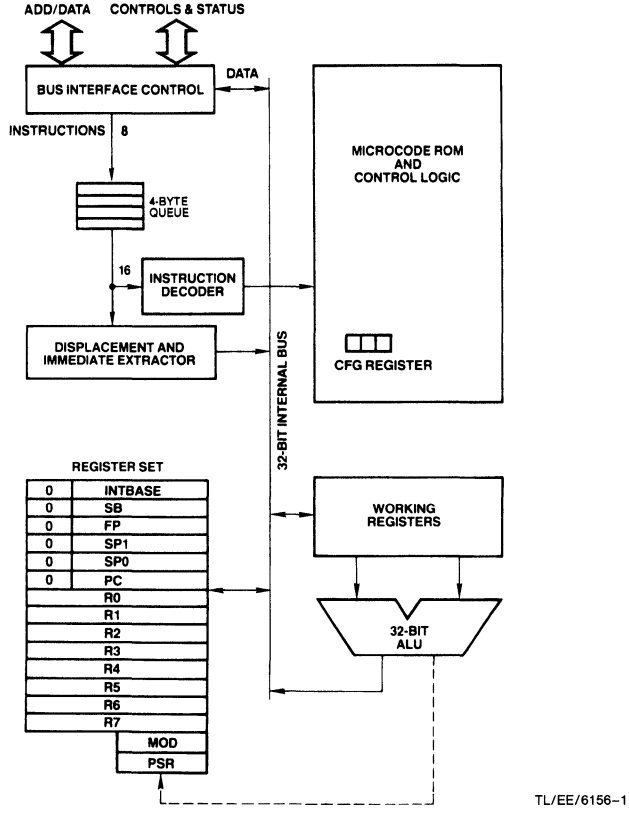
General Description

The NS32008 is a 32-bit microprocessor with a 16-MByte linear address space and a 8-bit external data bus. It has a 32-bit ALU, eight 32-bit general purpose registers, a four-byte prefetch queue, and a slave processor interface. The NS32008 is fabricated with National Semiconductor's advanced XMOS™ process, and is fully object code compatible with other Series 32000® processors. The Series 32000 instructions set is optimized for modular high-level languages (HLL). The set is very symmetric, it has a two address format, and it incorporates HLL oriented addressing modes. The capabilities of the NS32008 can be expanded with the use of the NS32081 floating point unit (FPU), which interfaces to the NS32008 as a slave processor. The NS32008 is a general purpose microprocessor that is ideal for a wide range of computational intensive applications.

Features

- 32-bit architecture and implementation
- 16-MByte linear address space
- 8-bit external data bus
- Powerful instruction set
  - — General 2-address capability
  - — High degree of symmetry
  - — Addressing modes optimized for high-level languages
- Series 32000 slave processor support
- High-speed XMOS technology
- 48-pin dual-in-line (DIP) package

Block Diagram



TL/EE/6156–1

Table of Contents

1.0 PRODUCT INTRODUCTION
1.1 NS32008 Design Goals

2.0 ARCHITECTURAL DESCRIPTION
2.1 Programming Model
2.1.1 General Purpose Registers
2.1.2 Dedicated Registers
2.1.3 The Configuration Register (CFG)
2.1.4 Memory Organization
2.1.5 Dedicated Tables
2.2 Instruction Set
2.2.1 General Instruction Format
2.2.2 Addressing Modes
2.2.3 Instruction Set Summary

3.0 FUNCTIONAL DESCRIPTION
3.1 Power and Grounding
3.2 Clocking
3.3 Resetting
3.4 Bus Cycles
3.4.1 Cycle Extension
3.4.2 Bus Status
3.4.3 Data Access Sequences
3.4.3.1 Bit Accesses
3.4.3.2 Bit Field Accesses
3.4.3.3 Extending Multiply Accesses
3.4.4 Instruction Fetches
3.4.5 Interrupt Control Cycles
3.4.6 Slave Processor Communication
3.4.6.1 Slave Processor Bus Cycles
3.4.6.2 Slave Operand Transfer Sequences
3.5 Bus Access Control
3.6 Instruction Status
3.7 NS32008 Interrupt Structure
3.7.1 General Interrupt/Trap Sequence
3.7.2 Interrupt/Trap Return

3.0 FUNCTIONAL DESCRIPTION (Continued)
3.7.3 Maskable Interrupts (The $\overline{\text{INT}}$ Plan)
3.7.3.1 Non-Vectored Mode
3.7.3.2 Vectored Mode: Non-Cascaded Case
3.7.3.3 Vectored Mode: Cascaded Case
3.7.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)
3.7.5 Traps
3.7.6 Prioritization
3.7.7 Interrupt/Trap Sequences: Detail Flow
3.7.7.1 Maskable/Non-Maskable Interrupt Sequence
3.7.7.2 Trap Sequences: Traps Other Than Trace
3.8 Slave Processor Instructions
3.8.1 Slave Processor Protocol
3.8.2 Floating Point Instructions
3.8.3 Custom Slave Instructions

4.0 DEVICE SPECIFICATIONS
4.1 Pin Descriptions
4.1.1 Supplies
4.1.2 Input Signals
4.1.3 Output Signals
4.1.4 Input/Output Signals
4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics
4.4.1 Definitions
4.4.2 Timing Tables
4.4.2.1 Output Signals: Internal Propagation Delays
4.4.2.2 Input Signals Requirements
4.4.2.3 Clocking Requirements
4.4.3 Timing Requirements

Appendix A: Instruction Formats

List of Illustrations

The General and Dedicated Registers 2-1
Processor Status Register 2-2
CFG Register 2-3
Data Formats for NS32008 Memory 2-4
Module Descriptor Format 2-5
A Sample Link Table 2-6
General Instruction Format 2-7
Index Byte Format 2-8
Displacement Encodings 2-9
Recommended Supply Connections 3-1
Clock Timing Relationships 3-2
Power-on Reset Requirements 3-3
General Reset Timing 3-4
Recommended Reset Connections 3-5

List of Illustrations (Continued)

Bus Connections 3-6
Read Cycle Timing 3-7
Write Cycle Timing 3-8
RDY Pin Timing 3-9
Extended Cycle Example 3-10
Slave Processor Connections 3-11
CPU Read from Slave Processor 3-12
CPU Write to Slave Processor 3-13
$\overline{\text{HOLD}}$ Timing, Bus Initially Idle 3-14
$\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle 3-15
Interrupt Dispatch and Cascade Tables 3-16
Interrupt/Trap Service Routine Calling Sequence 3-17
Return from Trap ($\overline{\text{RETT}}$ n) Instruction Flow 3-18
Return from Interrupt ($\overline{\text{RET}}$) Instruction Flow 3-19
Interrupt Control Connections (16 levels) 3-20
Cascaded Interrupt Control Unit Connections 3-21
Service Sequence 3-22
Slave Processor Protocol 3-23
Slave Processor Status Word Format 3-24
Connection Diagram 4-1
Timing Specification Standard (Signal Valid After Clock Edge) 4-2
Timing Specification Standard (Signal Valid Before Clock Edge) 4-3
Write Cycle 4-4
Read Cycle 4-5
Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially) 4-6
Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle) 4-7
Release from $\overline{\text{HOLD}}$ 4-8
Ready Sampling (CPU Initially READY) 4-9
Ready Sampling (CPU Initially NOT READY) 4-10
Slave Processor Write Timing 4-11
Slave Processor Read Timing 4-12
$\overline{\text{SPC}}$ Timing 4-13
Clock Waveforms 4-14
Relationship of $\overline{\text{PFS}}$ to Clock Cycles 4-14
Guaranteed Delay, $\overline{\text{PFS}}$ to Non-Sequential Fetch 4-15a
Guaranteed Delay, Non-Sequential Fetch to $\overline{\text{PFS}}$ 4-15b
Relationship of $\overline{\text{ILO}}$ to First Operand of an Interlocked Instruction 4-17
Relationship of $\overline{\text{ILO}}$ to Last Operand of an Interlocked Instruction 4-18
Relationship of $\overline{\text{ILO}}$ to Any Clock Cycle 4-19
$\overline{\text{U}}$/S Relationship to any Bus Cycle - Guaranteed Valid Interval 4-20
Power-On Reset 4-21
Non-Power-On Reset 4-22
$\overline{\text{INT}}$ Interrupt Signal Detection 4-23
$\overline{\text{NMI}}$ Interrupt Signal Timing 4-24
Relationship Between Last Data Transfer of an Instruction and $\overline{\text{PFS}}$ Pulse of Next Instruction 4-25

List of Tables

NS32008 Addressing Modes 2-1
NS32008 Instruction Set Summary 2-2
Interrupt Sequences 3-1
Floating-Point Instruction Protocols 3-2
Custom Slave Instruction Protocols 3-3

1.0 Product Introduction

The Series 32000 Microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from a minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operations. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes. Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types. The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set. While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations. The Series 32000 CPUs represent two-operand machines with each operand addressable by all addressing modes. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management. Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing. The NS32008 has 24-bit address pointers that can address up to 16 megabytes without requiring any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support. Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept. The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

1.1 NS32008 DESIGN GOALS

The NS32008 is aimed at small to medium size systems, and is designed to bridge the gap between 8-bit CPUs and the higher-end members of the Series 32000 family. The NS32008 provides an 8-bit data bus and is the only CPU in the Series 32000 family that does not support virtual memory.

The NS32008 is most suitable for systems designed with 8-bit memory and peripherals.

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32008 CPU.

2.1.1 General Purpose Registers

There are eight registers for meeting high-speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are 32 bits in length. If a general register is specified for an operand that is 8 or 16 bits long, only the low part of the register is used; the high part is not referenced or modified.

2.1.2 Dedicated Registers

The eight dedicated registers of the NS32008 are assigned specific functions:

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC

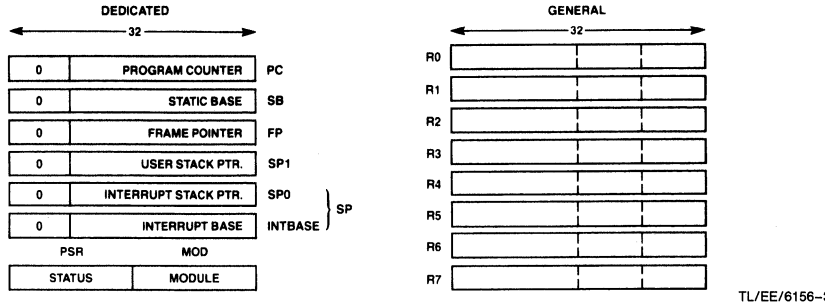


FIGURE 2-1. The General and Dedicated Registers

2.0 Architectural Description (Continued)

is used to reference memory in the program section. (In the NS32008, the upper eight bits of this register are always zero.)

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0, then SP refers to SP0. If the S bit in the PSR is 1, the SP refers to SP1. (In the NS32008, the upper eight bits of these registers are always zero.)

Stacks in the Series 32000 family grow downward in memory. A push operation pre-decrements the stack pointer by the operand length. A pop operation post-increments the stack pointer by the operand length.

FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32008, the upper eight bits of this register are always zero.)

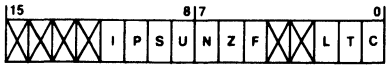
SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32008, the upper eight bits of this register are always zero.)

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.7). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32008, the upper eight bits of this register are always zero.)

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is 16 bits long, therefore the module table must be contained within the first 64K bytes of memory.

PSR: The PROCESSOR STATUS REGISTER holds the status codes for the NS32008 microprocessor.

The PSR is 16 bits long, divided into two 8-bit halves (*Figure 2-2*). The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/6156–4

FIGURE 2-2. The Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Section 3.7.5).

L: The L bit is altered by comparison instructions. In a comparison instruction, the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating-Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction, the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction, the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1", no privileged instructions may be executed. if the U bit is "0", then all instructions may be executed. When U=0, the NS32008 is said to be in Supervisor Mode; when U=1, the NS32008 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register on SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps it. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Section 3.7.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I = 1, then all interrupts will be accepted (Section 3.7). If I = 0, only the $\overline{\text{NMI}}$ interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32008 CPU is the 4-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3*.

TL/EE/6156–5

FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the INT pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Section 3.7.

The F and C bits declare the presence of the FPU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

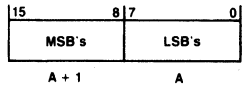
2.1.4 Memory Organization

The main memory of the NS32008 is a uniform linear address space. Memory locations are numbered sequentially

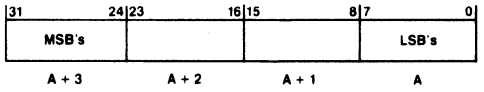
2.0 Architectural Description (Continued)

starting at zero and ending at $2^{24}-1$. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits (*Figure 2-4A*). Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.

A. Byte at Address A



B. Word at Address A



TL/EE/6156–6

C. Double Word at Address A

FIGURE 2-4. Data Formats for NS32008 Memory

Two contiguous bytes are called a word (*Figure 2-4B*). Except where noted (Section 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.

Two contiguous words are called a double word (*Figure 2-4C*). Except where noted (Section 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.

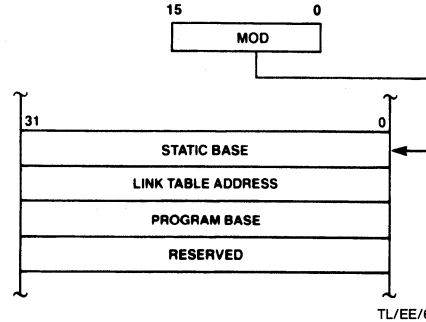
2.1.5 Dedicated Tables

Two of the NS32008 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Section 3.7.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by the NS32008. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically updated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in *Figure 2-5*. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.



TL/EE/6156–7

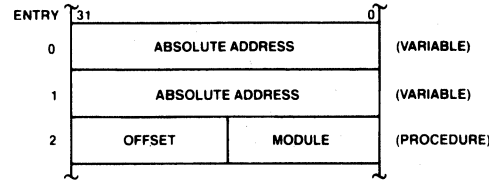
FIGURE 2-5. Module Descriptor Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1. Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2. Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-6*. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

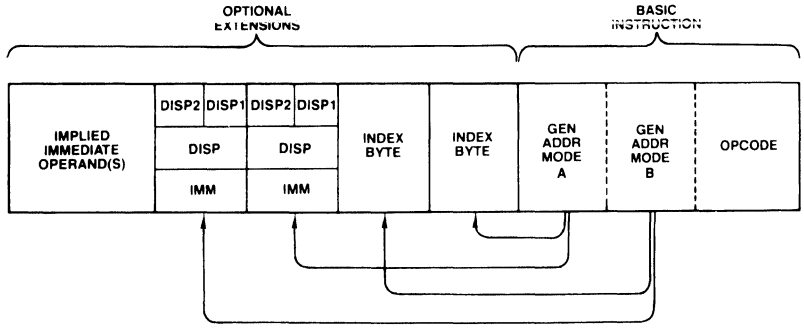
For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/6156–8

FIGURE 2-6. A Sample Link Table

2.0 Architectural Description (Continued)



TL/EE/6156–10

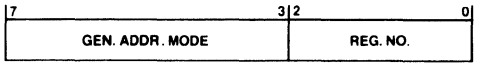
FIGURE 2-7. General Instruction Format

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-7 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions which may appear, depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-8*.



TL/EE/6156–9

FIGURE 2-8. Index Byte Format

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Displacement/Immediate field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in *Figure 2-9*, with the remaining bits interpreted as a signed (two's complement) value. The size of an Immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most-significant byte first. Note that this is different from the memory representation of data (Section 2.1.4.).

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Section 2.2.3).

2.2.2 Addressing Modes

The NS32008 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the NS32008 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32008 Addressing Modes fall into nine basic types:

Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the effective address of the operand in memory.

Memory Space: Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

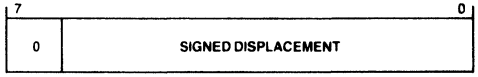
Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

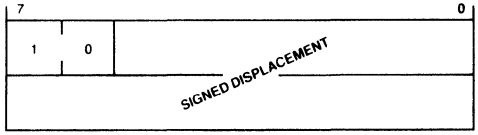
External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

2.0 Architectural Description (Continued)

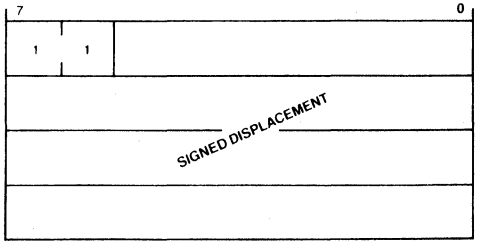
Byte Displacement: Range −64 to +63



Word Displacement: Range −8192 to +8191



Double Word Displacement: Range (Entire Addressing Space)



TL/EE/6156–13

FIGURE 2-9. Displacement Encodings

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32008 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating-Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction. (See Appendix A for encodings.)

imm = Immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16, 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0–R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, UPSR, (bottom eight PSR bits).

creg = A Custom Slave Processor Register (implementation dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction. (See Appendix A for encodings.)

2.0 Architectural Description (Continued)

TABLE 2-1
NS32008 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2 × Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn. 'Mode' and 'n' are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

TABLE 2-2
NS32008 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move effective address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDi	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply.
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to extended integer.
7	DEIi	gen,gen	Divide extended integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add packed.
6	SUBPi	gen,gen	Subtract packed.

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

2.0 Architectural Description (Continued)

TABLE 2-2
Instruction Set Summary (Continued)

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical shift, left or right.
6	ASHi	gen,gen	Arithmetic shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked.
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit.

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to bit field pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bound check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 – Comparison Value
R3 – Translation Table Pointer
R2 – String 2 Pointer
R1 – String 1 Pointer
R0 – Limit Count

Options on all string instructions are:

B (Backward): Decrement string pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Description
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare, translating String 1 bytes.
5	SKPSi	options	Skip over String 1 entries.
	SKPST	options	Skip, translating bytes for Until/While.

2.0 Architectural Description (Continued)

TABLE 2-2
Instruction Set Summary (Continued)

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor call.
1	FLAG		Flag trap.
1	BPT		Breakpoint trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame. (Enter Procedure)
1	EXIT	[reg list]	Restore registers and reclaim stack frame. (Exit Procedure)
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save general purpose registers.
1	RESTORE	[reg list]	Restore general purpose registers.
2	LPRi	areg,gen	Load dedicated register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store dedicated register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust stack pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set configuration register. (Privileged)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a floating point value.
9	MOVLF	gen,gen	Move and shorten a long value to standard.
9	MOVFL	gen,gen	Move and lengthen a standard value to long.
9	MOVif	gen,gen	Convert any integer to standard or long floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

2.0 Architectural Description (Continued)

TABLE 2-2
Instruction Set Summary (Continued)

CUSTOM SLAVE Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
	CCMP1c	gen,gen	
15.5	CCMP0c	gen,gen	Custom compare.
	CMOV3c	gen,gen	
15.1	CCV0ci	gen,gen	Custom convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load custom status register.
15.1	SCSR	gen	Store custom status register.
15.0	CATST0	gen	Custom address/test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load custom register. (Privileged)
15.0	SCR	creg,gen	Store custom register. (Privileged)

3.0 Functional Description

3.1 POWER AND GROUNDING

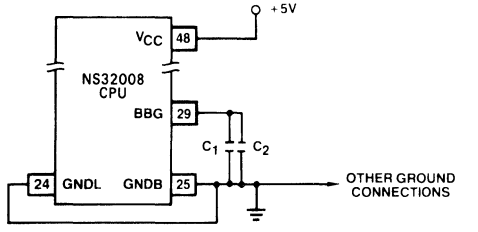
The NS32008 requires a single 5V power supply, applied on pin 48 (V_{CC}).

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (*Figure 3-1*).

In addition to V_{CC} and Ground, the NS32008 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Figure 3-1*) from the BBG pin to ground. Recommended values for these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.



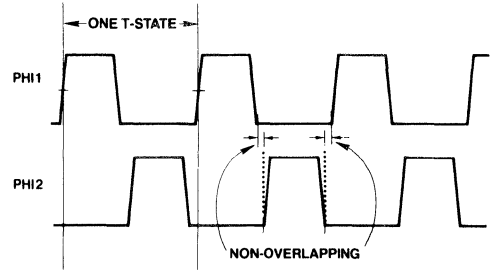
TL/EE/6156-14

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32008 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in *Figure 3-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU and/or one step of an external bus transfer. See Section 4 for complete specifications of PHI1 and PHI2.



TL/EE/6156-15

FIGURE 3-2. Clock Timing Relationships

As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be

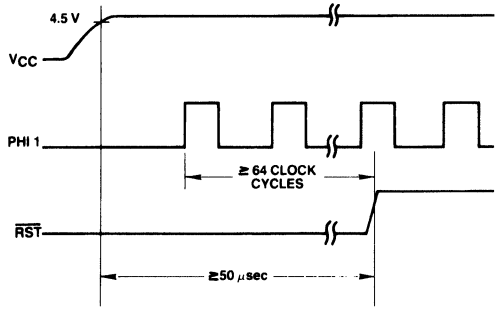
3.0 Functional Description (Continued)

connected anywhere except from the TCU to the CPU. A TTL clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 RESETTING

The $\overline{RST}$ pin serves as a reset for on-chip logic. The CPU may be reset at any time by pulling the $\overline{RST}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

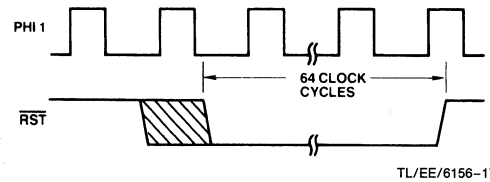
On application of power, $\overline{RST}$ must be held low for at least 50 μs after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4*.



TL/EE/6156–16

FIGURE 3-3. Power-On Reset Requirements

The NS32201 Timing Contol Unit (TCU) provides circuitry to meet the Reset requirements of the NS32008 CPU. *Figure 3-5* shows the recommended connections.



TL/EE/6156–17

FIGURE 3-4. General Reset Timing

3.4 BUS CYCLES

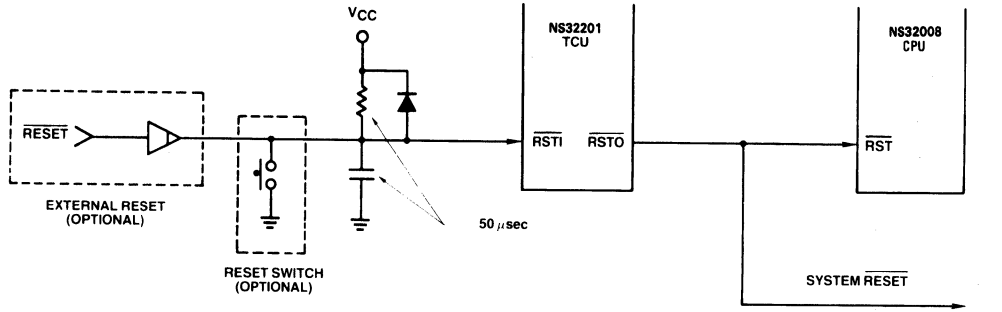
The NS32008 will perform a bus cycle for one of the following reasons:

1. To write or read data to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
2. To fetch instructions into the 4-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3. To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4. To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Section 4. The only external difference between them is the 4-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied and transfers are performed 16 bits at a time (Section 3.4.6).

Figure 3-6 shows typical bus connections for the NS32008. The address, data, and control signals referenced in the following discussion are shown in this figure.

The sequence of events in a non-Slave Processor bus cycle is shown in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Section 3.4.1).



TL/EE/6156–18

FIGURE 3-5. Recommended Reset Connections

3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

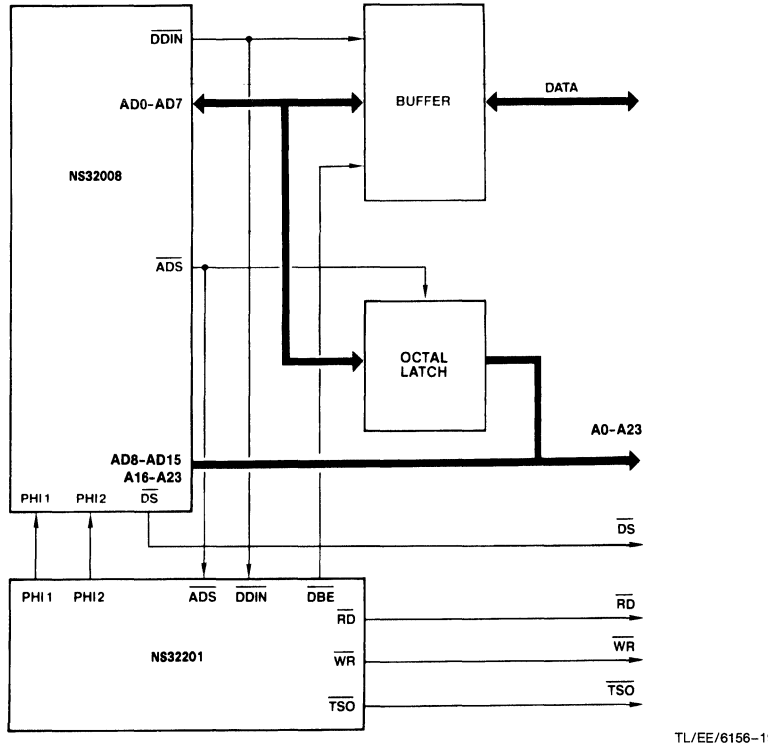
During T1, the CPU applies an address on pins AD0–AD15 and A16–A23. It also provides a low-going pulse on the $\overline{ADS}$ pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing address bits 0–7 from the AD0–AD7 pins. See *Figure 3-6*. Also during this time the status signal $\overline{DDIN}$, indicating the direction of the transfer, becomes valid.

During T2, the CPU switches the Data Bus, AD0–AD7, to either accept or present data. Note that the signals AD8–AD15 and AD16–AD23 remain valid, and need not be latched. It also starts the Data Strobe ($\overline{DS}$), signaling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time: $\overline{RD}$ (Read Strobe) or $\overline{WR}$ (Write Strobe), $\overline{TSO}$ (Timing State Output, indicating that T2 has been reached) and $\overline{DBE}$ (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the end of T2 or T3, on the falling edge of the PHI2 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Section 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD7) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Section 4. Data must, however, be held at least until the beginning of T4. $\overline{DS}$ and $\overline{RD}$ are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

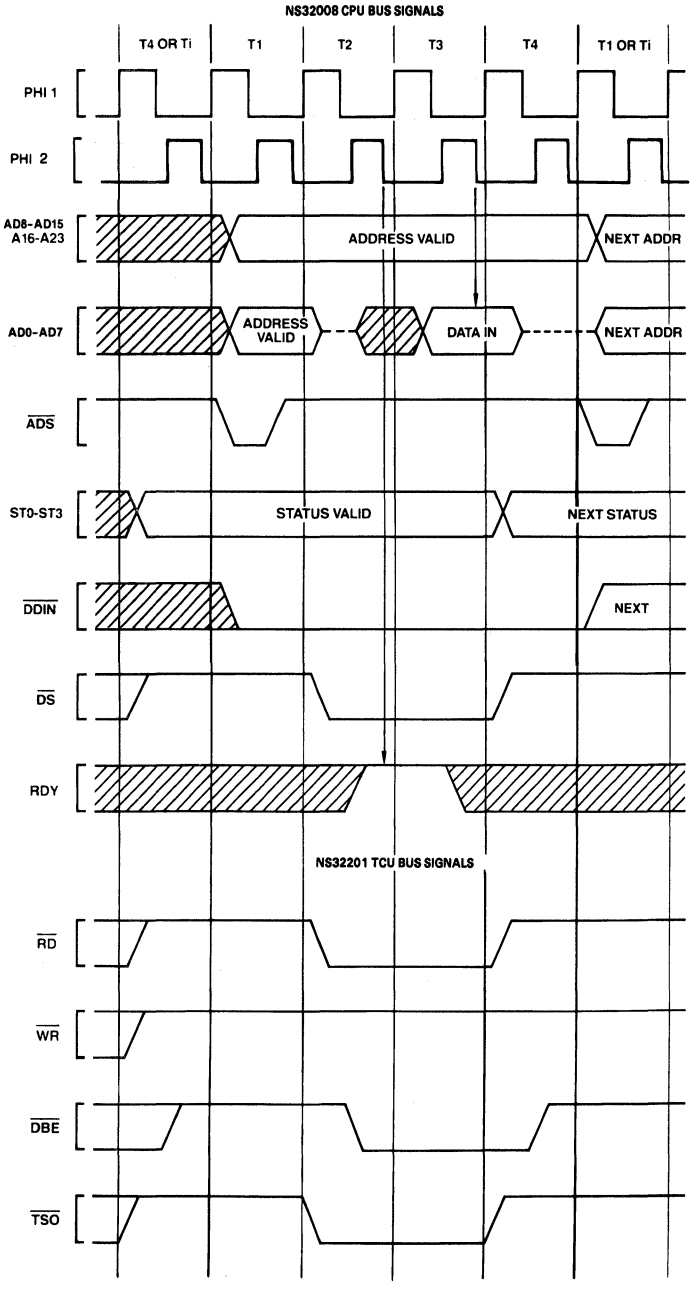
The T4 state finishes the bus cycle. At the beginning of T4, the $\overline{DS}$, $\overline{RD}$ or $\overline{WR}$, and $\overline{TSO}$ signals go inactive, and at the rising edge of PHI2, $\overline{DBE}$ goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).



TL/EE/6156–19

FIGURE 3-6. Bus Connections

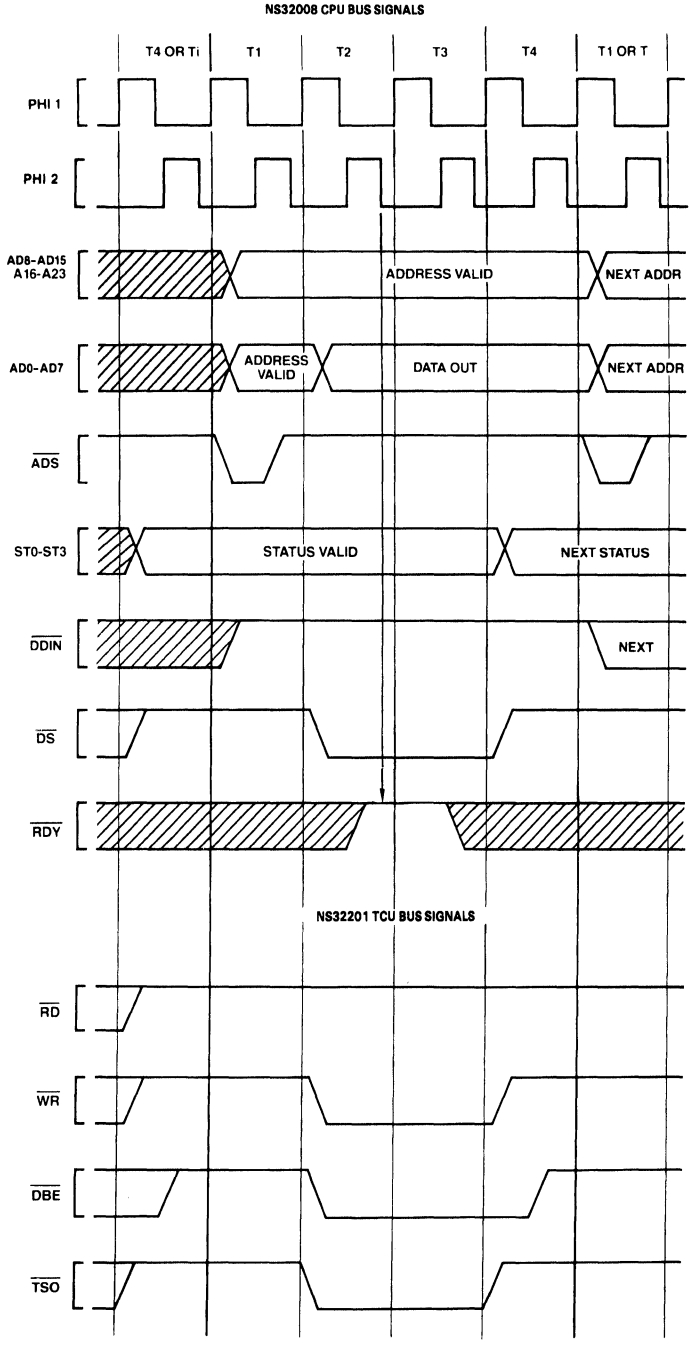
3.0 Functional Description (Continued)



TL/EE/6156–20

FIGURE 3-7. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/6156–21

FIGURE 3-8. Write Cycle Timing

3.0 Functional Description (Continued)

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32008 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

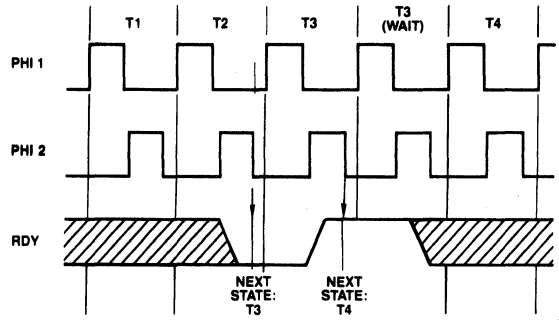
At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "WAIT STATE". See *Figure 3-9*.

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1. $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT States until removed.
2. $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (collectively, $\overline{\text{WAITn}}$), which may be given a 4-bit binary value requesting a specific number of WAIT States from 0 to 15.
3. $\overline{\text{PER}}$ (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU $\overline{\text{WAITn}}$ pins.



TL/EE/6156-22

FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS32008 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, why it is idle.

Referring to *Figures 3-7* and *3-8*, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the bus status and, if desired, latch the decoded signals before $\overline{\text{ADS}}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a 4-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 The bus is idle because the CPU does not need to perform a bus access.

0001 The bus is idle because the CPU is executing the WAIT instruction.

0010 (Reserved for future use.)

0011 The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$), it will read from address FFFF00_{16}, but will ignore any data provided. To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$), it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Section 3.4.5.

0101 Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Section 3.4.5.

0110 End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Section 3.4.5.

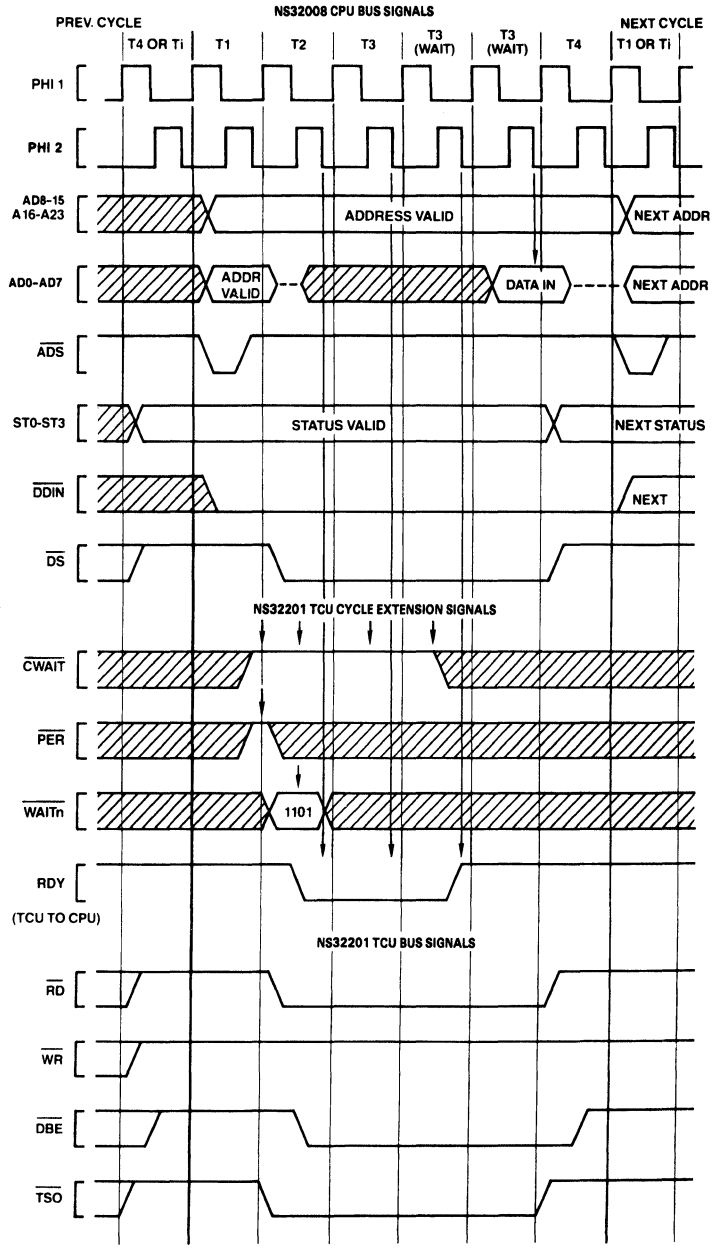
0111 End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Section 3.4.5.

1000 Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

3.0 Functional Description (Continued)



TL/EE/6156-23

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAITn}}$ indicate points at which the TCU samples.
Arrows on AD0–AD7 and RDY indicate points at which the CPU samples.

FIGURE 3-10. Extended Cycle Example

3.0 Functional Description (Continued)

1001 Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 Data Transfer.

The CPU is reading or writing an operand of an instruction.

1011 Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten.

1100 Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Section 3.8.1.

1110 Read Slave Processor Status.

The CPU is reading a status word from a Slave Processor. This occurs after the Slave Processor has signaled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions, it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Section 3.8.1.

1111 Broadcast Slave ID.

The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point, the CPU is communicating with only one Slave Processor. See Section 3.8.1.

3.4.3 Data Access Sequences

The NS32008 accesses all memory and peripheral devices in sequences of single-byte transfers. Transfer of values larger than bytes is performed from least-significant byte (lowest address) to most-significant byte.

3.4.3.1 Bit Accesses

The bit instructions access the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double Word transfer starting at the address containing the least-significant bit of the field. The Double Word is read by an Exact instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.3.3 Extending Multiply Accesses

The extending multiply instruction (MEI) will return a result which is twice the size in bytes of the operand that it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half.

3.4.4 Instruction Fetches

Instructions for the NS32008 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the 4-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Section 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full.

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the Instruction Queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status.

3.4.5 Interrupt Control Cycles

Activating the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0–ST3. All Interrupt Control cycles are Read cycles. Table 3-1 summarizes NS32008 interrupt sequences.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32008 interrupt structure, see Section 3.7.

3.0 Functional Description (Continued)

TABLE 3-1
Interrupt Sequences

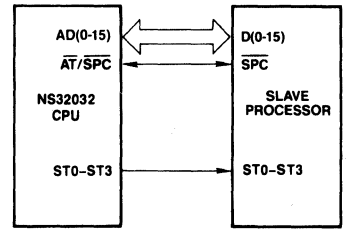
Cycle	Status	Address	$\overline{DDIN}$	Bus
A. Nonmaskable Interrupt Control Sequences.				
Interrupt Acknowledge				
1	0100	$FFFF00_{16}$	0	Don't Care
Interrupt Return				
None: Performed through Return from Trap (RETT) instruction.				
B. Nonvectored Interrupt Control Sequences.				
Interrupt Acknowledge				
1	0100	$FFFE00_{16}$	0	Don't Care
Interrupt Return				
None. Performed through return from Trap (RETT) instruction.				
C. Vectored Interrupt Sequences: Noncascaded.				
Interrupt Acknowledge				
1	0100	$FFFE00_{16}$	0	Vector: Range 0–127
Interrupt Return				
1	0110	$FFFE00_{16}$	0	Vector: Same as in Previous Interrupt Acknowledge Cycle
D. Vectored Interrupt Sequences: Cascaded.				
Interrupt Acknowledge				
1	0100	$FFFE00_{16}$	0	Cascade Index: Range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)				
2	0101	Cascade Address	0	Vector: Range 0–255
Interrupt Return				
1	0110	$FFFE00_{16}$	0	Cascade Index: Same as in Previous Interrupt Acknowledge Cycle
(The CPU here uses the Cascade Index to find the Cascade Address.)				
2	0111	Cascade Address	0	Don't Care

2

3.0 Functional Description (Continued)

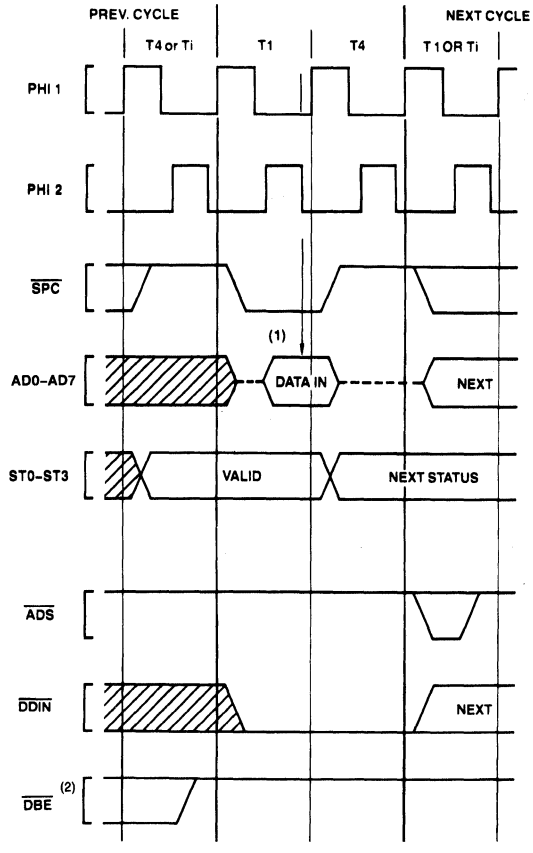
3.4.6 Slave Processor Communication

The $\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In a Slave Processor bus cycle, data is transferred 16 bits at a time on the Data Bus (AD0–AD15) and the status lines ST0–ST3 are monitored by each Slave Processor in order to determine the type of transfer being performed. *Figure 3-11* shows typical Slave Processor connections. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Section 3.8 for full protocol sequences.



TL/EE/6156–24

FIGURE 3-11. Slave Processor Connections



TL/EE/6156–25

Note 1. CPU samples Data Bus here.

Note 2. $\overline{DBE}$ and all other NS32201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-12. CPU Read from Slave Processor

3.0 Functional Description (Continued)

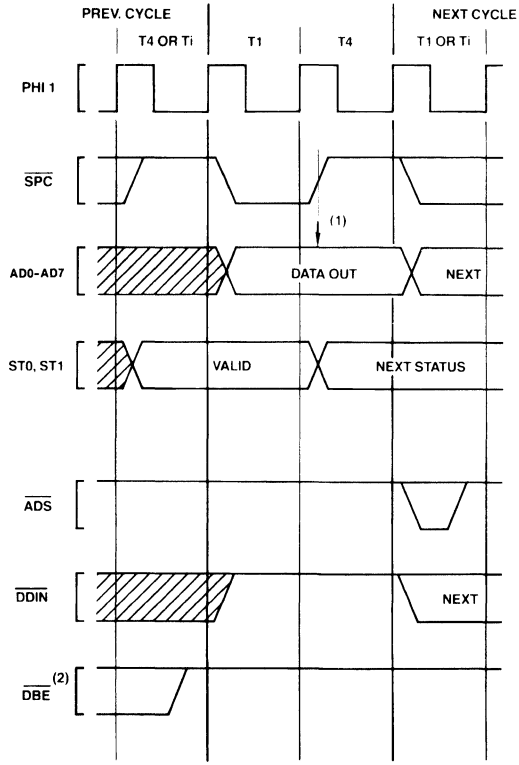
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-12* and *3-13*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ($\overline{ADS}$), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.6.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0–AD7), and a Word operand is transferred on the entire 16-bit bus (AD0–AD15). A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant to most-significant word.



TL/EE/6156–26

Note 1. Slave Processor samples Data Bus here.

Note 2. $\overline{DBE}$, being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on ADS, TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

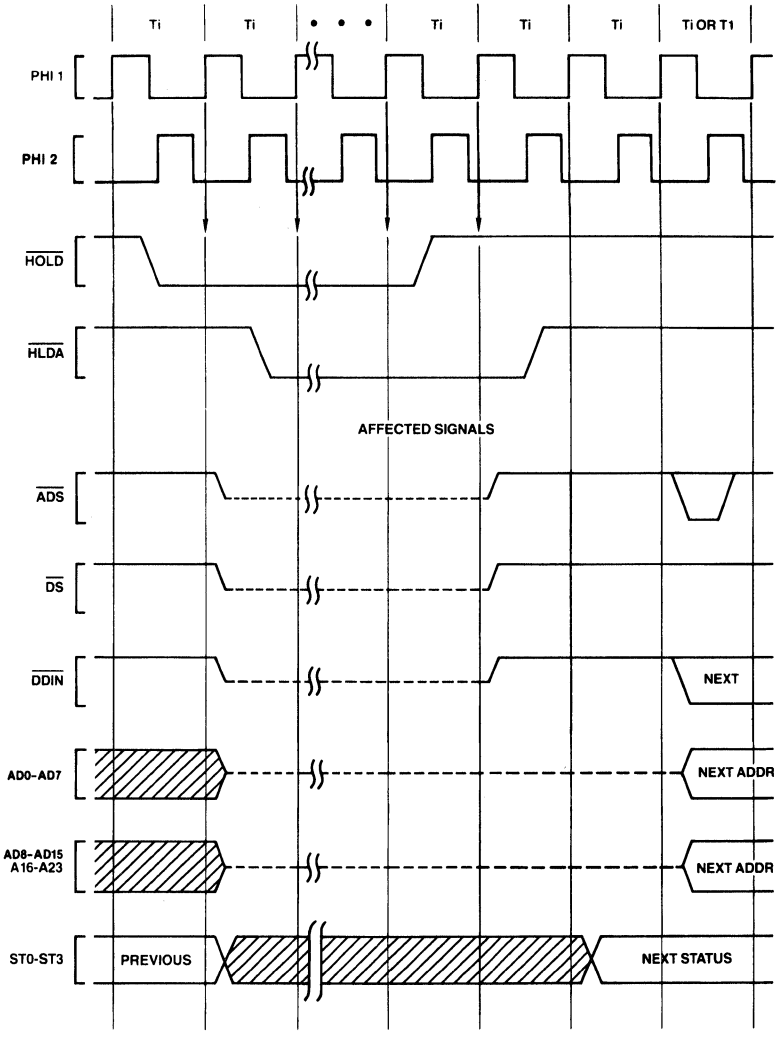
FIGURE 3-13. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 BUS ACCESS CONTROL

The NS32008 CPU has the capability of relinquishing its access to the bus request from a DMA device or another CPU. This capability is implemented on the $\overline{\text{HOLD}}$ (Hold Request) and $\overline{\text{HLDA}}$ (Hold Acknowledge) pins. By asserting $\overline{\text{HOLD}}$ low, an external device requests access to the bus. On receipt of $\overline{\text{HLDA}}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD15, A16–A23, $\overline{\text{ADS}}$ and $\overline{\text{DDIN}}$ pins to the TRI-STATE® condition. To return control of the bus to the CPU, the device sets $\overline{\text{HOLD}}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{\text{HLDA}}$ inactive.

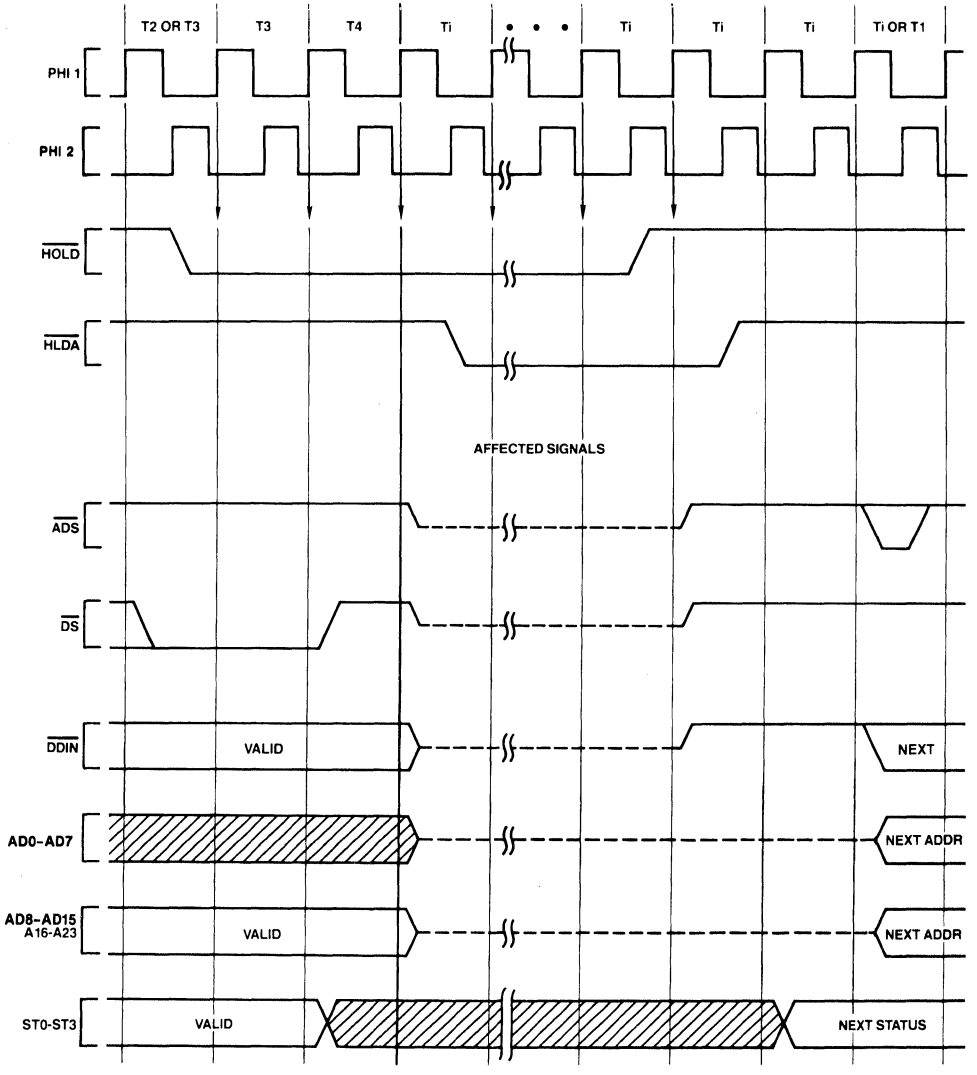
How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{\text{HOLD}}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-14* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-15* shows the sequence if the CPU is using the bus at the time that the $\overline{\text{HOLD}}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case, it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus, but has initiated a bus cycle internally.



TL/EE/6156–27

FIGURE 3-14. $\overline{\text{HOLD}}$ Timing, Bus Initially Idle

3.0 Functional Description (Continued)



TL/EE/6156-28

FIGURE 3-15. HOLD Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

3.6 INSTRUCTION STATUS

In addition to the four bits of bus cycle status (ST0–ST3), the NS32008 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus inerface section.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes.

U/$\overline{S}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-19.*

$\overline{ILO}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multiprocessor communication and resource sharing. As with the U/$\overline{S}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification section, *Figures 4-16* and *4-17.*

3.7 NS32008 INTERRUPT STRUCTURE

The NS32008 CPU has two interrupt pins: INT, on which maskable interrupts may be requested, and NMI, on which nonmaskable interrupts may be requested.

In addition, there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.7.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

1. Adjustment of Registers.

   Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2. Saving Processor Status.

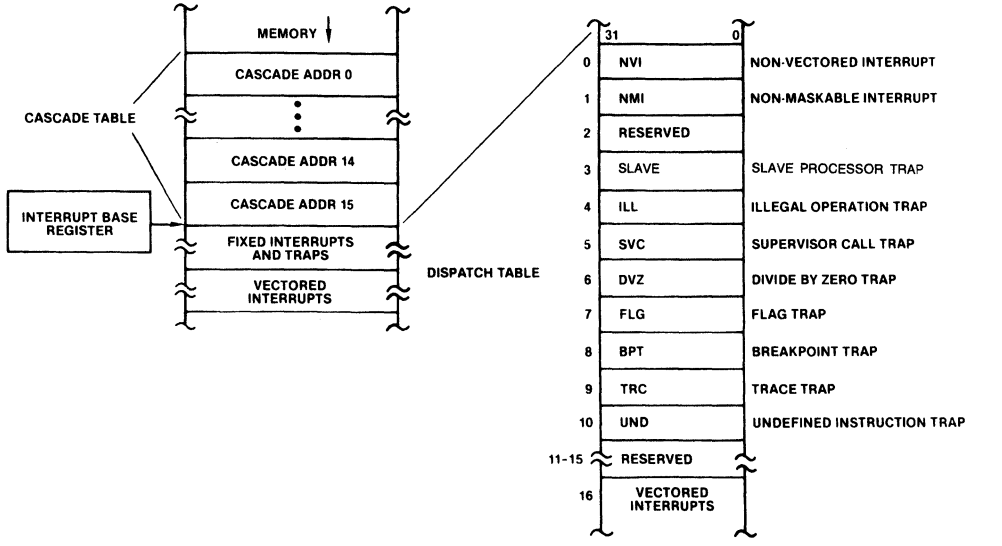
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

3. Vector Acquisition.

   A Vector is either obtained from the Data Bus or is supplied by default.

4. Service Call.

   The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-16.* A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.



TL/EE/6156-29

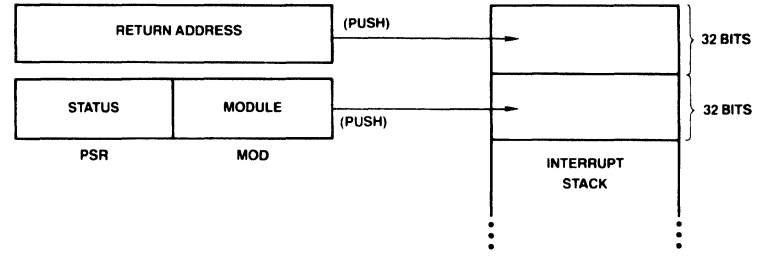
FIGURE 3-16. Interrupt Dispatch and Cascade Tables

3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-17*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Sec. 3.7.7.1
Traps (except Trace):	Sec. 3.7.7.2
Trace Trap:	Sec. 3.7.7.3



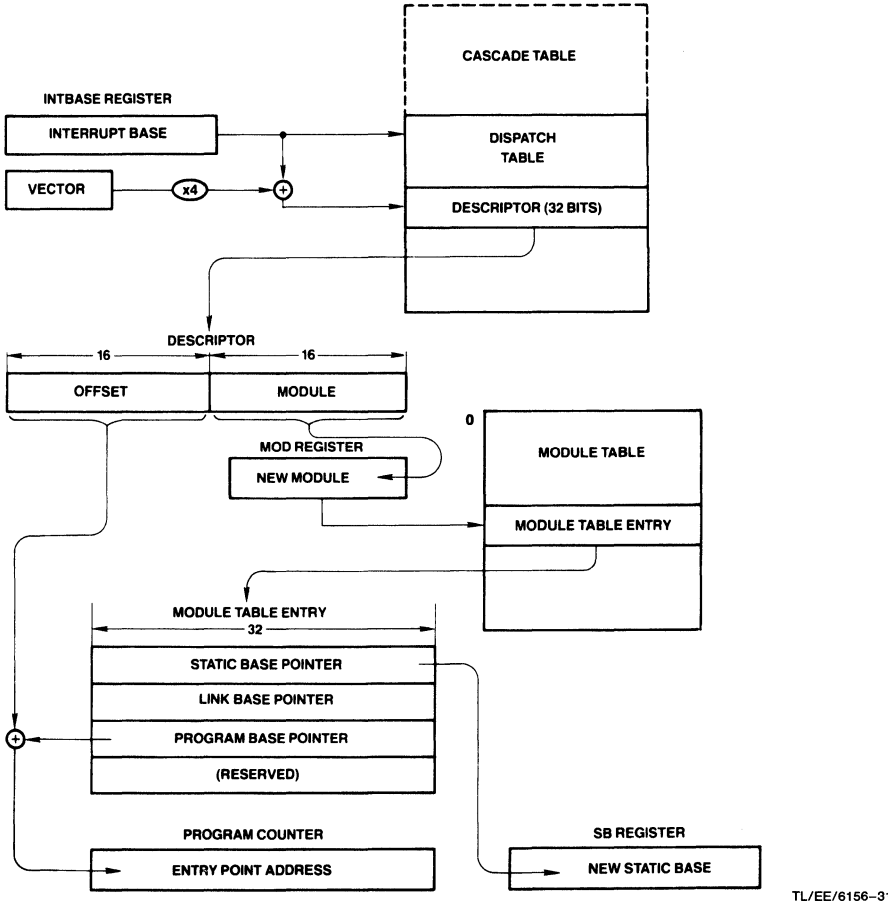


FIGURE 3-17. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.7.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instruction is used. The RETT (Return From Trap) instruction (*Figure 3-18*) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has been completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-19*.

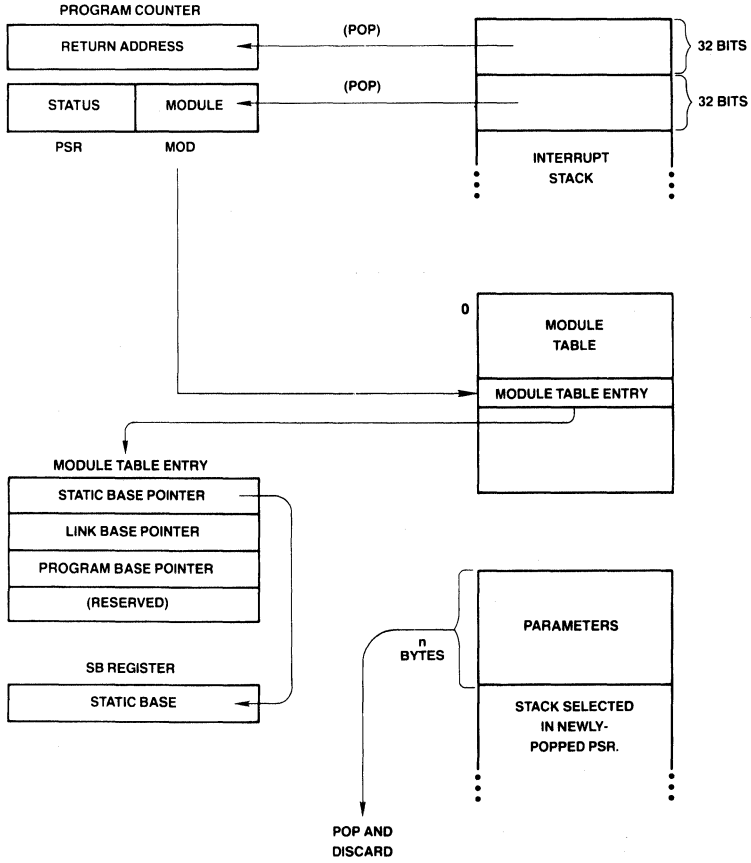
3.7.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG register bit I = 0) or Vectored (bit I = 1).

3.7.3.1 Non-Vectored Mode

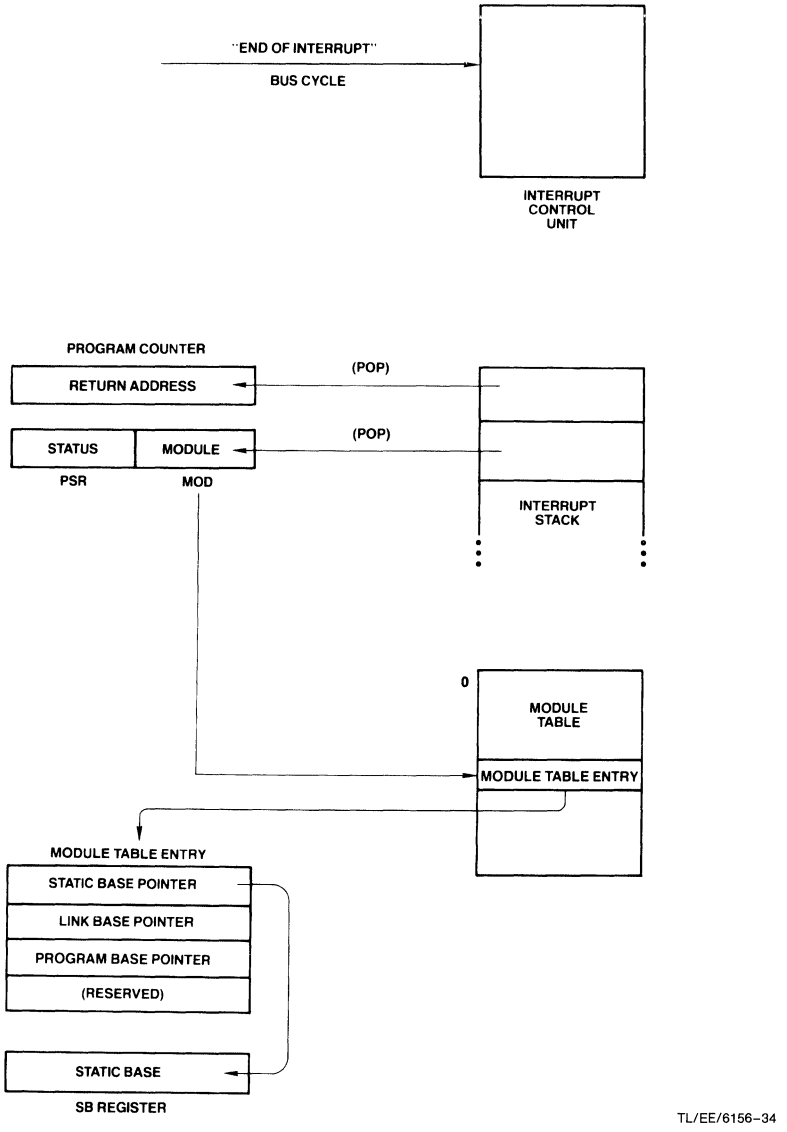
In the Non-Vectored Mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary. The RETT instruction should be used to return from an interrupt in Non-Vectored Mode.



TL/EE/6156–32

FIGURE 3-18. Return from Trap (RETTn) Instruction Flow

3.0 Functional Description (Continued)



TL/EE/6156–34

FIGURE 3-19. Return from Interrupt (RETI) Instruction Flow

3.0 Functional Description (Continued)

3.7.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. *Figure 3-20* shows the connections required for a single ICU. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) reading a vector value from the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may reprioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.7.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-21* shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{INT}$ pin.

In a system which uses cascading, two tasks must be performed upon initialization:

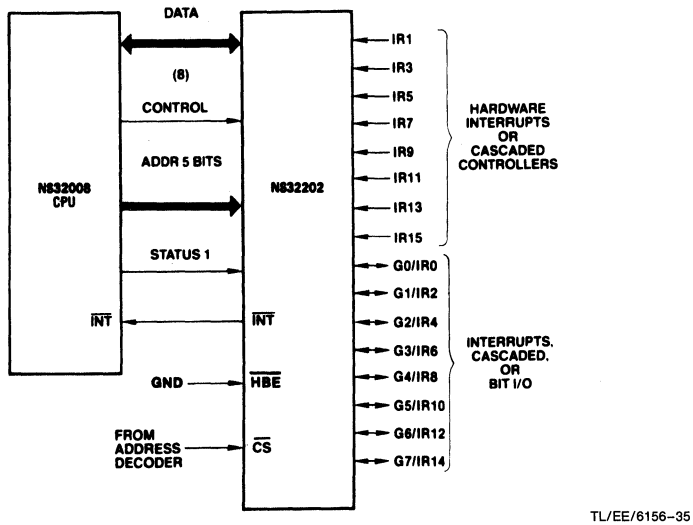
1. For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.
2. A Cascade Table must be established in memory. The Cascade Table is located in a *negative* direction from the location indicated by the CPU Interrupt Base (INTBASE) register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-16 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range −16 to −1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Section 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Section 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

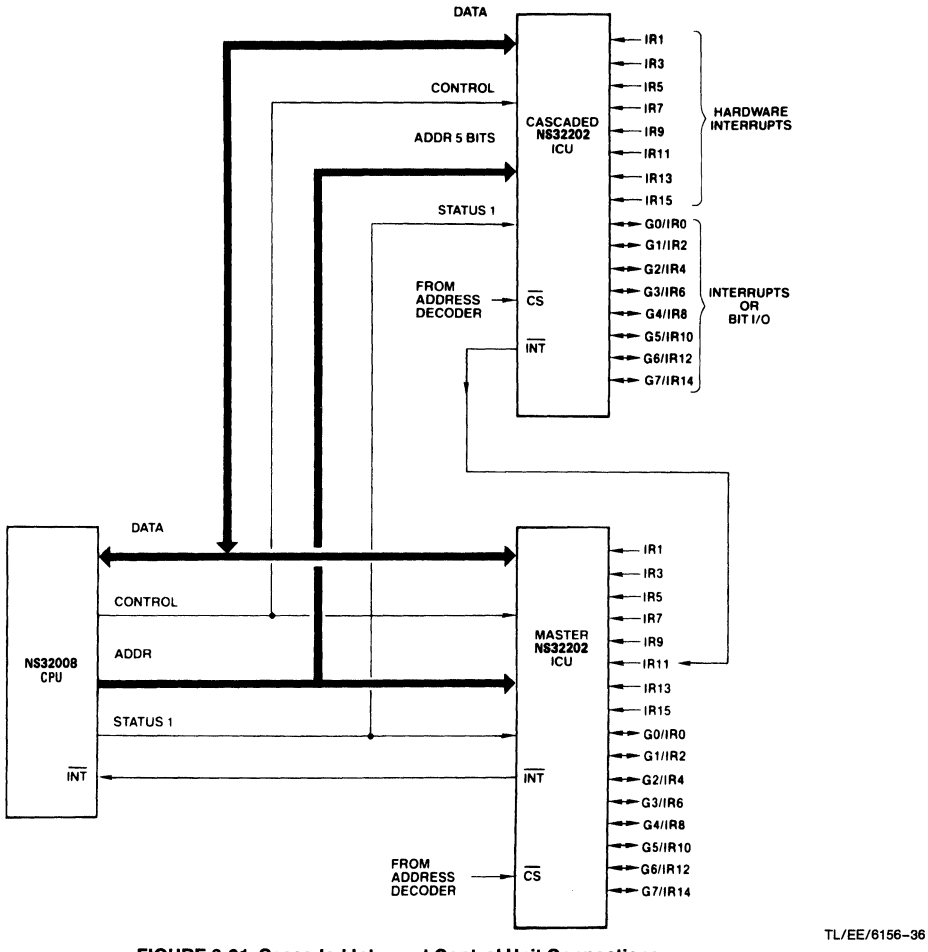
Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the interrupt mask register of the interrupt controller. However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.



TL/EE/6156–35

FIGURE 3-20. Interrupt Control Unit Connections (16 Levels)

3.0 Functional Description (Continued)



TL/EE/6156–36

FIGURE 3-21. Cascaded Interrupt Control Unit Connections

3.7.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable interrupt is triggered whenever a falling edge is detected on the $\overline{\text{NMI}}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is $FFFF00_{16}$. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Section 3.7.7.1.

3.7.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except TRC is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32008 are:

Trap (Slave): An exceptional condition was detected by the Floating-Point unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Section 3.8.1).

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU trap is used for Floating-Point division by zero.)

3.0 Functional Description (Continued)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

3.7.6 Prioritization

The NS32008 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1. Traps other than Trace (Highest priority)
2. Non-Maskable Interrupt
3. Maskable Interrupts
4. Trace Trap (Lowest priority)

3.7.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-22*. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{INT}$ or $\overline{NMI}$ pins, respectively), see Section 3.7.7.1. For the Trace Trap, see Section 3.7.7.3, and for all other traps, see Section 3.7.7.2.

3.7.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{NMI}$ pin receives a falling edge, or the $\overline{INT}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptable point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master, Section 3.4.2).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE $+4*$ Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded, Section 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-22*.

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector*4 + INTBASE Register contents.
2) Move the Module field of the Descriptor into the MOD Register.
3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
4) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.
5) Flush queue: Non-sequentially fetch first instruction of Interrupt routine.
6) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-22. Service Sequence
Invoked during all interrupt/trap sequences.

3.0 Functional Description (Continued)

3.7.7.2 Trap Sequence: Traps Other Than Trace

1. Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2. Set "Vector" to the value corresponding to the trap type.
   SLAVE: Vector = 3
   ILL: Vector = 4
   SVC: Vector = 5
   DVZ: Vector = 6
   FLG: Vector = 7
   BPT: Vector = 8
   UND: Vector = 10
3. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T and P.
4. Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5. Set "Return Address" to the address of the first byte of the trapped instruction.
6. Perform Service (Vector, Return Address), *Figure 3-22*.

3.7.7.3 Trace Trap Sequence

1. In the Processor Status Register (PSR), clear the P bit.
2. Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3. Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4. Set "Vector" to 9.
5. Set "Return Address" to the address of the next instruction.
6. Perform Service (Vector, Return Address), *Figure 3-22*.

3.8 SLAVE PROCESSOR INSTRUCTIONS

The NS32008 CPU recognizes two groups of instructions as being executable by external Slave Processors:

Floating-Point Instruction Set

Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Section 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a nonexistent Slave Processor. Slave Processor cycles use pins AD0–AD15 as a 16-bit data bus.

Status Combinations:

Send ID (ID): Code 1111
Xfer Operand (OP): Code 1101
Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operation Word.
3	OP	CPU Sends Required Operands.
4	—	Slave Starts Execution. CPU Pre-Fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-23. Slave Processor Protocol

3.8.1 Slave Processor Protocol

Slave Processor instructions have a 3-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1. It identifies the instruction as being a Slave Processor instruction.
2. It specifies which Slave Processor will execute it.
3. It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-23*. While applying Status Code 1111 (Broadcast ID, Section 3.4.2), the CPU transfers the ID Byte on the least-significant half of the data bus (AD0–AD7). All Slave Processors input this Byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this tranfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Section 3.4.2).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this, $\overline{SPC}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Section 3.4.2).

Upon receiving the pulse on SPC, the CPU uses SPC to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Section 3.4.2). This word has the format shown in *Figure 3-24*. If the Q bit ("Quit," Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the Slave vector

3.0 Functional Description (Continued)

in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2).

An exception to the protocol above is a Custom Slave instruction (LCR: Load Custom Register). In executing this instruction, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.8.2 Floating-Point Instructions

Table 3-2 gives the protocols followed for each Floating-Point instruction. The instructions are referenced by their mnemonics. For the bit encoding of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating-Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B=byte, W=word, D=double word). "f" indicates that the instruction specifies a Floating-Point size for the operand (F=32-bit standard floating, L=64-bit Long Floating).

The Returned Value type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-24*).

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating-Point Registers are physically on the Floating-Point Unit and are therefore available without CPU assistance.

TABLE 3-2
Floating-Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note:

D=Double word.

i=Integer size (B,W,D) specified in mnemonic.

f=Floating-point type (F,L) specified in mnemonic.

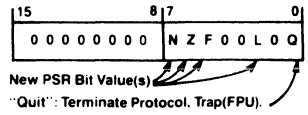
N/A=Not applicable to this instruction.

3.0 Functional Description (Continued)

3.8.3 Custom Slave Instruction

Provided in the NS32008 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the opcode fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-3 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.



TL/EE/6156-37

FIGURE 3-24. Slave Processor Status Word Format

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

TABLE 3-3
Custom Slave Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op.2	none
CCMP0c	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:

D = Double word.

i = Integer size (B, W, D) specified in mnemonic.

c = Custom size (D: 32 bits or Q: 64 bits) specified in mnemonic.

* = Privileged instruction; will trap if CPU is in User Mode.

N/A = Not applicable to this instruction.

4.0 Device Specifications

4.1 NS32008 PIN DESCRIPTIONS

The following is a brief description of all NS32008 pins. The descriptions reference portions of the Functional Description, Section 3.

4.1.1 Supplies

Power (V_{CC}): +5V Positive Supply. Section 3.1.

Logical Ground (GNDL): Ground reference for on-chip logic. Section 3.1.

Buffer Ground (GNDB): Ground reference for on-chip drivers connected to output pins. Section 3.1.

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Section 3.1.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Section 3.2.

Ready (RDY): Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Section 3.4.1.

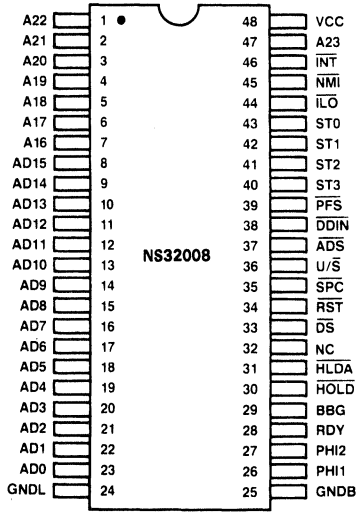
Hold Request ($\overline{\text{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.5.

Note: If the $\overline{\text{HOLD}}$ signal is generated asynchronously, it's set up and hold times may be violated. In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{\text{HLDA}}$ latency. This is to avoid speed degradations in cases of heavy $\overline{\text{HOLD}}$ activity (i.e. DMA controller cycles interleaved with CPU cycles.)

Interrupt ($\overline{\text{INT}}$): Active low. Maskable Interrupt Request. Section 3.7.

Dual-In-Line Package



Top View TL/EE/6156-2

Figure 4-1. NS32008 Connection Diagram

Order Number NS32008D or NS32008N
See NS Package Number D48A or N48A

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low. Non-Maskable Interrupt Request. Section 3.7.

Reset ($\overline{\text{RST}}$): Active low. It initiates a Reset. Section 3.3.

4.1.3 Output Signals

Address Bits 16–23 (A16–A23): These are the most significant eight bits of the memory address bus. Section 3.4.

Address Strobe ($\overline{\text{ADS}}$): Active low. Controls address latches; indicates start of a bus cycle. Section 3.4.

Data Direction In ($\overline{\text{DDIN}}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Section 3.4.

Status (ST0–ST3): Bus cycle status code, ST0 least significant. Section 3.4.2. Encodings are:

0000—Idle: CPU Inactive on Bus
0001—Idle: WAIT Instruction
0010—(Reserved)
0011—Idle: Waiting for Slave
0100—Interrupt Acknowledge, Master
0101—Interrupt Acknowledge, Cascaded
0110—End of Interrupt, Master
0111—End of Interrupt, Cascaded
1000—Sequential Instruction Fetch
1001—Nonsequential Instruction Fetch
1010—Data Transfer
1011—Read Read-Modify-Write Operand
1100—Read for Effective Address
1101—Transfer Slave Operand
1110—Read Slave Status Word
1111—Broadcast Slave ID.

Hold Acknowledge ($\overline{\text{HLDA}}$): Active low. Applied by the CPU in response to $\overline{\text{HOLD}}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Section 3.5.

User/Supervisor (U/$\overline{\text{S}}$): User or Supervisor Mode status. High state indicates User Mode, low indicates Supervisor Mode. Section 3.6.

Interlocked Operation ($\overline{\text{ILO}}$): Active low. Indicates that an interlocked instruction is being executed. Section 3.6.

Program Flow Status ($\overline{\text{PFS}}$): Active low. Pulse indicates beginning of an instruction execution. Section 3.6.

Data Strobe ($\overline{\text{DS}}$): Active low. Data strobe output. Section 3.4.

4.1.4 Input-Output Signals

Address/Data 0–15 (AD0–AD15): In all except Slave Processor bus cycles, pins AD0–AD7 serve as an 8-bit Multiplexed Address/Data bus, and pins AD8–AD15 hold address bits 8–15 throughout the bus cycle. Bit 0 is defined as the least-significant bit. Section 3.4.

In Slave Processor bus cycles, all 16 pins are used as a data bus (Section 3.4.6).

Slave Processor Control ($\overline{\text{SPC}}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of a slave instruction. Section 3.4.6 and Section 3.8. This pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Data Strobe ($\overline{\text{DS}}$): Active low. Data Strobe output. Section 3.4.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages With Respect to GND	−0.5V to +7V
Power Dissipation	1.5 Watt

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS T_A = 0 to + 70°C, V_{CC} = 5V ± 5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC}+0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC}-0.35$		$V_{CC}+0.5$	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	I_{OH} = −400 μA	2.4			V
V_{OL}	Logical 0 Output Voltage	I_{OL} = 2 mA			0.45	V
I_{ILS}	$\overline{SPC}$ Input Current (low)	V_{IN} = 0.4V, $\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Leakage Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{SPC}$	−10		10	μA
$I_{O(OFF)}$	Output Leakage Current (Output Pins in TRI-STATE Condition)	$0.4 \le V_{OUT} \le V_{CC}$	−20		30	μA
I_{CC}	Active Supply Current	$I_{OUT}=0$, $T_A=25°C$		180	300	mA

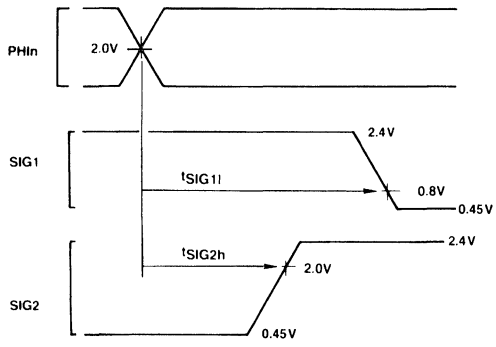
4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 and PHI2 and 0.8V or 2.0V on all other signals as illustrated in *Figures 4-2* and *4-3*, unless specifically stated otherwise.

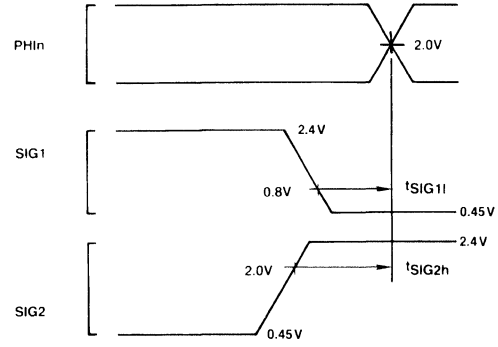
Abbreviations:

L.E.—leading edge
T.E.—trailing edge
R.E.—rising edge
F.E.—falling edge



TL/EE/6156-38

FIGURE 4-2. Timing Specification Standard (Signal Valid After Edge)



TL/EE/6156-39

FIGURE 4-3. Timing Specification Standard (Signal Valid Before Edge)

4.0 Device Specifications (Continued)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32008-6, NS32008-8, NS32008-10

Maximum times assume capacitive loading of 100 pF

Name	Figure	Description	Reference/Conditions	NS32008-6		NS32008-8		NS32008-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4-4	Address Bits 0–7 Valid	after R.E., PHI1 T1		80		65		50	ns
t_{ALh}	4-4	Address Bits 0–7 Hold	after R.E., PHI1 T2	5		5		5		ns
t_{Dv}	4-4	Data Valid (Write Cycle)	after R.E., PHI1 T2		80		65		50	ns
t_{Dh}	4-4	Data Hold (Write Cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{AHv}	4-4	Address Bits 8–23 Valid	after R.E., PHI1 T1		95		75		50	ns
t_{AHh}	4-4	Address Bits 8–23 Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	4-5	Address Bits 0–7 Set Up to $\overline{ADS}$ T.E.	before $\overline{ADS}$ reaches 2.0V	25		25		25		ns
t_{AHADSs}	4-5	Address Bits 8–23 Set Up to $\overline{ADS}$ T.E.	before $\overline{ADS}$ reaches 2.0V	25		25		25		ns
t_{ALADSh}	4-10	Address Bits 0–7 Hold from $\overline{ADS}$ T.E.	after $\overline{ADS}$ reaches 2.0V	15		15		15		ns
t_{ALf}	4-5	Address Bits 0–7 Floating	after R.E., PHI1 T2		25		25		25	ns
t_{STv}	4-4	Status (ST0–ST3) Valid	after R.E., PHI1 T4 (before T1, see note)		90		70		45	ns
t_{STh}	4-4	Status (ST0–ST3) Hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	4-5	$\overline{DDIN}$ Signal Valid	after R.E., PHI1 T1		83		62		50	ns
t_{DDINh}	4-5	$\overline{DDIN}$ Signal Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	4-4	$\overline{ADS}$ Signal Active (Low)	after R.E., PHI1 T1		55		45		35	ns
t_{ADSia}	4-4	$\overline{ADS}$ Signal Inactive	after R.E., PHI2 T1		60		55		45	ns
t_{ADSw}	4-4	$\overline{ADS}$ Pulse Width	at 0.8V (both edges)	50		40		30		ns
t_{DSa}	4-4	$\overline{DS}$ Signal Active (Low)	after R.E., PHI1 T2		70		60		45	ns
t_{DSia}	4-4	$\overline{DS}$ Signal Inactive	after R.E., PHI1 T4	10	60	10	50	10	40	ns
t_{ALf}	4-6	AD0–AD7 Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{AHf}	4-6	A8–A23 Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{ADSf}	4-6	$\overline{ADS}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	4-6	$\overline{DDIN}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HLDAa}	4-6	$\overline{HLDA}$ Signal Active (Low)	after R.E., PHI1 Ti		100		90		75	ns
t_{HLDAia}	4-8	$\overline{HLDA}$ Signal Inactive	after R.E., PHI1 Ti		100		90		75	ns
t_{ADSr}	4-8	$\overline{ADS}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINr}	4-8	$\overline{DDIN}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{SPCa}	4-13	$\overline{SPC}$ Output Active (Low)	after R.E., PHI1 T1		50		45		35	ns
t_{SPCia}	4-13	$\overline{SPC}$ Output Inactive	after R.E., PHI1 T4		50		45		35	ns
t_{SPCnf}	4-15	$\overline{SPC}$ Output Nonforcing	after R.E., PHI2 T4		40		25		10	ns
t_{Dv}	4-11	Data Valid (Slave Processor Write)	after R.E., PHI1 T1		80		65		50	ns
t_{Dh}	4-11	Data Hold (Slave Processor Write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	4-15	$\overline{PFS}$ Pulse Width	at 0.8V (both edges)	70		70		70		ns
t_{PFSa}	4-15	$\overline{PFS}$ Pulse Active (Low)	after R.E., PHI2		70		60		50	ns
t_{PFSia}	4-15	$\overline{PFS}$ Pulse Inactive	after R.E., PHI2		70		60		50	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32008-6, NS32008-8, NS32008-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ILOs}	4-17	$\overline{ILO}$ Signal Setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns
t_{ILOh}	4-18	$\overline{ILO}$ Signal Hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t_{ILOa}	4-19	$\overline{ILO}$ Signal Active (Low)	after R.E., PHI1		70		65		55	ns
t_{ILOia}	4-19	$\overline{ILO}$ Signal Inactive	after R.E., PHI1		70		65		55	ns
t_{USv}	4-20	U/$\overline{S}$ Signal Valid	after R.E., PHI1 T4		70		60		45	ns
t_{USh}	4-20	U/$\overline{S}$ Signal Hold	after R.E., PHI1 T1	10		10		10		ns
t_{NSPF}	4-16b	Nonsequential Fetch to Next $\overline{PFS}$ Clock Cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	4-16a	$\overline{PFS}$ Clock Cycle to Next Non-Sequential Fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	4-25	Last Operand Transfer of an Instruction to Next $\overline{PFS}$ clock Cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t_{Cp}

Note 1: Timing parameters for components with an "S" suffix are not guaranteed compatible with an NS32081 Slave Processor at a clock rate greater than 4 MHz.

Note 2: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . .Ti, T4, T1. . .". If the CPU was not idling, the sequence will be: ". . .T4, T1. . .".

4.4.2.2 Input Signal Requirements: NS32008-6, NS32008-8, NS32008-10

Name	Figure	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
				Min	Max	Min	Max	Min	Max	
t_{PWR}	4-21	Power Stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	4-5	Data in Setup (Read Cycle)	before F.E., PHI2 T3	20		15		10		ns
t_{DIh}	4-5	Data in Hold (Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	4-6	$\overline{HOLD}$ Active (Low) Setup Time (See Note)	before F.E., PHI2 TX1	25		25		25		ns
t_{HLDia}	4-8	$\overline{HOLD}$ Inactive Setup Time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	4-6	$\overline{HOLD}$ Hold Time	after R.E., PHI1 TX2	0		0		0		ns
t_{RDYs}	4-9, 4-10	RDY Setup Time	before F.E., PHI2 T2 or T3	25		25		25		ns
t_{RDYh}	4-9, 4-10	RDY Hold Time	after F.E., PHI1 T3	0		0		0		ns
t_{RSTs}	4-21, 4-22	$\overline{RST}$ Setup Time	before F.E., PHI1	20		15		10		ns
t_{RSTw}	4-22	$\overline{RST}$ Pulse Width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	4-23	$\overline{INT}$ Setup Time	before T.E., PHI1	20		20		20		ns
t_{NMIw}	4-28	$\overline{NMI}$ Pulse Width	at 0.8V (both edges)	70		70		70		ns
t_{DIs}	4-12	Data Setup (Slave Read Cycle)	before F.E., PHI2 T1	20		15		10		ns
t_{DIh}	4-12	Data Hold (Slave Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	4-13	$\overline{SPC}$ Pulse Delay from Slave	after R.E., PHI2 T4	17		13		10		ns
t_{SPCs}	4-13	$\overline{SPC}$ Setup Time	Before F.E., PHI1	42		32		25		ns
t_{SPCw}	4-13	SPC Pulse Width from Slave Processor (Async. Input)	at 0.8V (both edges)	30		25		20		ns

NOTE: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, and the state of the RDY input.

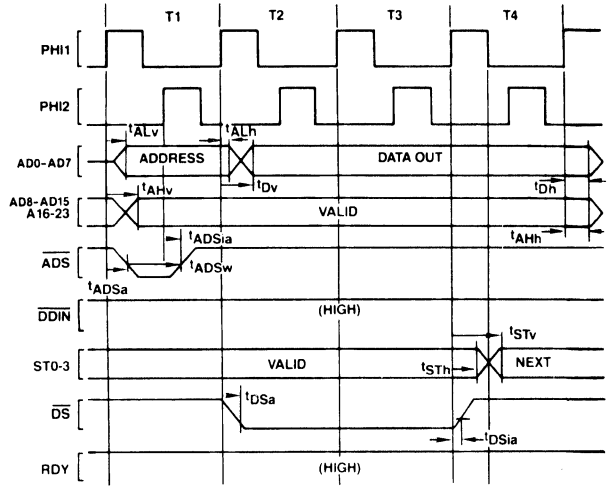
4.0 Device Specifications (Continued)

4.4.2.3 Clocking Requirements: NS32008-6, NS32008-8, NS32008-10

Name	Figure	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Unit
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4-14	PHI1, PHI2 Rise Time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	4-14	PHI1, PHI2 Fall Time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	4-14	Clock Period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4-14	PHI1, PHI2 Pulse Width	at 2.0V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 14		0.5 t_{Cp} − 12		0.5 t_{Cp} − 10		ns
$t_{CLh(1,2)}$	4-14	PHI1, PHI2 High Time	at V_{CC} − 0.9V on PHI1, PHI2 (both edges)	0.5t_{Cp}−18		0.5t_{Cp}−17		0.5t_{Cp}−15		ns
$t_{nOVL(1,2)}$	4-14	Non-Overlap Time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI1, PHI2	0	7	0	7	0	7	ns
t_{nOVLas}		Non-Overlap Asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	at 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}		PHI1, PHI2 Asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	at 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

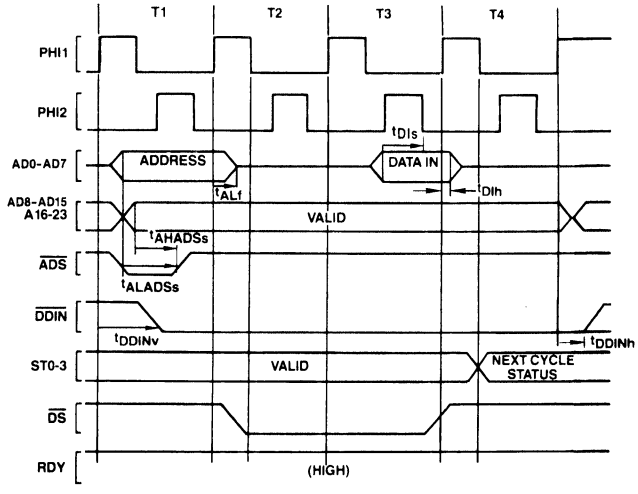
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



TL/EE/6156–40

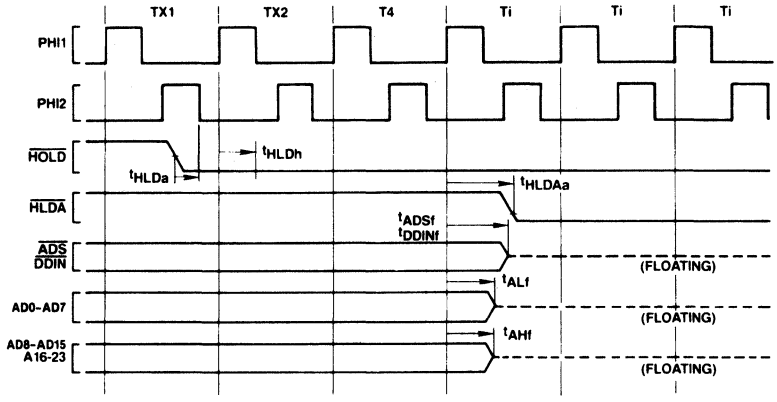
FIGURE 4-4. Write Cycle



TL/EE/6156–41

FIGURE 4-5. Read Cycle

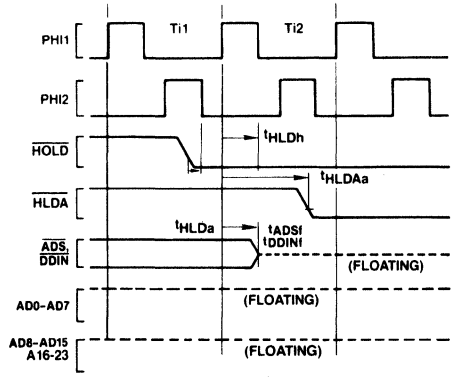
4.0 Device Specifications (Continued)



TL/EE/6156-42

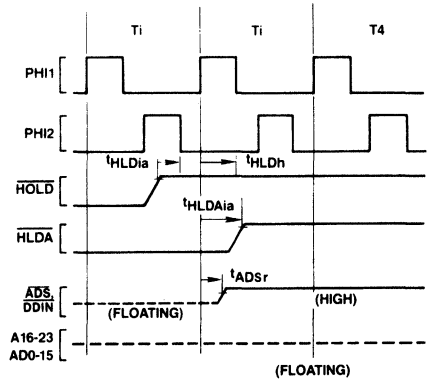
Note: Whenever the CPU is not idling (not in Ti), the $\overline{HOLD}$ request ($\overline{HOLD}$ low) must be active t_{HLDa} before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4(TX1) and stay low until t_{HLDh} after the rising edge of PHI1 of the clock cycle that precedes T4(TX2) for the request to be acknowledged.

FIGURE 4-6. Floating by $\overline{HOLD}$ Timing (CPU Not Idle Initially)



TL/EE/6156-43

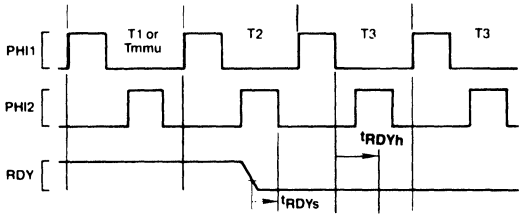
FIGURE 4-7. Floating by $\overline{HOLD}$ Timing (CPU Initially Idle)



TL/EE/6156-44

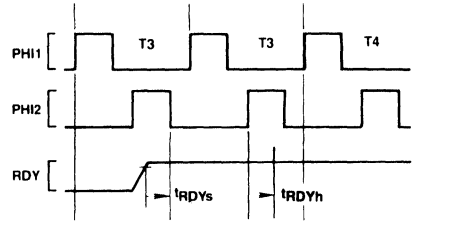
FIGURE 4-8. Release from $\overline{HOLD}$

4.0 Device Specifications (Continued)



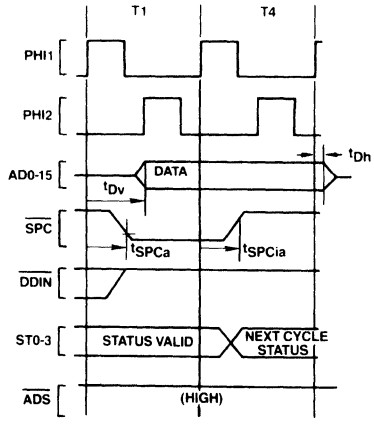
TL/EE/6156-45

FIGURE 4-9. Ready Sampling (CPU Initially READY)



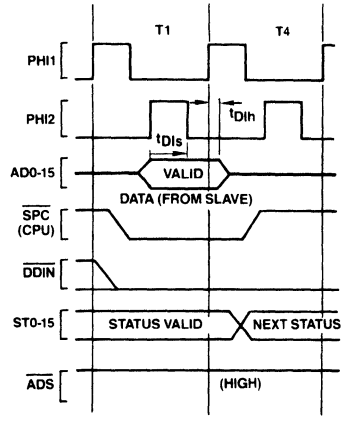
TL/EE/6156-46

FIGURE 4-10. Ready Sampling (CPU Initially NOT READY)



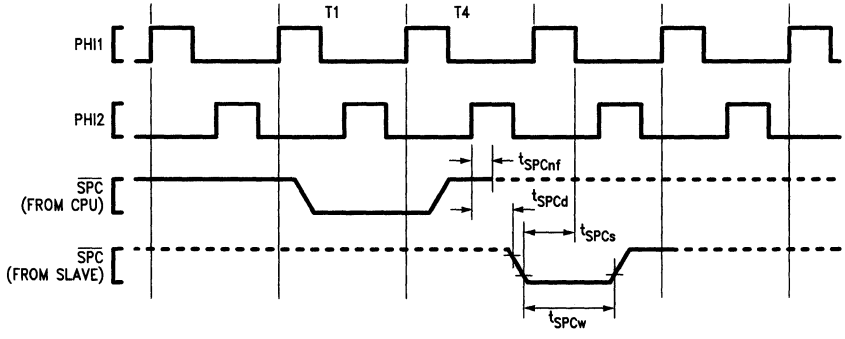
TL/EE/6156-47

FIGURE 4-11. Slave Processor Write Timing



TL/EE/6156-48

FIGURE 4-12. Slave Processor Read Timing

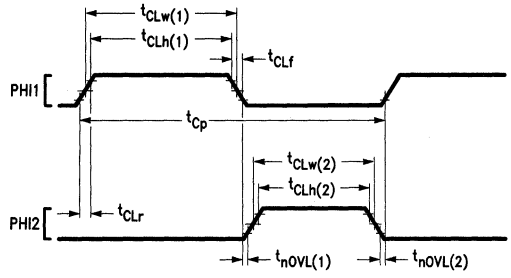


TL/EE/6156-82

Note: After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{SPC}$ high with internal 5 kΩ pullup.

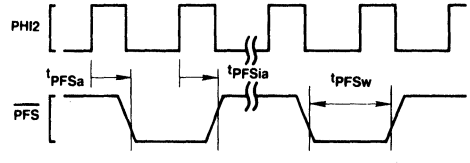
FIGURE 4-13. $\overline{SPC}$ Timing

4.0 Device Specifications (Continued)



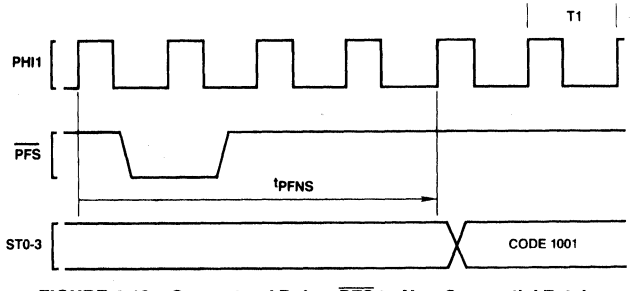
TL/EE/6156-50

FIGURE 4-14. Clock Waveforms



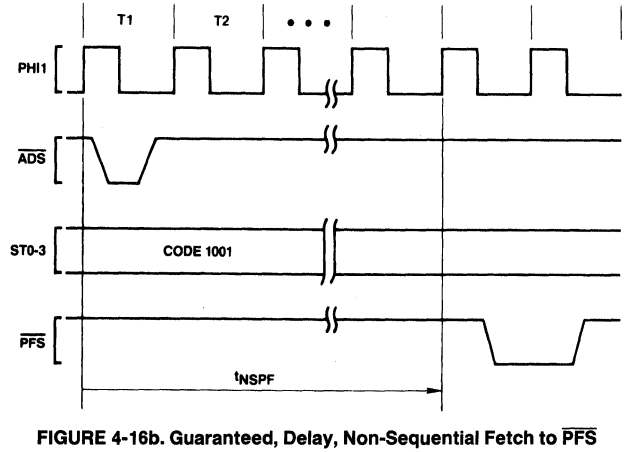
TL/EE/6156-51

FIGURE 4-15. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/6156-52

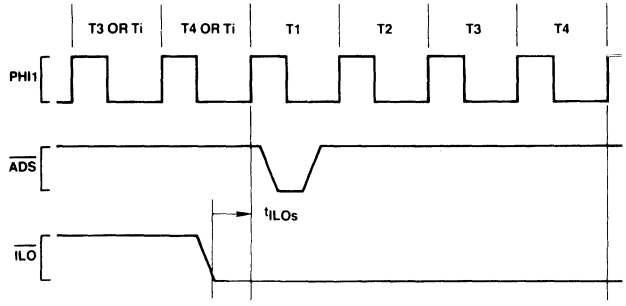
FIGURE 4-16a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/6156-53

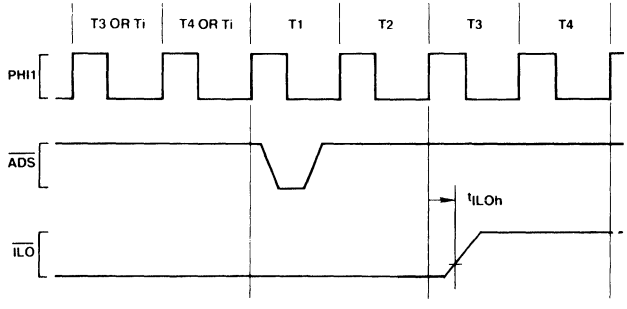
FIGURE 4-16b. Guaranteed, Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



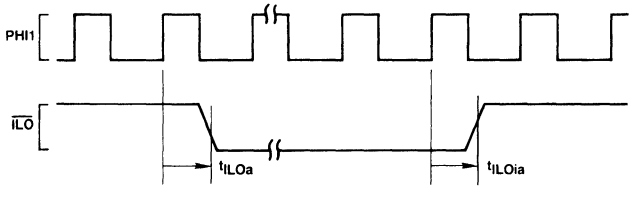
TL/EE/6156-54

FIGURE 4-17. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction



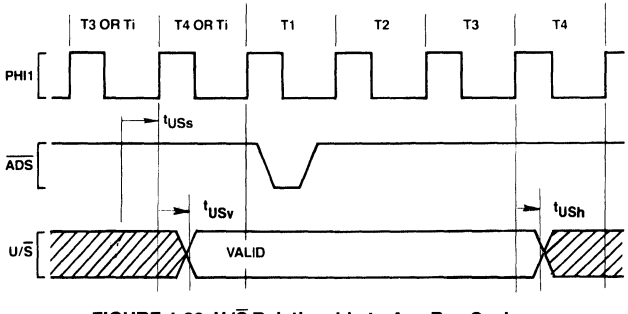
TL/EE/6156-55

FIGURE 4-18. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction



TL/EE/6156-56

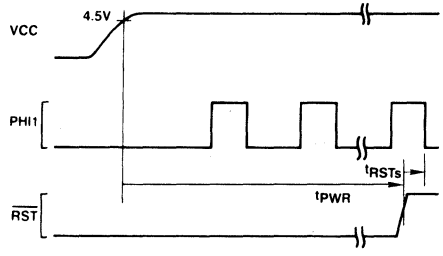
FIGURE 4-19. Relationship of $\overline{ILO}$ to Any Clock Cycle



TL/EE/6156-57

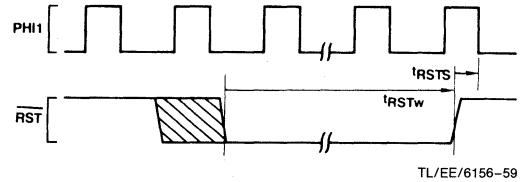
FIGURE 4-20. U/$\overline{S}$ Relationship to Any Bus Cycle—Guarantee Valid Interval

4.0 Device Specifications (Continued)



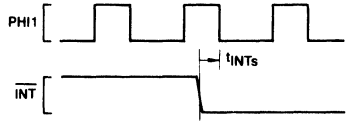
TL/EE/6156–58

FIGURE 4-21. Power-On Reset



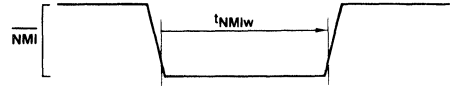
TL/EE/6156–59

FIGURE 4-22. Non-Power-On Reset



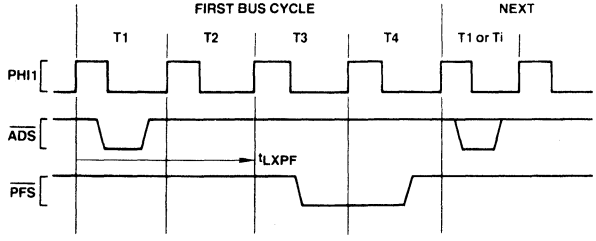
TL/EE/6156–60

FIGURE 4-23. $\overline{INT}$ Interrupt Signal Detection



TL/EE/6156–61

FIGURE 4-24. $\overline{NMI}$ Interrupt Signal Timing



TL/EE/6156–62

Note: In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

FIGURE 4-25. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction

Appendix A: Instruction Formats

NOTATIONS

i ..... Integer Type Field
B = 00 (Byte)
W = 01 (Word)
D = 11 (Double Word)

f ..... Floating-Point Type Field
F = 1 (Standard Floating: 32 bits)
L = 0 (Long Floating: 64 bits)

c ..... Custom Type Field
D = 1 (Double Word)
Q = 0 (Quad Word)

op ..... Operation Code
Valid encodings shown with each format.

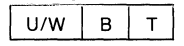
gen, gen1,
gen2 ..... General Addressing Mode Field.
See Section 2.2 for encodings.

reg ..... General Purpose Register Number

cond ..... Condition Code Field
0000 = EQual: Z = 1
0001 = Not Equal: Z = 0
0010 = Carry Set: C = 1
0011 = Carry Clear: C = 0
0100 = Higher: L = 1
0101 = Lower or Same: L = 0
0110 = Greater Than: N = 1
0111 = Less or Equal: N = 0
1000 = Flag Set: F = 1
1001 = Flag Clear: F = 0
1010 = LOwer: L = 0 and Z = 0
1011 = Higher or Same: L = 1 or Z = 1
1100 = Less Than: N = 0 and Z = 0
1101 = Greater or Equal: N = 1 or Z = 1
1110 = (Unconditionally True)
1111 = (Unconditionally False)

short ..... Short Immediate Value. May contain:
quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB
cond: Condition Code (above), in Scond.
areg: CPU Dedicated Register, in LPR, SPR.
0000 = US
0001 – 0111 = (Reserved)
1000 = FP
1001 = SP
1010 = SB
1011 = (Reserved)
1100 = (Reserved)
1101 = PSR
1110 = INTBASE
1111 = MOD

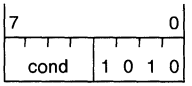
Options: in String Instructions



T = Translated
B = Backward
U/W = 00: None
01: While Match
11: Until Match

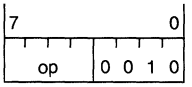
Configuration bits, in SETCFG:

TL/EE/6156–63



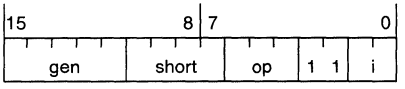
Format 0

Bcond (BR)



Format 1

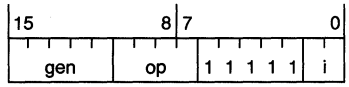
BSR	–0000	ENTER	–1000
RET	–0001	EXIT	–1001
CXP	–0010	NOP	–1010
RXP	–0011	WAIT	–1011
RETT	–0100	DIA	–1100
RETI	–0101	FLAG	–1101
SAVE	–0110	SVC	–1110
RESTORE	–0111	BPT	–1111



Format 2

ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		

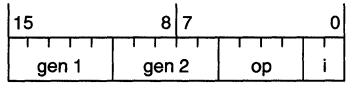
Appendix A: Instruction Formats (Continued)



Format 3

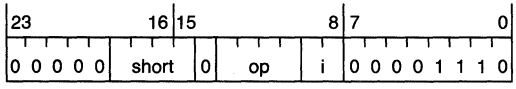
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

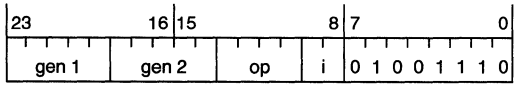
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

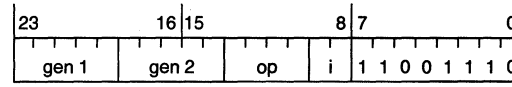
Trap (UND) on 1XXX, 01XX



Format 6

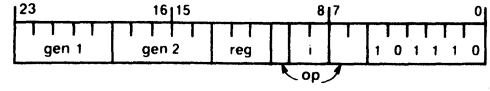
ROT	−0000	NEG	−1000
ASH	−0001	Trap (UND)	−1010
CBIT	−0010	SUBP	−1011
CBITI	−0011	ABS	−1100
Trap (UND)	−0100	COM	−1101
LSH	−0101	IBIT	−1110
SBIT	−0110	ADDP	−1111
SBITI	−0111		

Trap (UND) on all others



Format 7

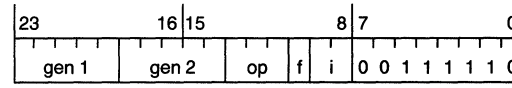
MOVM	−0000	MUL	−1000
CMPM	−0001	MEI	−1001
INSS	−0010	Trap (UND)	−1010
EXTS	−0011	DEI	−1011
MOVXBW	−0100	QUO	−1100
MOVZBW	−0101	REM	−1101
MOVZiD	−0110	MOD	−1110
MOVXiD	−0111	DIV	−1111



TL/EE/6156–64

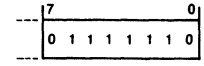
Format 8

EXT	−000	INDEX	−100
CVTP	−001	FFS	−101
INS	−010		
CHECK	−011		



Format 9

MOVif	−000	ROUND	−100
LFSR	−001	TRUNC	−101
MOVLF	−010	SFSR	−110
MOVFL	−011	FLOOR	−111

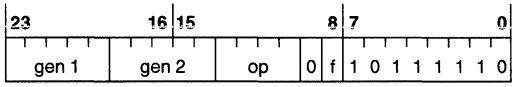


TL/EE/6156–65

Format 10

Trap (UND) Always

Appendix A: Instruction Formats (Continued)



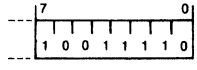
Format 11

ADDf	−0000	DIVf	−1000
MOVf	−0001	Trap (Slave)	−1001
CMPf	−0010	Trap (UND)	−1010
Trap (Slave)	−0011	Trap (UND)	−1011
SUBf	−0100	MULf	−1100
NEGf	−0101	ABSf	−1110
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111

TL/EE/6156-76

Format 12

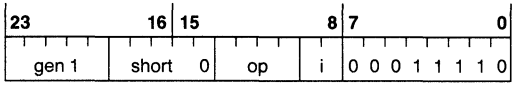
Trap (UND) Always



TL/EE/6156-77

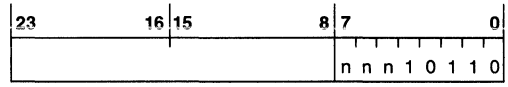
Format 13

Trap (UND) Always



Format 14

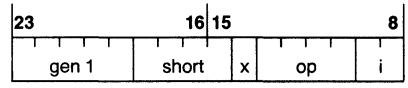
Trap (UND) Always



Format 15
(Custom Slave)

nnn **Operation Word Format**

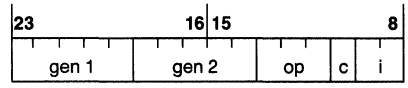
000



Format 15.0

CATST0	−0000	LCR	−0010
CATST1	−0001	SCR	−0011

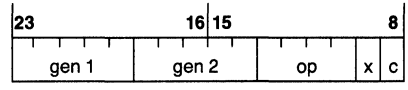
001



Format 15.1

CCV3	−000	CCV2	−100
LCSR	−001	CCV1	−101
CCV5	−010	SCSR	−110
CCV4	−011	CCV0	−111

101



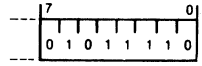
Format 15.5

CCAL0	−0000	CCAL3	−1000
CMOV0	−0001	CMOV3	−1001
CCMP0	−0010	Trap (UND)	−1010
CCMP1	−0011	Trap (UND)	−1011
CCAL1	−0100	CCAL2	−1100
CMOV2	−0101	CMOV1	−1101
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111

If nnn=010, 011, 100, 110, 111, then Trap (UND) Always

2

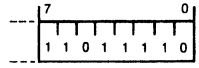
Appendix A: Instruction Formats (Continued)



TL/EE/6156-78

Format 16

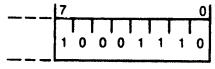
Trap (UND) Always



TL/EE/6156-79

Format 17

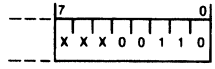
Trap (UND) Always



TL/EE/6156-80

Format 18

Trap (UND) Always

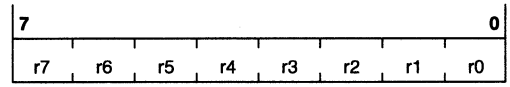


TL/EE/6156-81

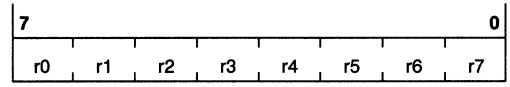
Format 19

Trap (UND) Always

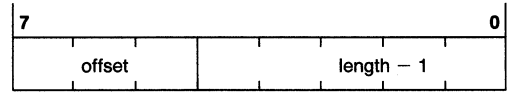
Implied Immediate Encodings:



Register Mask, appended to SAVE, ENTER



Register Mask, appended to RESTORE, EXIT



Offset/Length Modifier, appended to INSS, EXTS

Section 3
Slave Processors

Section 3 Contents

NS32081-6, -8, -10 Floating Point Unit (FPU) 3-3
NS32082-6, -8, -10 Memory Management Unit (MMU) 3-20
NS32382-10, -15 Memory Management Unit 3-59

National Semiconductor Corporation

NS32081-6/NS32081-8/NS32081-10 Floating-Point Unit

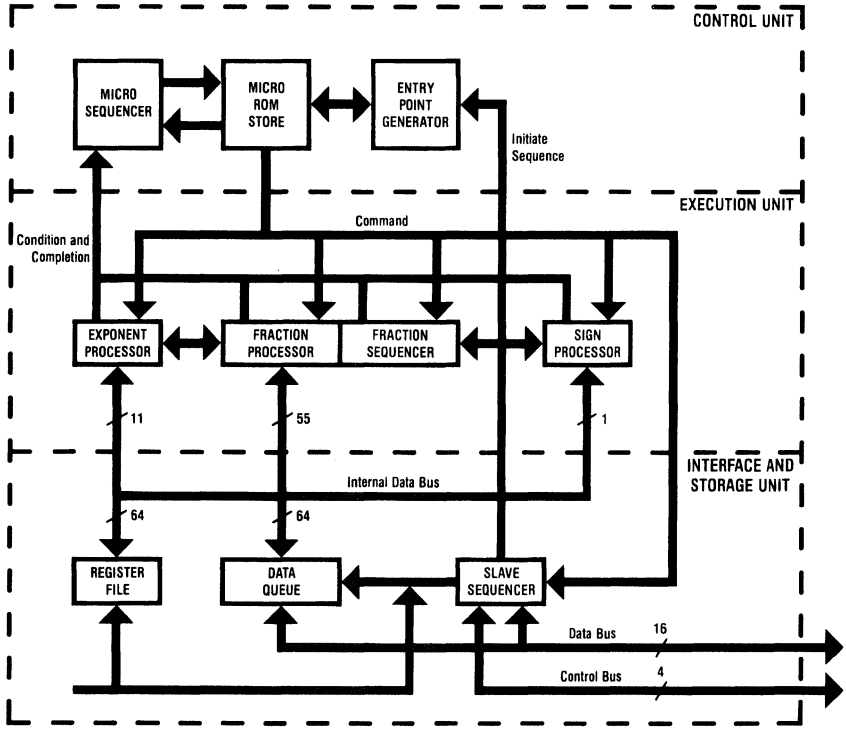
General Description

The NS32081 Floating-Point Unit functions as a slave processor in National Semiconductor's Series 32000® microprocessor family. It provides a high-speed floating-point instruction set for any Series 32000 family CPU, while remaining architecturally consistent with the full two-address architecture and powerful addressing modes of the Series 32000 micro-processor family.

Features

- Eight on-chip data registers
- 32-bit and 64-bit operations
- Supports proposed IEEE standard for binary floating-point arithmetic, Task P754
- Directly compatible with NS32016, NS32008 and NS32032 CPUs
- High-speed XMOS™ technology
- Single 5V supply
- 24-pin dual in-line package

Block Diagram



TL/EE/5234–1

Table of Contents

1.0 PRODUCT INTRODUCTION

1.1 Operand Formats

1.1.1 Normalized Numbers

1.1.2 Zero

1.1.3 Reserved Operands

1.1.4 Integers

1.1.5 Memory Representations

2.0 ARCHITECTURAL DESCRIPTION

2.1 Programming Model

2.1.1 Floating-Point Registers

2.1.2 Floating-Point Status Register (FSR)

2.1.2.1 FSR Mode Control Fields

2.1.2.2 FSR Status Fields

2.1.2.3 FSR Software Field (SWF)

2.2 Instruction Set

2.2.1 General Instruction Format

2.2.2 Addressing Modes

2.2.3 Floating-Point Instruction Set

2.3 Traps

3.0 FUNCTIONAL DESCRIPTION

3.1 Power and Grounding

3.2 Clocking

3.3 Resetting

3.0 FUNCTIONAL DESCRIPTION (Continued)

3.4 Bus Operation

3.4.1 Bus Cycles

3.4.2 Operand Transfer Sequences

3.5 Instruction Protocols

3.5.1 General Protocol Sequence

3.5.2 Floating-Point Protocols

4.0 DEVICE SPECIFICATIONS

4.1 Pin Descriptions

4.1.1 Supplies

4.1.2 Input Signals

4.1.3 Input/Output Signals

4.2 Absolute Maximum Ratings

4.3 Electrical Characteristics

4.4 Switching Characteristics

4.4.1 Definitions

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays

4.4.2.2 Input Signals Requirements

4.4.2.3 Clocking Requirements

4.4.3 Timing Diagrams

Appendix A: Instruction Formats

List of Illustrations

Floating-Point Operand Formats 1-1
Register Set 2-1
The Floating-Point Status Register 2-2
General Instruction Format 2-3
Index Byte Format 2-4
Displacement Encodings 2-5
Floating-Point Instruction Formats 2-6
Recommended Supply Connections 3-1
Power-On Reset Requirements 3-2
General Reset Timing 3-3
System Connection Diagram 3-4
Slave Processor Read Cycle 3-5
Slave Processor Write Cycle 3-6
FPU Protocol Status Word Format 3-7
Dual-In-Line Package 4-1
Timing Specification Standard (Signal Valid After Clock Edge) 4-2
Timing Specification Standard (Signal Valid Before Clock Edge) 4-3
Clock Timing 4-4
Power-On-Reset 4-5
Non-Power-On-Reset 4-6
Read Cycle From FPU 4-7
Write Cycle To FPU 4-8
$\overline{SPC}$ Pulse from FPU 4-9
$\overline{RST}$ Release Timing 4-10

List of Tables

Sample F Fields 1-1
Sample E Fields 1-2
Normalized Number Ranges 1-3
Series 32000 Family Addressing Modes 2-1
General Instruction Protocol 3-1
Floating-Point Instruction Protocols 3-2

1.0 Product Introduction

The NS32081 Floating-Point Unit (FPU) provides high speed floating-point operations for the Series 32000 family, and is fabricated using National high-speed XMOS technology. It operates as a slave processor for transparent expansion of the Series 32000 CPU's basic instruction set. The FPU can also be used with other microprocessors as a peripheral device by using additional TTL interface logic. The NS32081 is compatible with the IEEE Floating-Point Formats by means of its hardware and software features.

1.1 OPERAND FORMATS

The NS32081 FPU operates on two floating-point data types—single precision (32 bits) and double precision (64 bits). Floating-point instruction mnemonics use the suffix F (Floating) to select the single precision data type, and the suffix L (Long Floating) to select the double precision data type.

A floating-point number is divided into three fields, as shown in *Figure 1-1*.

The F field is the fractional portion of the represented number. In Normalized numbers (Section 1.1.1), the binary point is assumed to be immediately to the left of the most significant bit of the F field, with an implied 1 bit to the left of the binary point. Thus, the F field represents values in the range $1.0 \leq x \leq 2.0$.

TABLE 1-1. Sample F Fields

F Field	Binary Value	Decimal Value
000...0	1.000...0	1.000...0
010...0	1.010...0	1.250...0
100...0	1.100...0	1.500...0
110...0	1.110...0	1.750...0
	↑	
	Implied Bit	

The E field contains an unsigned number that gives the binary exponent of the represented number. The value in the E field is biased; that is, a constant bias value must be subtracted from the E field value in order to obtain the true exponent. The bias value is $011\ldots11_2$, which is either 127 (single precision) or 1023 (double precision). Thus, the true exponent can be either positive or negative, as shown in Table 1-2.

TABLE 1-2. Sample E Fields

E Field	F Field	Represented Value
011...110	100...0	$1.5\times2^{-1} = 0.75$
011...111	100...0	$1.5\times2^{0} = 1.50$
100...000	100...0	$1.5\times2^{1} = 3.00$

Two values of the E field are not exponents. 11...11 signals a reserved operand (Section 2.1.3). 00...00 represents the number zero if the F field is also all zeroes, otherwise it signals a reserved operand.

The S bit indicates the sign of the operand. It is 0 for positive and 1 for negative. Floating-point numbers are in sign-magnitude form, that is, only the S bit is complemented in order to change the sign of the represented number.

1.1.1 Normalized Numbers

Normalized numbers are numbers which can be expressed as floating-point operands, as described above, where the E field is neither all zeroes nor all ones.

The value of a Normalized number can be derived by the formula:

$$(-1)^S \times 2^{(E\text{-}Bias)} \times (1 + F)$$

The range of Normalized numbers is given in Table 1-3.

1.1.2 Zero

There are two representations for zero—positive and negative. Positive zero has all-zero F and E fields, and the S bit is zero. Negative zero also has all-zero F and E fields, but its S bit is one.

1.1.3 Reserved Operands

The proposed IEEE Standard for Binary Floating-Point Arithmetic (Task P754) provides for certain exceptional forms of floating-point operands. The NS32081 FPU treats these forms as reserved operands. The reserved operands are:

- Positive and negative infinity
- Not-a-Number (NaN) values
- Denormalized numbers

Both Infinity and NaN values have all ones in their E fields. Denormalized numbers have all zeroes in their E fields and non-zero values in their F fields.

The NS32081 FPU causes an Invalid Operation trap (Section 2.1.2.2) if it receives a reserved operand, unless the operation is simply a move (without conversion). The FPU does not generate reserved operands as results.

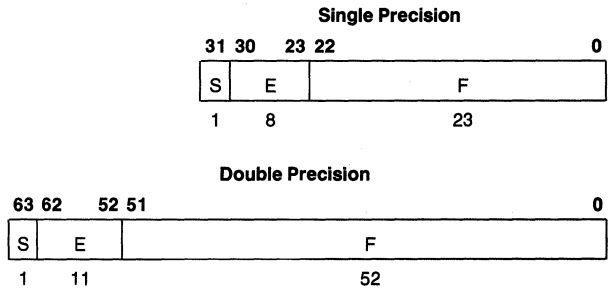


FIGURE 1-1. Floating-Point Operand Formats

1.0 Product Introduction (Continued)

TABLE 1-3. Normalized Number Ranges

	Single Precision	Double Precision
Most Positive	$2^{127}\times(2-2^{-23})$ $=3.40282346\times10^{38}$	$2^{1023}\times(2-2^{-52})$ $=1.7976931348623157\times10^{308}$
Least Positive	2^{-126} $=1.17549436\times10^{-38}$	2^{-1022} $=2.2250738585072014\times10^{-308}$
Least Negative	$-(2^{-126})$ $=-1.17549436\times10^{-38}$	$-(2^{-1022})$ $=-2.2250738585072014\times10^{-308}$
Most Negative	$-2^{127}\times(2-2^{-23})$ $=-3.40282346\times10^{38}$	$-2^{1023}\times(2-2^{-52})$ $=-1.7976931348623157\times10^{308}$

Note: The values given are extended one full digit beyond their represented accuracy to help in generating rounding and conversion algorithms.

1.1.4 Integers

In addition to performing floating-point arithmetic, the NS32081 FPU performs conversions between integer and floating-point data types. Integers are accepted or generated by the FPU as two's complement values of byte (8 bits), word (16 bits) or double word (32 bits) length.

1.1.5 Memory Representations

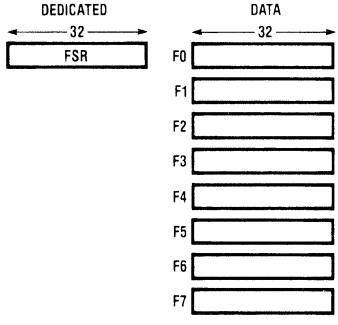
The NS32081 FPU does not directly access memory. However, it is cooperatively involved in the execution of a set of two-address instructions with its Series 32000 Family CPU. The CPU determines the representation of operands in memory.

In the Series 32000 family of CPUs, operands are stored in memory with the least significant byte at the lowest byte address. The only exception to this rule is the Immediate addressing mode, where the operand is held (within the instruction format) with the most significant byte at the lowest address.

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes nine registers that are implemented on the NS32081 Floating-Point Unit (FPU).



TL/EE/5234-4

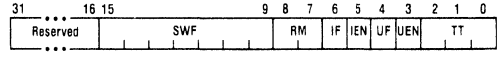
FIGURE 2-1. Register Set

2.1.1 Floating-Point Registers

There are eight registers (F0–F7) on the NS32081 FPU for providing high-speed access to floating-point operands. Each is 32 bits long. A floating-point register is referenced whenever a floating-point instruction uses the Register addressing mode (Section 2.2.2) for a floating-point operand. All other Register mode usages (i.e., integer operands) refer to the General Purpose Registers (R0–R7) of the CPU, and the FPU transfers the operand as if it were in memory. When the Register addressing mode is specified for a double precision (64-bit) operand, a pair of registers holds the operand. The programmer must specify the even register of the pair. The even register contains the least significant half of the operand and the next consecutive register contains the most significant half.

2.1.2 Floating-Point Status Register (FSR)

The Floating-Point Status Register (FSR) selects operating modes and records any exceptional conditions encountered during execution of a floating-point operation. *Figure 2-2* shows the format of the FSR.



TL/EE/5234-5

FIGURE 2-2. The Floating-Point Status Register

2.1.2.1 FSR Mode Control Fields

The FSR mode control fields select FPU operation modes. The meanings of the FSR mode control bits are given below.

Rounding Mode (RM): Bits 7 and 8. This field selects the rounding method. Floating-point results are rounded whenever they cannot be exactly represented. The rounding modes are:

00 Round to nearest value. The value which is nearest to the exact result is returned. If the result is exactly halfway between the two nearest values the even value (LSB = 0) is returned.

01 Round toward zero. The nearest value which is closer to zero or equal to the exact result is returned.

2.0 Architectural Description (Continued)

10 Round toward positive infinity. The nearest value which is greater than or equal to the exact result is returned.

11 Round toward negative infinity. The nearest value which is less than or equal to the exact result is returned.

Underflow Trap Enable (UEN): Bit 3. If this bit is set, the FPU requests a trap whenever a result is too small in absolute value to be represented as a normalized number. If it is not set, any underflow condition returns a result of exactly zero.

Inexact Result Trap Enable (IEN): Bit 5. If this bit is set, the FPU requests a trap whenever the result of an operation cannot be represented exactly in the operand format of the destination. If it is not set, the result is rounded according to the selected rounding mode.

2.1.2.2 FSR Status Fields

The FSR Status Fields record exceptional conditions encountered during floating-point data processing. The meanings of the FSR status bits are given below:

Trap Type (TT): bits 0-2. This 3-bit field records any exceptional condition detected by a floating-point instruction. The TT field is loaded with zero whenever any floating-point instruction except LFSR or SFSR completes without encountering an exceptional condition. It is also set to zero by a hardware reset or by writing zero into it with the Load FSR (LFSR) instruction. Underflow and Inexact Result are always reported in the TT field, regardless of the settings of the UEN and IEN bits.

000 No exceptional condition occurred.

001 Underflow. A non-zero floating-point result is too small in magnitude to be represented as a normalized floating-point number in the format of the destination operand. This condition is always reported in the TT field and UF bit, but causes a trap only if the UEN bit is set. If the UEN bit is not set, a result of Positive Zero is produced, and no trap occurs.

010 Overflow. A result (either floating-point or integer) of a floating-point instruction is too great in magnitude to be held in the format of the destination operand. Note that rounding, as well as calculations, can cause this condition.

011 Divide by zero. An attempt has been made to divide a non-zero floating-point number by zero. Dividing zero by zero is considered an Invalid Operation instead (below).

100 Illegal Instruction. Two undefined floating-point instruction forms are detected by the FPU as being illegal. The binary formats causing this trap are:

xxxxxxxxx0011xx10111110

xxxxxxxxx1001xx10111110

101 Invalid Operation. One of the floating-point operands of a floating-point instruction is a Reserved operand, or an attempt has been made to divide zero by zero using the DIVf instruction.

110 Inexact Result. The result (either floating-point or integer) of a floating-point instruction cannot be represented exactly in the format of the destination operand, and a rounding step must alter it to fit. This condition is always reported in the TT field and IF bit unless any other exceptional condition has occurred in the same instruction. In this case, the TT field always contains the code for the other exception and the IF bit is not altered. A trap is caused by this condition only if the IEN bit is set; otherwise the result is rounded and delivered, and no trap occurs.

111 (Reserved for future use.)

Underflow Flag (UF): Bit 4. This bit is set by the FPU whenever a result is too small in absolute value to be represented as a normalized number. Its function is not affected by the state of the UEN bit. The UF bit is cleared only by writing a zero into it with the Load FSR instruction or by a hardware reset.

Inexact Result Flag (IF): Bit 6. This bit is set by the FPU whenever the result of an operation must be rounded to fit within the destination format. The IF bit is set only if no other error has occurred. It is cleared only by writing a zero into it with the Load FSR instruction or by a hardware reset.

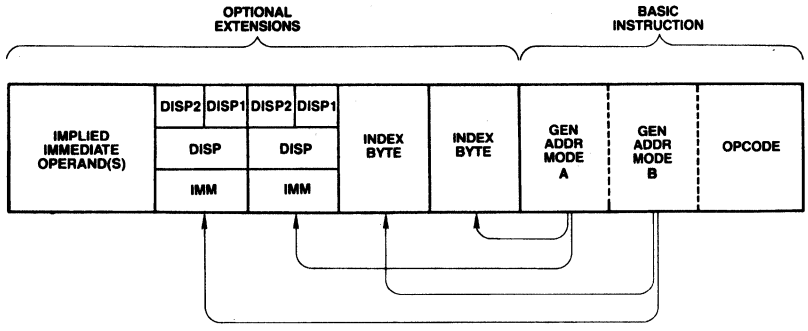
2.1.2.3 FSR Software Field (SWF)

Bits 9-15 of the FSR hold and display any information written to them (using the LFSR and SFSR instructions), but are not otherwise used by FPU hardware. They are reserved for use with NSC floating-point extension software.

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-3 shows the general format of an Series 32000 instruction. The Basic Instruction is one to three bytes long



TL/EE/5234-6

FIGURE 2-3. General Instruction Format

2.0 Architectural Description (Continued)

and contains the opcode and up to two 5-bit General Addressing Mode (Gen) fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

The only form of extension issued to the NS32081 FPU is an Immediate operand. Other extensions are used only by the CPU to reference memory operands needed by the FPU.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-4*.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in *Figure 2-5*, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first.

Some non-FPU instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition.

2.2.2 Addressing Modes

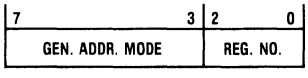
The Series 32000 Family CPUs generally access an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the Series 32000 family are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode within the instruction which acts upon that variable. Extraneous data movement is therefore minimized.

Series 32000 Addressing Modes fall into nine basic types:

Register: In floating-point instructions, these addressing modes refer to a Floating-Point Register (F0–F7) if the operand is of a floating-point type. Otherwise, a CPU General Purpose Register (R0–R7) is referenced. See Section 2.1.1.

Register Relative: A CPU General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.



TL/EE/5234–7

FIGURE 2-4. Index Byte Format

Memory Space: Identical to Register Relative above, except that the register used is one of the dedicated CPU registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

Memory Relative: A pointer variable is found within the memory space pointed to by the CPU SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written. Floating-point operands as well as integer operands may be specified using Immediate mode.

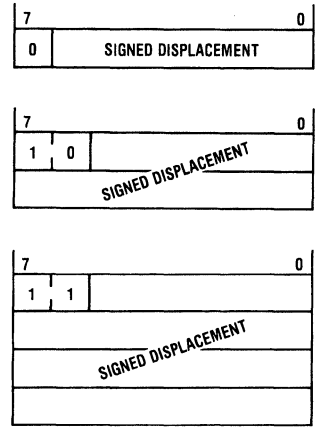
Absolute: The address of the operand is specified by a Displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected CPU Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

The following table, Table 2-1, is a brief summary of the addressing modes. For a complete description of their actions, see the Series 32000 Instruction Set Reference Manual.



TL/EE/5234–10

FIGURE 2-5. Displacement Encodings

2.0 Architectural Description (Continued)

TABLE 2-1. Series 32000 Family Addressing Modes

Encoding	Mode	Assembler Syntax	Effective Address
REGISTER			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
REGISTER RELATIVE			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
MEMORY SPACE			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	*+disp	
MEMORY RELATIVE			
10000	Frame memory relative	disp2(disp1(FP))	Disp2+Pointer; Pointer found at address Disp1+Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
IMMEDIATE			
10100	Immediate	value	None: Operand is issued from CPU instruction queue.
ABSOLUTE			
10101	Absolute	@disp	Disp.
EXTERNAL			
10110	External	EXT (disp1)+disp2	Disp2+Pointer; Pointer is found at Link Table Entry number Disp1.
TOP OF STACK			
10111	Top of Stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
SCALED INDEX			
11100	Index, bytes	mode[Rn:B]	Mode + Rn.
11101	Index, words	mode[Rn:W]	Mode + 2 × Rn.
11110	Index, double words	mode[Rn:D]	Mode + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	Mode + 8 × Rn. "Mode" and "n" are contained within the Index Byte.
10011	(Reserved for Future Use)		

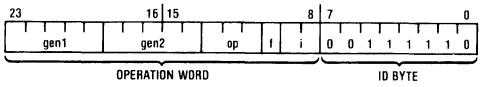
2.0 Architectural Description (Continued)

2.2.3 Floating-Point Instruction Set

The NS32081 FPU instructions occupy formats 9 and 11 of the Series 32000 Family instruction set (*Figure 2-6*). A list of all Series 32000 family instruction formats is found in the applicable CPU data sheet.

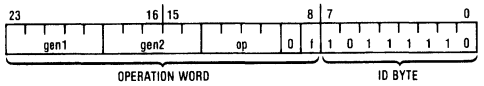
Certain notations in the following instruction description tables serve to relate the assembly language form of each instruction to its binary format in *Figure 2-6*.

Format 9



TL/EE/5234–11

Format 11



TL/EE/5234–12

FIGURE 2-6. Floating-Point Instruction Formats

The Format column indicates which of the two formats in *Figure 2-6* represents each instruction.

The Op column indicates the binary pattern for the field called "op" in the applicable format.

The Instruction column gives the form of each instruction as it appears in assembly language. The form consists of an instruction mnemonic in upper case, with one or more suffixes (i or f) indicating data types, followed by a list of operands (gen1, gen2).

An i suffix on an instruction mnemonic indicates a choice of integer data types. This choice affects the binary pattern in the i field of the corresponding instruction format (*Figure 2-6*) as follows:

Suffix i	Data Type	i Field
B	Byte	00
W	Word	01
D	Double Word	11

An f suffix on an instruction mnemonic indicates a choice of floating-point data types. This choice affects the setting of the f bit of the corresponding instruction format (*Figure 2-6*) as follows:

Suffix f	Data Type	f Bit
F	Single Precision	1
L	Double Precision (Long)	0

An operand designation (gen1, gen2) indicates a choice of addressing mode expressions. This choice affects the binary pattern in the corresponding gen1 or gen2 field of the instruction format (*Figure 2-6*). Refer to Table 2-1 for the options available and their patterns.

Further details of the exact operations performed by each instruction are found in the Series 32000 Instruction Set Reference Manual.

Movement and Conversion

The following instructions move the gen1 operand to the gen2 operand, leaving the gen1 operand intact.

Format	Op	Instruction		Description
11	0001	MOVf	gen1, gen2	Move without conversion
9	010	MOVLF	gen1, gen2	Move, converting from double precision to single precision.
9	011	MOVFL	gen1, gen2	Move, converting from single precision to double precision.
9	000	MOVif	gen1, gen2	Move, converting from any integer type to any floating-point type.
9	100	ROUNDfi	gen1, gen2	Move, converting from floating-point to the nearest integer.
9	101	TRUNCfi	gen1, gen2	Move, converting from floating-point to the nearest integer closer to zero.
9	111	FLOORfi	gen1, gen2	Move, converting from floating-point to the largest integer less than or equal to its value.

Note: The MOVLF instruction f bit must be 1 and the i field must be 10.
The MOVFL instruction f bit must be 0 and the i field must be 11.

Arithmetic Operations

The following instructions perform floating-point arithmetic operations on the gen1 and gen2 operands, leaving the result in the gen2 operand.

Format	Op	Instruction		Description
11	0000	ADDf	gen1, gen2	Add gen1 to gen2.
11	0100	SUBf	gen1, gen2	Subtract gen1 from gen2.
11	1100	MULf	gen1, gen2	Multiply gen2 by gen1.
11	1000	DIVf	gen1, gen2	Divide gen2 by gen1.
11	0101	NEGf	gen1, gen2	Move negative of gen1 to gen2.
11	1101	ABSf	gen1, gen2	Move absolute value of gen1 to gen2.

2.0 Architectural Description (Continued)

Comparison

The Compare instruction compares two floating-point values, sending the result to the CPU PSR Z and N bits for use as condition codes. See *Figure 3-6*. The Z bit is set if the gen1 and gen2 operands are equal; it is cleared otherwise. The N bit is set if the gen1 operand is greater than the gen2 operand; it is cleared otherwise. The CPU PSR L bit is unconditionally cleared. Positive and negative zero are considered equal.

Format	Op	Instruction		Description
11	0010	CMPf	gen1, gen2	Compare gen1 to gen2.

Floating-Point Status Register Access

The following instructions load and store the FSR as a 32-bit integer.

Format	Op	Instruction		Description
9	001	LFSR	gen1	Load FSR
9	110	SFSR	gen2	Store FSR

2.3 TRAPS

Upon detecting an exceptional condition in executing a floating-point instruction, the NS32081 FPU requests a trap by setting the Q bit of the status word transferred during the slave protocol (Section 3.5). The CPU responds by performing a trap using a default vector value of 3. See the Series 32000 Instruction Set Reference Manual and the applicable CPU data sheet for trap service details.

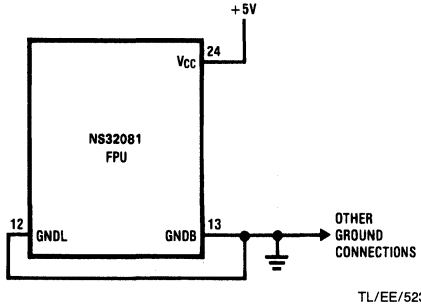
A trapped floating-point instruction returns no result, and does not affect the CPU Processor Status Register (PSR). The FPU displays the reason for the trap in the Trap Type (TT) field of the FSR (Section 2.1.2.2).

3.0 Functional Description

3.1 POWER AND GROUNDING

The NS32081 requires a single 5V power supply, applied on pin 24 (V_{CC}). See DC Electrical Characteristics table.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 12) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 13) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (*Figure 3-1*).



TL/EE/5234–13

FIGURE 3-1. Recommended Supply Connections

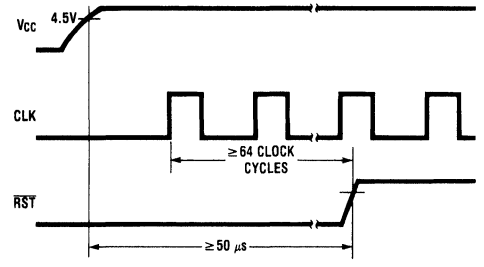
3.2 CLOCKING

The NS32081 FPU requires a single-phase TTL clock input on its CLK pin (pin 14). When the FPU is connected to a Series 32000 CPU, the CLK signal is provided from the CTTL pin of the NS32201 Timing Control Unit.

3.3 RESETTING

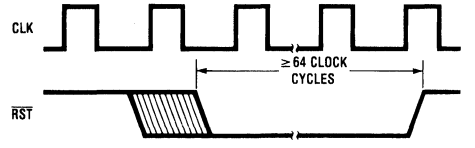
The $\overline{RST}$ pin serves as a reset for on-chip logic. The FPU may be reset at any time by pulling the $\overline{RST}$ pin low for at least 64 clock cycles. Upon detecting a reset, the FPU terminates instruction processing, resets its internal logic, and clears the FSR to all zeroes.

On application of power, $\overline{RST}$ must be held low for at least 50 μs after V_{CC} is stable. This ensures that all on-chip voltages are completely stable before operation. See *Figures 3-2* and *3-3*.



TL/EE/5234–14

FIGURE 3-2. Power-On Reset Requirements

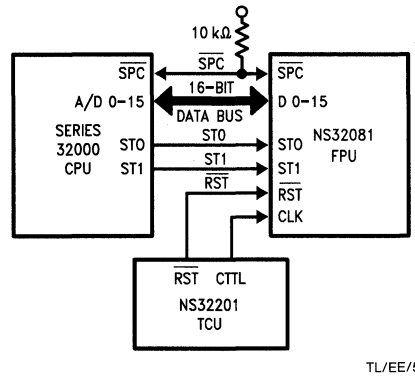


TL/EE/5234–15

FIGURE 3-3. General Reset Timing

3.4 BUS OPERATION

Instructions and operands are passed to the NS32081 FPU with slave processor bus cycles. Each bus cycle transfers either one byte (8 bits) or one word (16 bits) to or from the FPU. During all bus cycles, the $\overline{SPC}$ line is driven by the CPU as an active low data strobe, and the FPU monitors



TL/EE/5234–2

FIGURE 3-4. System Connection Diagram

3.0 Functional Description (Continued)

pins ST0 and ST1 to keep track of the sequence (protocol) established for the instruction being executed. This is necessary in a virtual memory environment, allowing the FPU to retry an aborted instruction.

3.4.1 Bus Cycles

A bus cycle is initiated by the CPU, which asserts the proper status on ST0 and ST1 and pulses $\overline{SPC}$ low. ST0 and ST1 are sampled by the FPU on the leading (falling) edge of the $\overline{SPC}$ pulse. If the transfer is from the FPU (a slave processor read cycle), the FPU asserts data on the data bus for the duration of the $\overline{SPC}$ pulse. If the transfer is to the FPU (a slave processor write cycle), the FPU latches data from the data bus on the trailing (rising) edge of the $\overline{SPC}$ pulse. *Figures 3-5* and *3-6* illustrate these sequences.

The direction of the transfer and the role of the bidirectional $\overline{SPC}$ line are determined by the instruction protocol being performed. $\overline{SPC}$ is always driven by the CPU during slave processor bus cycles. Protocol sequences for each instruction are given in Section 3.5.

3.4.2 Operand Transfer Sequences

An operand is transferred in one or more bus cycles. A 1-byte operand is transferred on the least significant byte of the data bus (D0–D7). A 2-byte operand is transferred on the entire bus. A 4-byte or 8-byte operand is transferred in consecutive bus cycles, least significant word first.

3.5 INSTRUCTION PROTOCOLS

3.5.1 General Protocol Sequence

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID byte followed by an Operation Word. See *Figure 2-7* for FPU instruction encodings. The ID Byte has three functions:

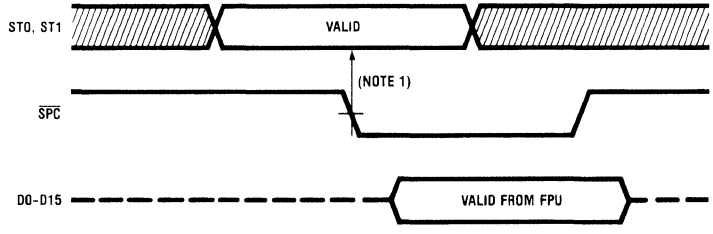
1) It identifies the instruction to the CPU as being a Slave Processor instruction.

2) It specifies which Slave Processor will execute it.

3) It determines the format of the following Operation Word of the instruction.

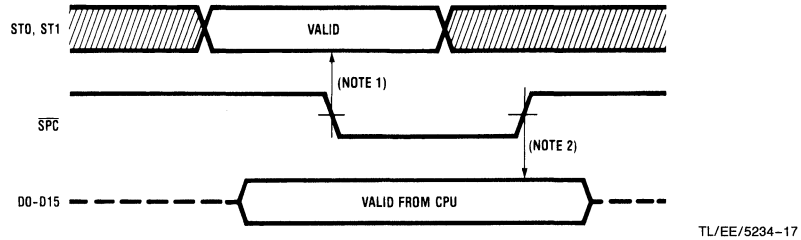
Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in Table 3-2. While applying Status Code 11 (Broadcast ID. Table 3-1), the CPU transfers the ID Byte on the least significant half of the Data Bus (D0–D7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 01 (Transfer Slave Operand, Table 3-1). Upon receiving it, the FPU decodes it, and at this point both the CPU and the FPU are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0–7 appear on pins D8–D15, and bits 8–15 appear on pins D0–D7.



Note 1: FPU samples CPU status here.

FIGURE 3-5. Slave Processor Read Cycle



Note 1: FPU samples CPU status here.

Note 2: FPU samples data bus here.

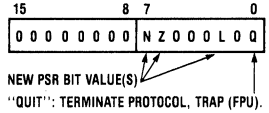
FIGURE 3-6. Slave Processor Write Cycle

3.0 Functional Description (Continued)

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the FPU. To do so, it references any Addressing Mode extensions appended to the FPU instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 01 (Transfer Slave Processor Operand, Table 3-1).

After the CPU has issued the last operand, the FPU starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SPC}$ low. To allow for this, the CPU releases the $\overline{SPC}$ signal, causing it to float. $\overline{SPC}$ must be held high by an external pull-up resistor.

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the FPU, applying Status Code 10. This word has the format shown in *Figure 3-7*. If the Q bit ("Quit", Bit 0) is set, this indicates that an error has been detected by the FPU. The CPU will not continue the protocol, but will immediately trap through the Slave vector in the Interrupt Table. If the instruction being performed is CMPf (Section 2.3.3) and the Q bit is not set, the CPU loads Processor Status Register (PSR) bits N, Z and L from the corresponding bits in the Status Word. The NS32081 FPU always sets the L bit to zero.



TL/EE/5234–18

FIGURE 3-7. FPU Protocol Status Word Format

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the FPU are performed by the CPU while applying Status Code 01 (Section 4.1.2).

TABLE 3-1. General Instruction Protocol

Step	Status	Action
1	11	CPU sends ID Byte.
2	01	CPU sends Operation Word.
3	01	CPU sends required operands.
4	XX	FPU starts execution.
5	XX	FPU pulses $\overline{SPC}$ low.
6	10	CPU reads Status Word.
7	01	CPU reads result (if any).

3.5.2 Floating-Point Protocols

Table 3-2 gives the protocols followed for each floating-point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Section 2.3.3.

The Operand Class columns give the Access Classes for each general operand, defining how the addressing modes are interpreted by the CPU (see Series 32000 Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating-Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a floating-point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-6*).

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified, because the Floating-Point Registers are physically on the Floating-Point Unit and are therefore available without CPU assistance.

TABLE 3-2. Floating Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

D = Double Word

i = Integer size (B, W, D) specified in mnemonic.

f = Floating-Point type (F, L) specified in mnemonic.

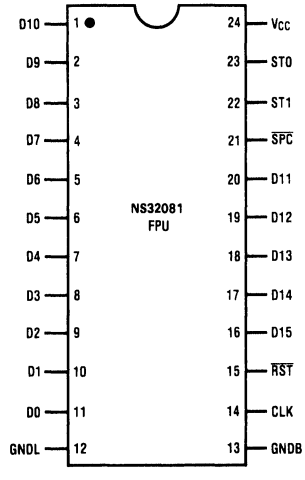
N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 PIN DESCRIPTIONS

The following are brief descriptions of all NS32081 FPU pins. The descriptions reference the relevant portions of the Functional Description, Section 3.

Dual-In-Line Package



TL/EE/5234–3

Top View

FIGURE 4-1. Connection Diagram

Order Number NS32081D-6, NS32081D-8 or NS32081D-10
See NS Package Number D24C

Order Number NS32081N-6, NS32081N-8 or NS32081N-10
See NS Package Number N24A

4.1.1 Supplies

Power (V_{CC}): +5V positive supply. Section 3.1.

Logic Ground (GNDL): Ground reference for on-chip logic. Section 3.1.

Buffer Ground (GNDB): Ground reference for on-chip drivers connected to output pins. Section 3.1.

4.1.2 Input Signals

Clock (CLK): TTL-level clock signal.

Reset ($\overline{RST}$): Active low. Initiates a Reset, Section 3.3.

Status (ST0, ST1): Input from CPU. ST0 is the least significant bit. Section 3.4 encodings are:

00—(Reserved)

01—Transferring Operation Word or Operand

10—Reading Status Word

11—Broadcasting Slave ID

4.1.3 Input/Output Signals

Slave Processor Control ($\overline{SPC}$): Active low. Driven by the CPU as the data strobe for bus transfers to and from the NS32081 FPU, Section 3.4. Driven by the FPU to signal completion of an operation, Section 3.5.1. Must be held high with an external pull-up resistor while floating.

Data Bus (D0–D15): 16-bit bus for data transfer. D0 is the least significant bit. Section 3.4.

4.2 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7.0V
Power Dissipation	1.5W

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		V_{CC} +0.5	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{OH}	Logical 1 Output Voltage	I_{OH} = −400 μA	2.4			V
V_{OL}	Logical 0 Output Voltage	I_{OL} = 2 mA			0.45	V
I_I	Input Load Current	0 ≤ V_{IN} ≤ V_{CC}	−10.0		10.0	μA
$I_{O(OFF)}$	Output Leakage Current (Output Pins in TRI-STATE Condition)	0.45 ≤ V_{IN} ≤ 2.4V	−20.0		20.0	μA
I_{CC}	Active Supply Current	I_{OUT} = 0, T_A = 0°C			300	mA

4.0 Device Specifications (Continued)

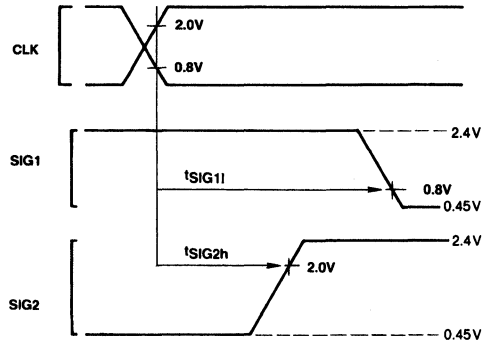
4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the Timing Specifications given in this section refer to 0.8V and 2.0V on all the input and output signals as illustrated in *Figures 4.2* and *4.3,* unless specifically stated otherwise.

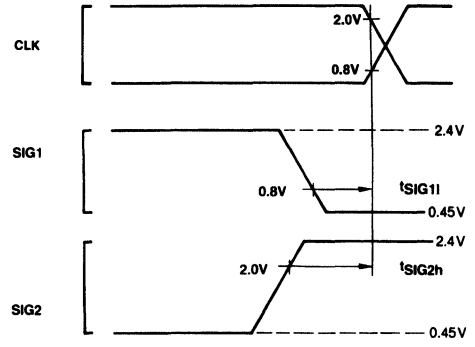
ABBREVIATIONS

L.E. — Leading Edge
T.E. — Trailing Edge
R.E. — Rising Edge
F.E. — Falling Edge



TL/EE/5234–26

FIGURE 4-2. Timing Specification Standard (Signal Valid After Clock Edge)



TL/EE/5234–27

FIGURE 4-3. Timing Specification Standard (Signal Valid Before Clock Edge)

4.0 Device Specifications (Continued)

4.4.2 Timing Tables

4.4.2.1 Output Signal Propagation Delays

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/ Conditions	NS32081-6		NS32081-8		NS32081-10		Units
				Min	Max	Min	Max	Min	Max	
t_{Dv}	4-7	Data Valid	After $\overline{SPC}$ L.E.		65		55		45	ns
t_{Df}	4-7	D_0–D_{15} Floating	After $\overline{SPC}$ T.E.		50		50		50	ns
t_{SPCFw}	4-9	$\overline{SPC}$ Pulse Width from FPU	At 0.8V (Both Edges)	$t_{CLKp} - 50$	$t_{CLKp} + 50$	$t_{CLKp} - 50$	$t_{CLKp} + 50$	$t_{CLKp} - 50$	$t_{CLKp} + 50$	ns
t_{SPCFl}	4-9	$\overline{SPC}$ Output Active	After CLK R.E.		90		70		55	ns
t_{SPCFh}	4-9	$\overline{SPC}$ Output Inactive	After CLK R.E.		90		70		55	ns
t_{SPCFnf}	4-9	$\overline{SPC}$ Output Nonforcing	After CLK F.E.		75		55		45	ns

4.4.2.2 Input Signal Requirements

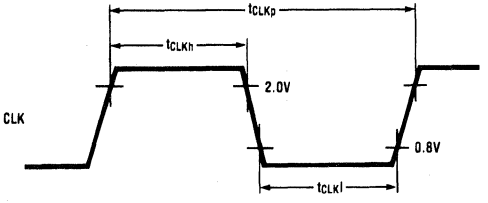
Name	Figure	Description	Reference/ Conditions	Min	Max	Min	Max	Min	Max	Units
t_{PWR}	4-5	Power Stable to $\overline{RST}$ R.E.	After V_{CC} Reaches 4.5V	50		50		50		μs
t_{RSTw}	4-6	$\overline{RST}$ Pulse Width	At 0.8V (Both Edges)	64		64		64		t_{CLKp}
t_{Ss}	4-7	Status (ST0–ST1) Setup	Before $\overline{SPC}$ L.E.	75		60		50		ns
t_{Sh}	4-7	Status (ST0–ST1) Hold	After $\overline{SPC}$ L.E.	90		70		40		ns
t_{Ds}	4-8	D0–D15 Setup Time	Before $\overline{SPC}$ T.E.	75		55		40		ns
t_{Dh}	4-8	D0–D15 Hold Time	After $\overline{SPC}$ T.E.	80		65		50		ns
t_{SPCw}	4-7	SPC Pulse Width from CPU	At 0.8V (Both Edges)	100		85		70		ns
t_{SPCs}	4-7	$\overline{SPC}$ Input Active	Before CLK R.E.	50		45		40		ns
t_{SPCh}	4-7	$\overline{SPC}$ Input Inactive	After CLK R.E.	0		0		0		ns
t_{RSTs}	4-10	$\overline{RST}$ Setup	Before CLK F.E.	10		10		10		ns
t_{RSTh}	4-10	$\overline{RST}$ R.E. Delay	After CLK R.E.	0		0		0		ns

4.4.2.3 Clocking Requirements

Name	Figure	Description	Reference/ Conditions	Min	Max	Min	Max	Min	Max	Units
t_{CLKh}	4-4	Clock High Time	At 2.0V (Both Edges)	60	1000	50	1000	42	1000	ns
t_{CLKl}	4-4	Clock Low Time	At 0.8V (Both Edges)	60	1000	50	1000	42	1000	ns
t_{CLKp}	4-4	Clock Period	CLK R.E. to Next CLK R.E.	160	2000	125	2000	100	2000	ns

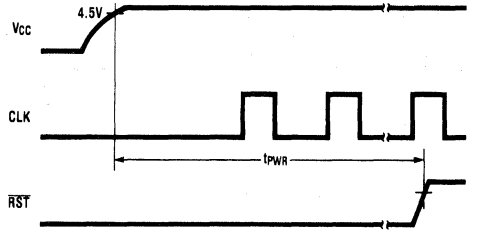
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



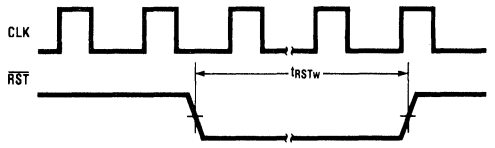
TL/EE/5234-19

FIGURE 4-4. Clock Timing



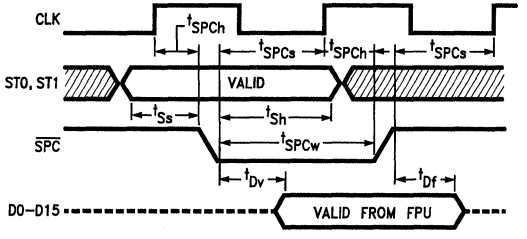
TL/EE/5234-20

FIGURE 4-5. Power-On Reset



TL/EE/5234-21

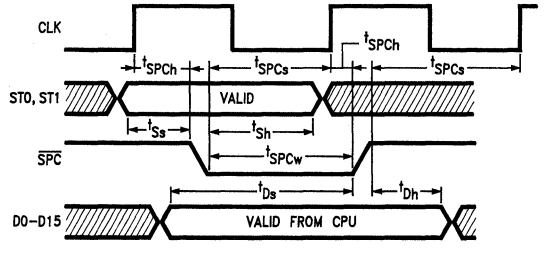
FIGURE 4-6. Non-Power-On Reset



TL/EE/5234-22

FIGURE 4-7. Read Cycle from FPU

Note: $\overline{SPC}$ pulse must be (nominally) 1 clock wide when writing into FPU.

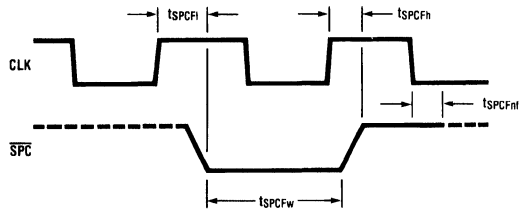


TL/EE/5234-23

FIGURE 4-8. Write Cycle to FPU

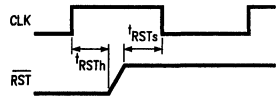
Note: $\overline{SPC}$ pulse may also be 2 clocks wide, but its edges must meet the t_{SPCs} and t_{SPCh} requirements with respect to CLK.

4.0 Device Specifications (Continued)



TL/EE/5234–24

FIGURE 4-9. $\overline{SPC}$ Pulse from FPU



TL/EE/5234–25

FIGURE 4-10. $\overline{RST}$ Release Timing

Note: The rising edge of $\overline{RST}$ must occur while CLK is high, as shown.

National Semiconductor Corporation

NS32082-6/NS32082-8/NS32082-10 Memory Management Unit (MMU)

General Description

The NS32082 Memory Management Unit (MMU) provides hardware support for demand-paged virtual memory implementations. The NS32082 functions as a slave processor in Series 32000 microprocessor-based systems. Its specific capabilities include fast dynamic translation, protection, and detailed status to assist an operating system in efficiently managing up to 32 Mbytes of physical memory. Support for multiple address spaces, virtual machines, and program debugging is provided.

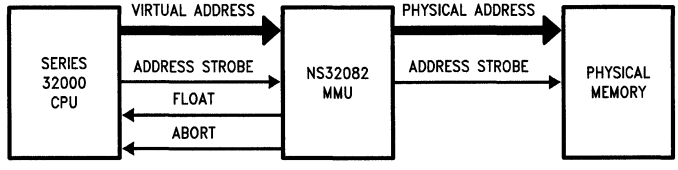
High-speed address translation is performed on-chip through a 32-entry fully associative translation look-aside buffer (TLB), which maintains itself from tables in memory with no software intervention. Protection violations and page faults (references to non-resident pages) are automatically detected by the MMU, which invokes the instruction abort feature of the CPU.

Additional features for program debugging include two breakpoint registers and a breakpoint counter, which provide the programmer with powerful stand-alone debugging capability.

Features

- Totally automatic mapping of 16 Mbyte virtual address space using memory based tables
- On-chip translation look-aside buffer allows 97% of translations to occur in one clock for most applications
- Full hardware support for virtual memory and virtual machines
- Implements "referenced" bits for simple, efficient working set management
- Protection mechanisms implemented via access level checking and dual space mapping
- Program debugging support
- Compatible with NS32016, NS32032 and NS32332 CPUs
- 48-pin dual-in-line package

Conceptual Address Translation Model



TL/EE/8692–1

Table Of Contents

1.0 PRODUCT INTRODUCTION
1.1 Programming Considerations
2.0 FUNCTIONAL DESCRIPTION
2.1 Power and Grounding
2.2 Clocking
2.3 Resetting
2.4 Bus Operation
2.4.1 Interconnections
2.4.2 CPU-Initiating Cycles
2.4.3 MMU-Initiated Cycles
2.4.4 Cycle Extension
2.5 Slave Processor Interface
2.5.1 Slave Processor Bus Cycles
2.5.2 Instruction Protocols
2.6 Bus Access Control
2.7 Breakpointing
2.7.1 Breakpoints on Execution
3.0 ARCHITECTURAL DESCRIPTION
3.1 Programming Model
3.2 Memory Management Functions
3.2.1 Page Table Structure
3.2.2 Virtual Address Spaces
3.2.3 Page Table Entry Formats
3.2.4 Physical Address Generation
3.3 Page Table Base Registers (PTBO, PTBI)
3.4 Error/Invalidate Address Register (EIA)
3.0 ARCHITECTURAL DESCRIPTION (Continued)
3.5 Breakpoint Registers (BPR0, BPR1)
3.6 Breakpoint Count Register (BCNT)
3.7 Memory Management Status Register (MSR)
3.7.1 MSR Fields for Address Translation
3.7.2 MSR Fields for Debugging
3.8 Translation Lookaside Buffer (TLB)
3.9 Entry/Re-entry into Programs Under Debugging
3.10 Address Translation Algorithm
3.11 Instruction Set
4.0 DEVICE SPECIFICATIONS
4.1 Pin Descriptions
4.1.1 Supplies
4.1.2 Input Signals
4.1.3 Output Signals
4.1.4 Input-Output Signals
4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics
4.4.1 Definitions
4.4.2 Timing Tables
4.4.2.1 Output Signals; Internal Propagation Delays
4.4.2.2 Input Signal Requirements
4.4.2.3 Clocking Requirements
Appendix A: Interfacing Suggestions

List of Illustrations

The Virtual Memory Model 1-1
NS32082 Address Translation Model 1-2
Recommended Supply Connections 2-1
Clock Timing Relationships 2-2
Power-On Reset Requirements 2-3
General Reset Timing 2-4
Recommended Reset Connections, Memory Managed System 2-5
CPU Read Cycle; Translation in TLB 2-6
Abort Resulting from Protection Violation; Translation in TLB 2-7
Page Table Lookup 2-8
Abort Resulting After a Page Table Lookup 2-9
Slave Access Timing; CPU Reading from MMU 2-10
Slave Access Timing; CPU Writing to MMU 2-11
$\overline{FLT}$ Deassertation During RDVAL/WRVAL Execution 2-12
Bus Timing with Breakpoint on Physical Address Enabled 2-13
Execution Breakpoint Timing; Insertion of DIA Instruction 2-14
Two-Level Page Tables 3-1
A Page Table Entry 3-2
Virtual to Physical Address Translation 3-3
Page Table Base Registers (PTB0, PTB1) 3-4
EIA Register 3-5
Breakpoint Registers (BPR0, BPR1) 3-6
Breakpoint Counter Register (BCNT) 3-7
Memory Managment Status Register (MSR) 3-8

List of Illustrations (Continued)

TLB Model 3-9
Slave Instruction Format 3-10
Dual-In-Line Package 4-1
Timing Specification Standard (Signal Valid After Clock Edge) 4-2
Timing Specification Standard (Signal Valid Before Clock Edge) 4-3
CPU Read (Write) Cycle Timing (32-Bit Mode) 4-4
MMU Read Cycle Timing (32-Bit Mode) after a TLB Miss 4-5
MMU Write Cycle Timing After a TLB Miss 4-6
$\overline{FLT}$ Deassertation Timing 4-7
Abort Timing ($\overline{FLT}$ = 1) 4-8
Abort Timing ($\overline{FLT}$ = 0) 4-9
CPU Operand Access Cycle with Breakpoint On Physical Address Enabled 4-10
Slave Access Timing; CPU Reading from MMU 4-11
Slave Access Timing; CPU Writing to MMU 4-12
$\overline{SPC}$ Pulse From the MMU 4-13
$\overline{HOLD}$ Timing ($\overline{FLT}$ = 1); SMR Instruction Not Being Executed 4-14
$\overline{HOLD}$ Timing ($\overline{FLT}$ = 1); SMR Instruction Being Executed 4-15
$\overline{HOLD}$ Timing ($\overline{FLT}$ = 0) 4-16
Clock Waveforms 4-17
Reset Timing 4-18
Power-On Reset 4-19
System Connection Diagram A-1
System Connection Diagram A-2

Tables

ST0–ST3 Encodings 2-1
LMR Instruction Protocol 2-2
SMR Instruction Protocol 2-3
RDVAL/WRVAL Instruction Protocol 2-4
Access Protection Levels 3-1
Instructions Causing Non-Sequential Fetches 3-2
"Short" Field Encodings 3-3

1.0 Product Introduction

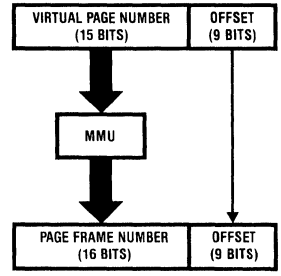
The NS32082 MMU provides hardware support for three basic features of the Series 32000; dynamic address translation, access level checking and software debugging. Dynamic Address Translation is required to implement demand-paged virtual memory. Access level checking is performed during address translation, ensuring that unauthorized accesses do not occur. Because the MMU resides on the local bus and is in an ideal location to monitor CPU activity, debugging functions are also included.

The MMU is intended for use in implementing demand-paged virtual memory. The concept of demand-paged virtual memory is illustrated in *Figure 1-1.* At any point in time, a program sees a uniform addressing space of up to 16 megabytes (the "virtual" space), regardless of the actual size of the memory physically present in the system (the "physical" space). The full virtual space is recorded as an image on a mass storage device. Portions of the virtual space needed by a running program are copied into physical memory when needed.

To make the virtual information directly available to a running program, a mapping must be established between the virtual addresses asserted by the CPU and the physical addresses of the data being referenced.

To perform this mapping, the MMU divides the virtual memory space into 512-byte blocks called "pages." It interprets the 24-bit address from the CPU as a 15-bit "page number" followed by a 9-bit offset, which indicates the position of a byte within the selected page. Similarly, the MMU divides the physical memory into 512-byte frames, each of which can hold a virtual page.

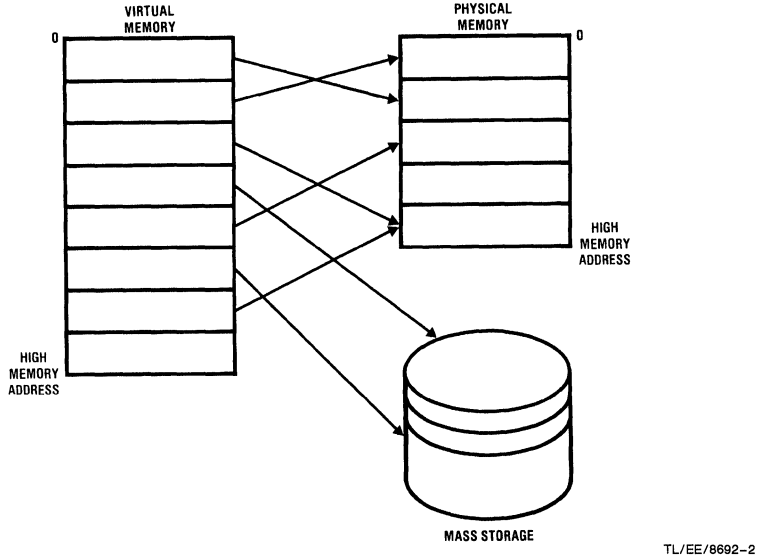
The translation process is therefore modeled as accepting a virtual page number from the CPU and substituting the corresponding physical page frame number for it, as shown in *Figure 1-2.* The offset is not changed. The translated page frame number is 16 bits long, including an additional address bit (A24) intended for physical bank selection. Physical addresses issued by the MMU are 25 bits wide.



TL/EE/8692-3

FIGURE 1-2. NS32082 Address Translation Model

Generally, in virtual memory systems the available physical memory space is smaller than the maximum virtual memory space. Therefore, not all virtual pages are simultaneously resident. Nonresident pages are not directly addressable by the CPU. Whenever the CPU issues a virtual address for a nonresident or nonexistent page, a "page fault" will result. The MMU signals this condition by invoking the Abort feature of the CPU. The CPU then halts the memory cycle,



TL/EE/8692-2

FIGURE 1-1. The Virtual Memory Model

1.0 Product Introduction (Continued)

restores its internal state to the point prior to the instruction being executed, and enters the operating system through the abort trap vector.

The operating system reads from the MMU the virtual address which caused the abort. It selects a page frame which is either vacant or not recently used and, if necessary, writes this frame back to mass storage. The required virtual page is then copied into the selected page frame.

The MMU is informed of this change by updating the page tables (Section 3.2), and the operating system returns control to the aborted program using the RETT instruction. Since the return address supplied by the abort trap is the address of the aborted instruction, execution resumes by retrying the instruction.

This sequence is called paging. Since a page fault encountered in normal execution serves as a demand for a given page, the whole scheme is called demand-paged virtual memory.

The MMU also provides debugging support. It may be programmed to monitor the bus for two virtual or physical addresses in real time. A counter register is associated with one of these, providing a "break-on-N-occurrences" capability.

1.1 PROGRAMMING CONSIDERATIONS

When a CPU instruction is aborted as a result of a page fault, some memory resident data might have been already modified by the instruction before the occurrence of the abort.

This could compromise the restartability of the instruction when the CPU returns from the abort routine.

To guarantee correct results following the re-execution of the aborted instruction, the following actions should not be attempted:

a) No instruction should try to overlay part of a source operand with part of the result. It is, however, permissible to rewrite the result into the source operand exactly (for example, the instruction "ABSW X, X", which replaces X with its absolute value). Also, never write to any memory location which is necessary for calculating the effective address of either operand (i.e. the pointer in "Memory Relative" addressing mode; the Link Table pointer or Link Table Entry in "External" addressing mode).

b) No instruction should perform a conversion in place from one data type to another larger data type (Example: MOVWF X, X which replaces the 16-bit integer value in memory location X with its 32-bit floating-point value). The addressing mode combination "TOS, TOS" is an exception, and is allowed. This is because the least-significant part of the result is written to the possibly invalid page before the source operand is affected. Also, integer conversions to larger integers always work correctly in place, because the low-order portion of the result always matches the source value.

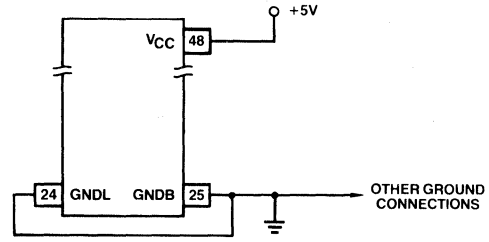
c) When performing the MOVM instruction, the entire source and destination blocks must be considered "operands" as above, and they must not overlap.

2.0 Functional Description

2.1 POWER AND GROUNDING

The NS32082 requires a single 5V power supply, applied on pin 48 (V_{CC}).

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (*Figure 2-1*).



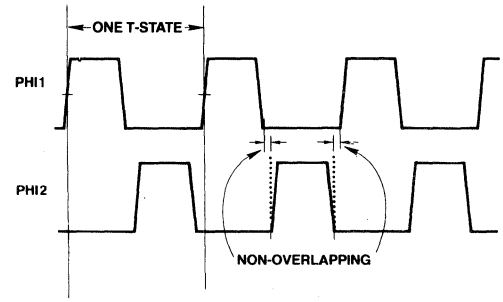
TL/EE/8692–4

FIGURE 2-1. Recommended Supply Connections

2.2 CLOCKING

The NS32082 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in *Figure 2-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the MMU. One T-State represents one hardware cycle within the MMU, and/or one step of an external bus transfer. See Section 4 for complete specifications of PHI1 and PHI2.



TL/EE/8692–5

FIGURE 2-2. Clock Timing Relationships

As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected to any devices other than the CPU and MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

2.0 Functional Description (Continued)

2.3 RESETTING

The $\overline{RSTI}$ input pin is used to reset the NS32082. The MMU responds to $\overline{RSTI}$ by terminating processing, resetting its internal logic and clearing the appropriate bits in the MSR register.

Only the MSR register is changed on reset. No other program accessible registers, including the TLB are affected.

The $\overline{RST}/\overline{ABT}$ signal is activated by the MMU on reset. This signal should be used to reset the CPU. $\overline{AT}/\overline{SPC}$ is held low for five clock cycles after the rising edge of $\overline{RSTI}$ to indicate to the CPU that the address translation mode must be selected.

The A24/$\overline{HBF}$ signal is sampled by the MMU on the rising edge of $\overline{RSTI}$. It indicates the bus size of the attached CPU. A24/$\overline{HBF}$ must be sampled high for a 16-bit bus and low for a 32-bit bus.

On application of power, $\overline{RSTI}$ must be held low for at least 50 μs after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 2-3* and *2-4.*

The NS32201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32082 MMU. *Figure 2-5* shows the recommended connections.

2.4 BUS OPERATION

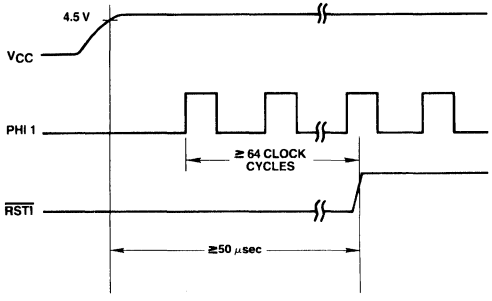
2.4.1 Interconnections

The MMU runs synchronously with the CPU, sharing with it a single multiplexed address/data bus. The interconnections used by the MMU for bus control, when used in conjunction with the NS32016, are shown in *Figure A-1* (Appendix A).

The CPU issues 24-bit virtual addresses on the bus, and status information on other pins, pulsing the signal $\overline{ADS}$ low. These are monitored by the MMU. The MMU issues 25-bit physical addresses on the bus, pulsing the $\overline{PAV}$ line low. The $\overline{PAV}$ pulse triggers the address latches and signals the NS32201 TCU to begin a bus cycle. The TCU in turn generates the necessary bus control signals and synchronizes the insertion of WAIT states, by providing the signal RDY to the MMU and CPU. Note that it is the MMU rather than the CPU that actually triggers bus activity in the system.

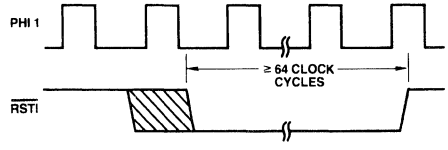
The functions of other interface signals used by the MMU to control bus activity are described below.

The ST0–ST3 pins indicate the type of cycle being initiated by the CPU. ST0 is the least-significant bit of the code. Table 2-1 shows the interpretations of the status codes presented on these lines.



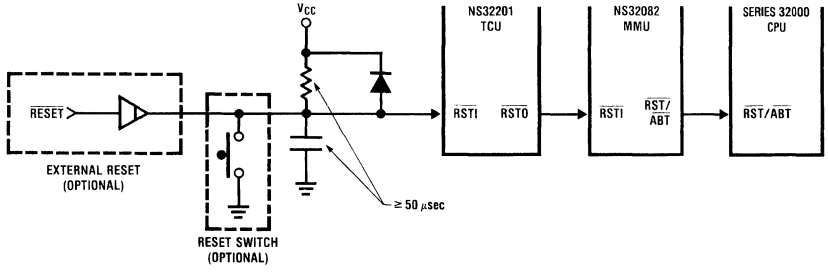
TL/EE/8692-6

FIGURE 2-3. Power-On Reset Requirements



TL/EE/8692-7

FIGURE 2-4. General Reset Timing



TL/EE/8692-8

FIGURE 2-5. Recommended Reset Connections, Memory-Managed System

2.0 Functional Description (Continued)

Status codes that are relevant to the MMU's function during a memory reference are:

1000, 1001 Instruction Fetch status, used by the debugging features to distinguish between data and instruction references.

1010 Data Transfer. A data value is to be transferred.

1011 Read RMW Operand. Although this is always a Read cycle, the MMU treats it as a Write cycle for purposes of protection and breakpointing.

All other status codes are treated as data accesses if they occur in conjunction with a pulse on the $\overline{ADS}$ pin. Note that these include Interrupt Acknowledge and End of Interrupt cycles performed by the CPU. The status codes 1101, 1110 and 1111 are also recognized by the MMU in conjunction with pulses on the $\overline{SPC}$ line while it is executing Slave Processor instructions, but these do not occur in a context relevant to address translation.

TABLE 2-1. ST0–ST3 Encodings (ST0 is the Least Significant)

0000 — Idle: CPU Inactive on Bus
0001 — Idle: WAIT Instruction
0010 — (Reserved)
0011 — Idle: Waiting for Slave
0100 — Interrupt Acknowledge, Master
0101 — Interrupt Acknowledge, Cascaded
0110 — End of Interrupt, Master
0111 — End of Interrupt, Cascaded
1000 — Sequential Instruction Fetch
1001 — Non-Sequential Instruction Fetch
1010 — Data Transfer
1011 — Read Read-Modify-Write Operand
1100 — Read for Effective Address
1101 — Transfer Slave Operand
1110 — Read Slave Status Word
1111 — Broadcast Slave ID

The $\overline{DDIN}$ line indicates the direction of the transfer: 0 = Read, 1 = Write.

$\overline{DDIN}$ is monitored by the MMU during CPU cycles to detect write operations, and is driven by the MMU during MMU-initiated bus cycles.

The U/$\overline{S}$ pin indicates the privilege level at which the CPU is making the access: 0 = Supervisor Mode, 1 = User Mode. It is used by the MMU to select the address space for translation and to perform protection level checking. Normally, the U/$\overline{S}$ pin is a direct reflection of the U bit in the CPU's Processor Status Register (PSR). The MOVUS and MOVSU CPU instructions, however, toggle this pin on successive operand accesses in order to move data between virtual spaces.

The MMU uses the $\overline{FLT}$ line to take control of the bus from the CPU. It does so as necessary for updating its internal TLB from the Page Tables in memory, for maintaining the contents of the status bits (R and M) in the Page Table Entries, and for implementing bus timing adjustments needed by the debugging features.

The MMU also aborts invalid accesses attempted by the CPU. This is done by pulsing the $\overline{RST}/\overline{ABT}$ pin low for one clock period. (A pulse longer than one clock period is interpreted by the CPU as a Reset command).

Because the MMU performs only 16-bit transfers, some additional circuitry is needed to interface it to the 32-bit data bus of an NS32032-based system. However, since the MMU never writes to the most-significant word of a Page Table Entry, the only special requirement is that it must be able to read from the top half of the bus. This can be accomplished as shown in *Figure A-2* (Appendix A) by using a 16-bit unidirectional buffer and some gating circuitry that enables it whenever an MMU-initiated bus cycle accesses an address ending in binary "10".

The bus connections required in conjunction with the NS32332 CPU are somewhat more complex (see the NS32332 data sheet), but the sequences of events documented here still hold.

2.4.2 CPU-Initiated Bus Cycles

A CPU-initiated bus cycle is performed in a minimum of five clock cycles (four in the case of the NS32332): T1, TMMU, T2, T3 and T4, as shown in *Figure 2-6*.

During period T1, the CPU places the virtual address to be translated on the bus, and the MMU latches it internally and begins translation. The MMU also samples the $\overline{DDIN}$ pin, the status lines ST0–ST3, and the U/$\overline{S}$ pin to determine how the CPU intends to use the bus.

During period TMMU the CPU floats it bus drivers and the MMU takes one of three actions:

1) If the translation for the virtual address is resident in the MMU's TLB, and the access being attempted by the CPU does not violate the protection level of the page being referenced, the MMU presents the translated address and generates a $\overline{PAV}$ pulse to trigger a bus cycle in the rest of the system. See *Figure 2-6*.
2) If the translation for the virtual address is resident in the MMU's TLB, but the access being attempted by the CPU is not allowed due to the protection level of the page being referenced, the MMU generates a pulse on the $\overline{RST}/\overline{ABT}$ pin to abort the CPU's access. No $\overline{PAV}$ pulse is generated. See *Figure 2-7*.
3) If the translation for the virtual address is not resident in the TLB, or if the CPU is writing to a page whose M bit is not yet set, the MMU takes control of the bus asserting the $\overline{FLT}$ signal as shown in *Figure 2-8*. This causes the CPU to float its bus and wait. The MMU then initiates a sequence of bus cycles as described in Section 2.4.3.

From state T2 through T4 data is transferred on the bus between the CPU and memory, and the TCU provides the strobes for the transfer. During this time the MMU floats

2.0 Functional Description (Continued)

pins AD0–AD15, and handles pins A16–A24 according to the mode of operation (16-bit or 32-bit) selected during reset (Section 2.3).

In 16-bit bus mode, the MMU drives address lines A16–A24 from TMMU through T4 and they need not be latched externally. This is appropriate for the NS32016 CPU, which uses only AD0–AD15 for data transfers. In 32-bit bus mode, the MMU asserts the physical address on pins A16–A24 only during TMMU, and floats them from T2 through T4 because the CPU uses them for data transfer. In this case the physical address presented on these lines must be latched externally using $\overline{PAV}$.

Whenever the MMU generates an Abort pulse on the $\overline{RST}/\overline{ABT}$ pin, the CPU enters state T2 and then Ti (idle), ending the bus cycle. Since no $\overline{PAV}$ pulse is issued by the MMU, the rest of the system remains unaware that an access has been attempted. The MMU requires that no further memory references be attempted by the CPU for at least two clock cycles after the T2 state, as shown in *Figure 2-7*. This requirement is met by all Series 32000 CPU's. During this time, the RDY line must remain high. This requirement is met by the NS32201 TCU.

2.4.3 MMU-Initiated Cycles

Bus cycles initiated by the MMU are always nested within CPU-initiated bus cycles; that is, they appear after the MMU has accepted a virtual address from the CPU and has set the $\overline{FLT}$ line active. The MMU will initiate memory cycles in the following cases:

1) There is no translation in the MMU's TLB for the virtual address issued by the CPU, meaning that the MMU must reference the Page Tables in memory to obtain the translation.
2) There is a translation for that virtual address in the TLB, but the page is being written for the first time (the M bit in its Level-2 Page Table Entry is 0). The MMU treats this case as if there were no translation in the TLB, and performs a Page Table lookup in order to set the M bit in the Level-2 Page Table Entry as well as in the TLB.

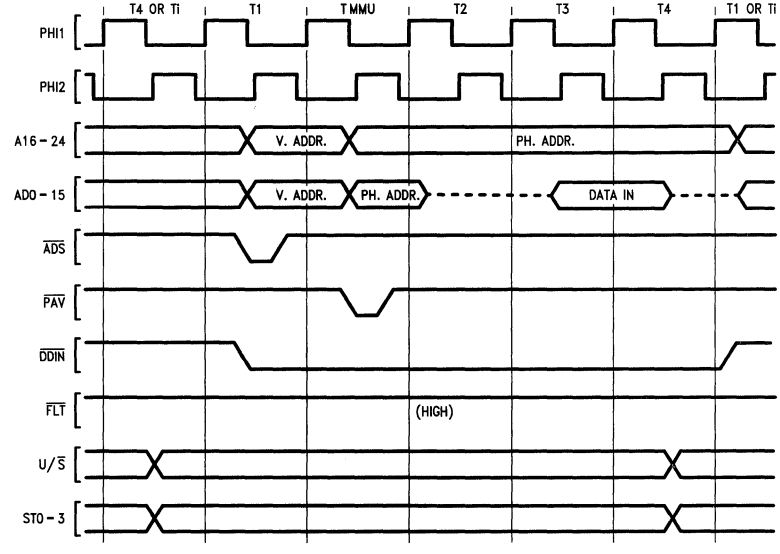
Having made the necessary memory references, the MMU either aborts the CPU access or it provides the translated address and allows the CPU's access to continue to T2.

Figure 2-8 shows the sequence of events in a Page Table lookup. After asserting $\overline{FLT}$, the MMU waits for one additional clock cycle, then reads the Level-1 Page Table Entry and the Level-2 Page Table Entry in four consecutive memory Read cycles. Note that the MMU performs two 16-bit transfers to read each Page Table Entry, regardless of the width of the CPU's data bus. There are no idle clock cycles between MMU-initiated bus cycles unless a bus request is made on the $\overline{HOLD}$ line (Section 2.6).

During the Page Table lookup the MMU drives the $\overline{DDIN}$ signal. The status lines ST0–ST3 and the U/$\overline{S}$ pin are not released by the CPU, and retain their original settings while the MMU uses the bus. The Byte Enable signals from the CPU ($\overline{HBE}$ in 16-bit systems, $\overline{BE0}$–$\overline{BE3}$ in 32-bit systems) should in general be handled externally for correct memory referencing. (The current NS32016 CPU does, however, handle $\overline{HBE}$ in a manner that is acceptable in many systems at clock rates of 12.5 MHz or less.)

In the clock cycle immediately after T4 of the last lookup cycle, the MMU removes the $\overline{FLT}$ signal, issues the translated address, and pulses $\overline{PAV}$ to continue the CPU's access.

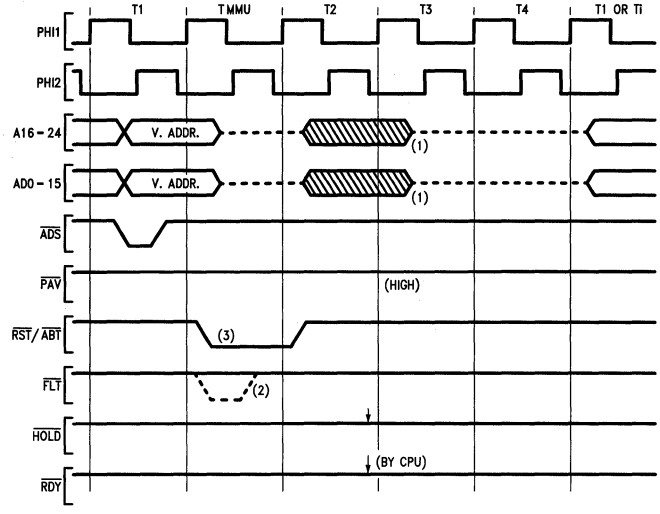
Note that when the MMU sets $\overline{FLT}$ active, the clock cycle originally called TMMU is redesignated Tf. Clock cycles in which the $\overline{PAV}$ pulse occurs are designated TMMU.



TL/EE/8692-9

FIGURE 2-6. CPU Read Cycle; Translation in TLB (TLB Hit)

2.0 Functional Description (Continued)



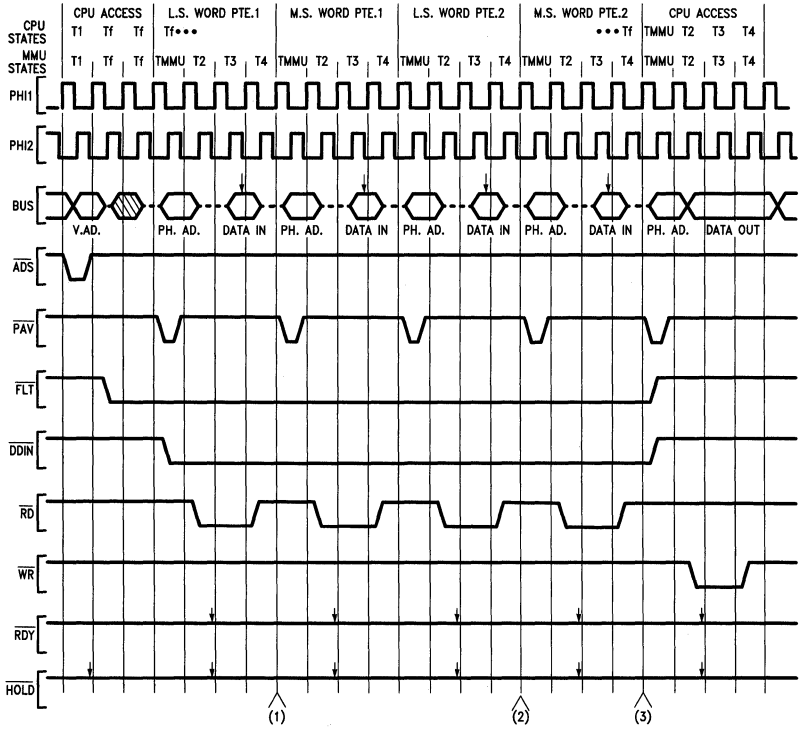
TL/EE/8692-10

Note 1: The CPU drives the bus if a write cycle is aborted.

Note 2: $\overline{FLT}$ may be pulsed if a breakpoint on physical address is enabled or an execution breakpoint is triggered.

Note 3: If this bus cycle is a write cycle to a write-protected page, $\overline{FLT}$ is asserted for two clock cycles and the abort pulse is delayed by one clock cycle.

FIGURE 2-7. Abort Resulting from Protection Violation; Translation in TLB



TL/EE/8692-11

Note 1: If the R bit on the Level-1 PTE must be set, a write cycle is inserted here.

Note 2: If either the R or the M bit on the Level-2 PTE must be set, a write cycle is inserted here.

Note 3: If a breakpoint on physical address is enabled, an extra clock cycle is inserted here.

FIGURE 2-8. Page Table Lookup

2.0 Functional Description (Continued)

The Page Table Entries are read starting with the low-order word. If the V bit (bit 0) of the low-order word is zero, or the protection level field PL (bits 1 and 2) indicates that the CPU's attempted access is illegal, the MMU does not generate any further memory cycles, but instead issues an Abort pulse during the clock cycle after T4 and removes the $\overline{FLT}$ signal. The CPU continues to T2 and then becomes idle on the bus, as shown in *Figure 2-9*.

If the R and/or M bit (bit 3 or 4) of the low-order word must be updated, the MMU does this immediately in a single Write cycle, before reading the high-order word of the Page Table Entry. All bits except those updated are rewritten with their original values.

At most, the MMU writes two 16-bit words to memory during a translation: the first to the Level-1 table to update the R bit, and the second to the Level-2 table to update the R and/or M bits.

2.4.4 Cycle Extension

To allow sufficient strobe widths and access time requirements for any speed of memory or peripheral device, the NS32082 provides for extension of a bus cycle. Any type of bus cycle, CPU-initiated or MMU-initiated, can be extended, except Slave Processor cycles, which are not memory or peripheral references.

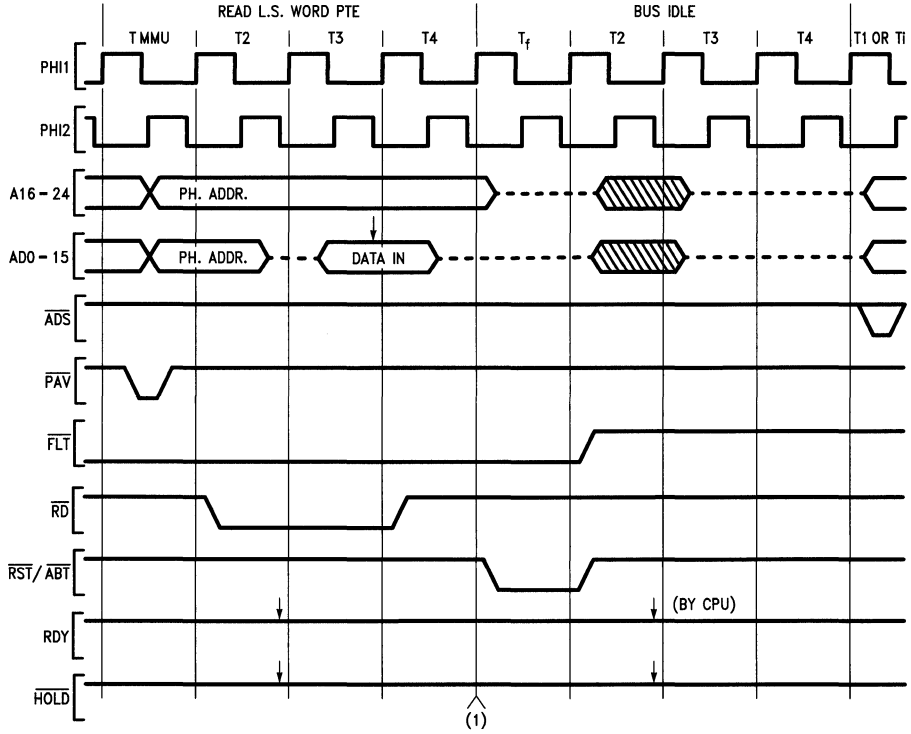
In *Figures 2-6* and *2-8*, note that during T3 all bus control signals are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

Immediately before T3 begins, on the falling edge of clock phase PHI2, the RDY line is sampled by the CPU and/or the MMU. If RDY is high, the next state after T3 will be T4, ending the bus cycle. If it is low, the next state after T3 will be another T3 and the RDY line will be sampled again. RDY is sampled in each following clock period, with insertion of additional T3 states, until it is sampled high. Each additional T3 state inserted is called a "WAIT state."

During CPU bus cycles, the MMU monitors the RDY pin only if the 16-bit mode is selected. This is necessary since the MMU drives the address lines A16–A24, and needs to detect the end of the bus cycle in order to float them.

If the 32-bit mode is selected, the above address lines are floated following the TMMU state. The MMU will be ready to perform another translation after three clock cycles, and the RDY line is ignored.

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT states to the CPU and MMU as requested on its own WAIT request input pins.



TL/EE/8692-12

Note 1: If a breakpoint on physical address is enabled, an extra clock cycle is inserted here.

FIGURE 2-9. Abort Resulting after a Page Table Lookup

2.0 Functional Description (Continued)

2.5 SLAVE PROCESSOR INTERFACE

The CPU and MMU execute four instructions cooperatively. These are LMR, SMR, RDVAL and WRVAL, as described in Section 2.5.2. The MMU takes the role of a Slave Processor in executing these instructions, accepting them as they are issued to it by the CPU. The CPU calculates all effective addresses and performs all operand transfers to and from memory and the MMU. The MMU does not take control of the bus except as necessary in normal operation; i.e., to translate and validate memory addresses as they are presented by the CPU.

The sequence of transfers ("protocol") followed by the CPU and MMU involves a special type of bus cycle performed by the CPU. This "Slave Processor" bus cycle does not involve the issuing of an address, but rather performs a fast data transfer whose purpose is pre-determined by the form of the instruction under execution and by status codes asserted by the CPU.

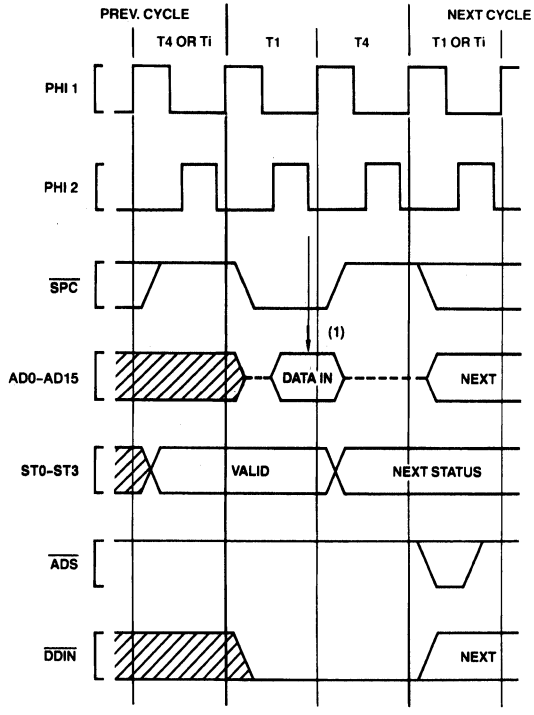
2.5.1 Slave Processor Bus Cycles

The interconnections between the CPU and MMU for Slave Processor communication are shown in *Figures A-1* and *A-2* (Appendix A). The low-order 16 bits of the bus are used for data transfers. The $\overline{SPC}$ signal is bidirectional. It is pulsed by the CPU as a low-active data strobe for Slave Processor transfers, and is also pulsed low by the MMU to acknowledge, when necessary, that it is ready to continue execution of an MMU instruction. Since $\overline{SPC}$ is normally in a high-impedance state, it must be pulled high with a 10 kΩ resistor, as shown. The MMU also monitors the status lines ST0–ST3 to follow the protocol for the instruction being executed.

Data is transferred between the CPU and the MMU with Slave Processor bus cycles, illustrated in *Figures 2-10* and *2-11*. Each bus cycle transfers one byte or one word (16 bits) to or from the MMU.

Slave Processor bus cycles are performed by the CPU in two clock periods, which are labeled T1 and T4. During T1, the CPU activates $\overline{SPC}$ and, if it is writing to the MMU, it presents data on the bus. During T4, the CPU deactivates $\overline{SPC}$ and, if it is reading from the MMU, it latches data from the bus. The CPU guarantees that data written to the MMU is held through T4 to provide for the MMU's hold time requirements. The CPU also guarantees that the status code on ST0–ST3 becomes valid, at the latest, during the clock period preceding T1. The status code changes during T4 to anticipate the next bus cycle, if any.

Note that Slave Processor bus cycles are never extended with WAIT states. The RDY line is not sampled.

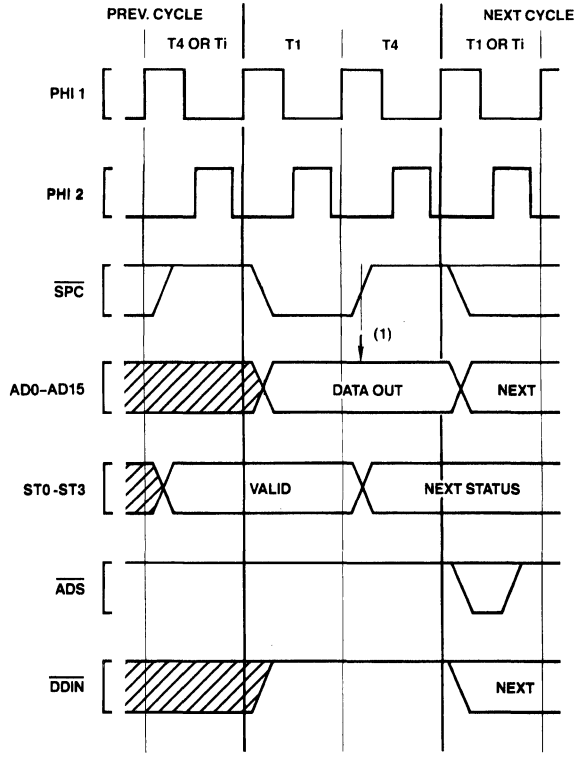


TL/EE/8692–13

Note 1: CPU samples Data Bus here.

FIGURE 2-10. Slave Access Timing; CPU Reading from MMU

2.0 Functional Description (Continued)



Note 1: MMU samples Data Bus here.

FIGURE 2-11. Slave Access Timing; CPU Writing to MMU

2.5.2 Instruction Protocols

MMU instructions have a three-byte Basic Instruction field consisting of an ID byte followed by an Operation Word. See *Figure 3-10* for the MMU instruction encodings. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.
2) It specifies that the MMU will execute it.
3) It determines the format of the following Operation Word of the instruction.

The CPU initiates an MMU instruction by issuing first the ID Byte and then the Operation Word, using Slave Processor bus cycles. The ID Byte is sent on the least-significant byte of the bus, in conjunction with status code 1111 (Broadcast ID Byte). The Operation Word is sent on the entire 16-bit data bus, with status code 1101 (Transfer Operation Word / Operand). The Operation Word is sent with its bytes swapped; i.e., its least-significant byte is presented to the MMU on the most-significant half of the 16-bit bus.

Other actions are taken by the CPU and the MMU according to the instruction under execution, as shown in Tables 2-2, 2-3 and 2-4.

In executing the LMR instruction (Load MMU Register, Table 2-2), the CPU issues the ID Byte, the Operation Word, and then the operand value to be loaded by the MMU. The register to be loaded is specified in a field within the Operation Word of the instruction.

In executing the SMR instruction (Store MMU Register, Table 2-3), the CPU also issues the ID Byte and the Operation Word of the instruction to the MMU. It then waits for the MMU to signal (by pulsing $\overline{SPC}$ low) that it is ready to present the specified register's contents to the CPU. Upon receiving this "Done" pulse, the CPU reads first a "Status Word" (dictated by the protocol for Slave Processor instructions) which the MMU provides as a word of all zeroes. The CPU then reads the contents of the selected register in two successive Slave Processor bus cycles, and places this result value into the instruction's destination (a CPU general-purpose register or a memory location).

In executing the RDVAL (Read-Validate) or WRVAL (Write-Validate) instruction, the CPU again issues the ID Byte and the Operation Word to the MMU. However, its next action is to initiate a one-byte Read cycle from the memory address whose protection level is being tested. It does so while presenting status code 1010; this being the only place that this status code appears during a RDVAL or WRVAL instruction. This memory access triggers a special address translation from the MMU. The translation is performed by the MMU using User-Mode mapping, and any protection violation occurring during this memory cycle does not cause an Abort. The MMU will, however, abort the CPU if the Level-1 Page Table Entry is invalid.

Upon completion of the address translation, the MMU pulses $\overline{SPC}$ to acknowledge that the instruction may continue execution.

2.0 Functional Description (Continued)

TABLE 2-2. LMR Instruction Protocol

CPU Action	Status	MMU Action
Issues ID Byte of instruction, pulsing $\overline{SPC}$.	1111	Accepts ID Byte.
Sends Operation Word of Instruction, pulsing $\overline{SPC}$.	1101	Decodes instruction.
Issues low-order word of new register value to MMU, pulsing $\overline{SPC}$.	1101	Accepts word from bus; places it into low-order half of referenced MMU register.
Issues high-order word of new register value to MMU, pulsing $\overline{SPC}$.	1101	Accepts word from bus; places it into high-order half of referenced MMU register.

TABLE 2-3. SMR Instruction Protocol

CPU Action	Status	MMU Action
Issues ID Byte of Instruction, pulsing $\overline{SPC}$.	1111	Accepts ID Byte.
Sends Operation Word of instruction, pulsing $\overline{SPC}$.	1101	Decodes instruction.
Waits for Done pulse from MMU.	xxxx	Sends Done pulse on $\overline{SPC}$.
Pulses $\overline{SPC}$ and reads Status Word from MMU.	1110	Presents Status Word (all zeroes) on bus.
Pulses $\overline{SPC}$, reading low-order word of result from MMU.	1101	Presents low-order word of referenced MMU register on bus.
Pulses $\overline{SPC}$, reading high-order word of result from MMU.	1101	Presents high-order word of referenced MMU register on bus.

TABLE 2-4. RDVAL/WRVAL Instruction Protocol

CPU Action	Status	MMU Action
Issues ID Byte of instruction, pulsing $\overline{SPC}$.	1111	Accepts ID Byte.
Sends Operation Word of instruction, pulsing $\overline{SPC}$.	1101	Decodes instruction.
Performs dummy one-byte memory read from operand's location.	1010	Translates CPU's address, using User-Mode mapping, and performs requested test on the address presented by the CPU. Aborts the CPU if the level-1 page table entry is invalid. Starts a Memory Cycle from the Translated Address if the translation is successful. Aborts on protection violations are temporarily suppressed.
Waits for Done pulse from MMU	xxxx	Sends Done pulse on $\overline{SPC}$.
Sends $\overline{SPC}$ pulse and reads Status Word from MMU; places bit 5 of this word into the F bit of the PSR register.	1110	Presents Status Word on bus, indicating in bit 5 the result of the test.

If the translation is successful the MMU will also start a dummy memory cycle from the translated address. See *Figure 2-12*. Note that, during this time the CPU will monitor the RDY line. Therefore, for proper operation, the RDY line must be kept high if the memory cycle is not performed.

The CPU then reads from the MMU a Status Word. Bit 5 of this Status Word indicates the result of the instruction:

0 if the CPU in User Mode could have made the corresponding access to the operand at the specified address (Read in RDVAL, Write in WRVAL),

1 if the CPU would have been aborted for a protection violation.

Bit 5 of the Status Word is placed by the CPU into the F bit of the PSR register, where it can be tested by subsequent instructions as a condition code.

Note: The MMU sets the R bit on RDVAL; R and M bits on WRVAL.

2.6 BUS ACCESS CONTROL

The NS32082 MMU has the capability of relinquishing its access to the bus upon rquest from a DMA device. It does this by using $\overline{HOLD}$, $\overline{HLDAI}$ and $\overline{HLDAO}$.

Details on the interconnections of these pins are provided in *Figures A-1* and *A-2* (Appendix A).

Requests for DMA are presented in parallel to both the CPU and MMU on the $\overline{HOLD}$ pin of each. The component that currently controls the bus then activates its Hold Acknowledge output to grant bus access to the requesting device. When the CPU grants the bus, the MMU passes the CPU's $\overline{HLDA}$ signal to its own $\overline{HLDAO}$ pin. When the MMU grants the bus, it does so by activating its $\overline{HLDAO}$ pin directly, and the CPU is not involved. $\overline{HLDAI}$ in this case is ignored.

Refer to *Figures 4-14, 4-15* and *4-16* for details on bus granting sequences.

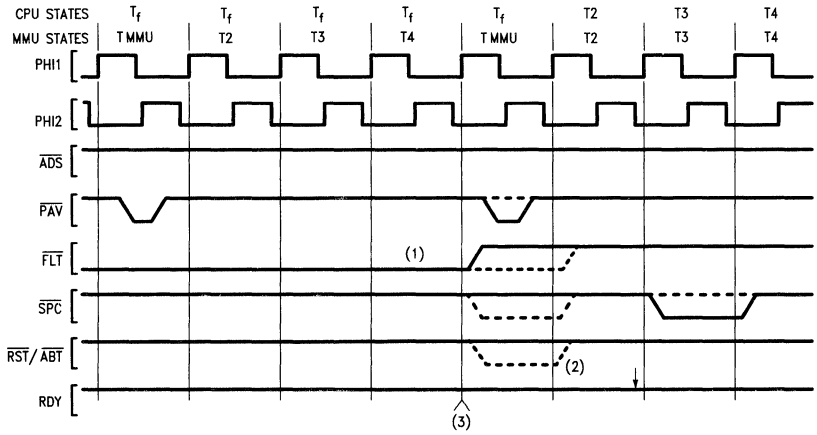
2.7 BREAKPOINTING

The MMU provides the ability to monitor references to two memory locations in real time, generating a Breakpoint trap on occurrence of any specified type of reference to either location made by a program. In addition, a Breakpoint trap may be inhibited until a specified number of such references have been performed.

Breakpoint monitoring is enabled and regulated by the setting of appropriate bits in the MSR and BPR0-1 registers. See Sections 3.5 and 3.7.

A Breakpoint trap is signalled to the CPU as either a Non-Maskable Interrupt or an Abort trap, depending on the setting of the AI bit in the MSR register.

2.0 Functional Description (Continued)



TL/EE/8692–15

Note 1: $\overline{FLT}$ is asserted if the translation is not in the TLB or a WRVAL instruction is executed and the M Bit is not set.

Note 2: If the Level-1 PTE is not valid, an abort is generated, $\overline{SPC}$ is issued in TMMU and $\overline{FLT}$ is deasserted in T_2.

Note 3: If a protection violation occurs or the Level-2 PTE is invalid, an Idle State is inserted here, $\overline{PAV}$ is not pulsed and $\overline{SPC}$ is pulsed during this Idle State.

FIGURE 2-12. $\overline{FLT}$ Deassertion During RDVAL/WRVAL Execution

The MSR register also indicates which breakpoint register triggered the break, and the direction (read or write) and type of memory cycle that was detected. The breakpoint address is not placed into the EIA register, as this register holds the addresses of address translation errors only. The breakpoint address is, however, available in the indicated Breakpoint register.

On occurrence of any trap generated by the MMU, including the Breakpoint trap, the BEN bit in the MSR register is immediately cleared, disabling any further Breakpoint traps.

Enabling breakpoints may cause variations in the bus timing given in the previous sections. Specifically:

1) While either breakpoint is enabled to monitor physical addresses, the MMU inserts an additional clock period into all bus cycles by asserting the $\overline{FLT}$ line for one clock. See *Figure 2-13.*

2) If the CPU initiates an instruction prefetch from a location at which a breakpoint is enabled on Execution, the MMU asserts the $\overline{FLT}$ line to the CPU, performs the memory cycle itself, and issues an edited instruction word to the CPU. See *Figure 2-14* and Section 2.7.1.

Note: Instructions which use two operands, a read-type and a write-type (e.g., MOVD 0(r1).0(r2), with the first operand valid and protected to allow user reads, and the second operand either invalid (page fault) or write protected, cause a read-type break event to occur for the first operand regardless of the outcome of the instruction. Each time the instruction is retried, the read-event is recorded. Hence, the breakpoint count register may reflect a different count than a casual assumption would lead one to. The same effect can occur on a RMW type operand with read only protection.

2.7.1 Breakpoints on Execution

The Series 32000 CPUs have an instruction prefetch which requires synchronization with execution breakpoints. In consideration of this, the MMU only issues an execution breakpoint when an instruction is prefetched with a nonsequential status code and the conditions specified in a breakpoint register are met. This guarantees that the instruction prefetch queue is empty and there are not pending instructions in the pipeline. There are three cases to consider:

Case 1: A nonsequential instruction prefetch is made to a breakpointed address.

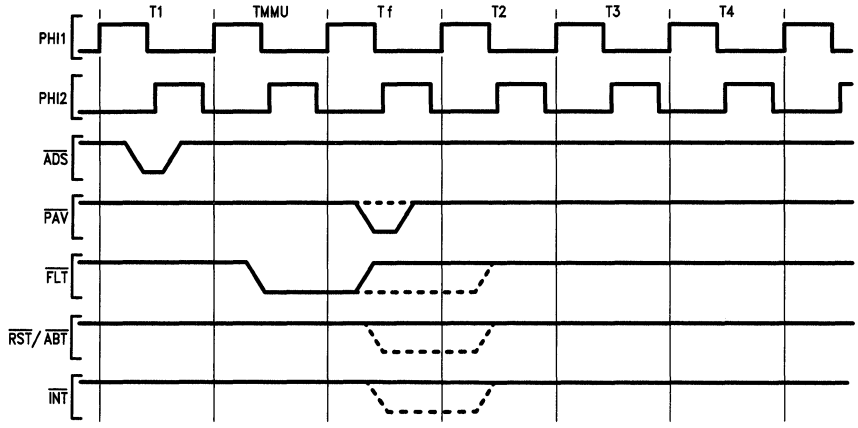
Response: The queue is necessarily empty. The breakpoint is issued.

Case 2, 3: A sequential prefetch is made to a breakpointed address OR a prefetch is made to an even address and the breakpoint is on the next odd address.

Response: In these cases, there may be instructions pending in the queue which must finish before the breakpoint is fired. Instead of putting the opcode byte (the one specified by the breakpointed address) in the queue, a DIA instruction is substituted for it. DIA is a single byte instruction which branches to itself, causing a queue flush. When the DIA executes, the breakpoint address is again issued, this time with nonsequential fetch status and the problem is reduced to case 1.

Note: Execution breakpoints cannot be used when the MMU is connected to either an NS32032 or an NS32332 CPU.

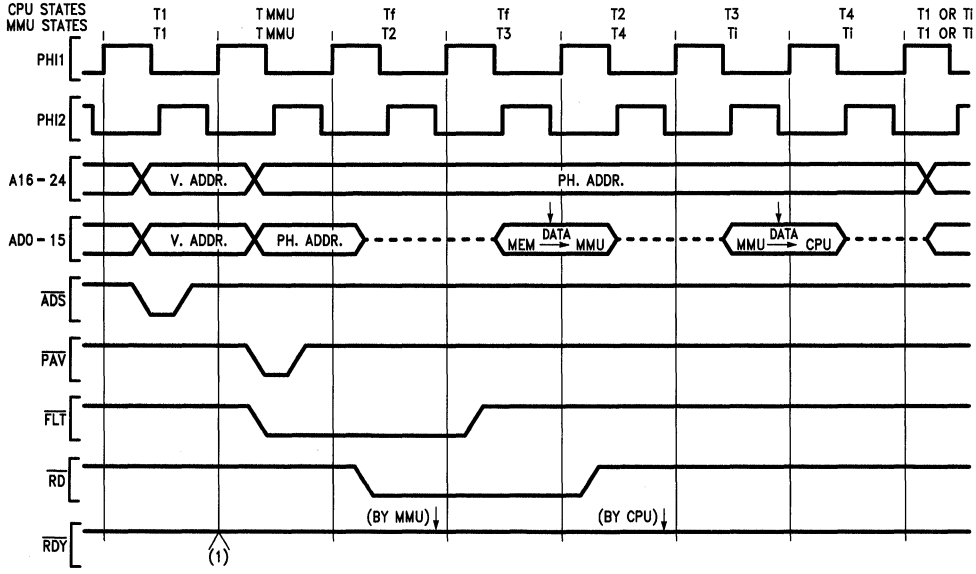
2.0 Functional Description (Continued)



TL/EE/8692-16

Note: If a breakpoint condition is met and abort on breakpoint is enabled, the bus cycle is aborted. In this case FLT is stretched by one clock cycle.

FIGURE 2-13. Bus Timing with Breakpoint on Physical Address Enabled



TL/EE/8692-17

Note 1: If a breakpoint on physical address is enabled, an extra clock cycle is inserted here.

FIGURE 2-14. Execution Breakpoint Timing; Insertion of DIA Instruction

3.0 Architectural Description

3.1 PROGRAMMING MODEL

The MMU contains a set of registers through which the CPU controls and monitors management and debugging functions. These registers are not memory-mapped. They are examined and modified by executing the Slave Processor instructions LMR (Load Memory Management Register) and SMR (Store Memory Management Register). These instructions are explained in Section 3.11, along with the other Slave Processor instructions executed by the MMU.

A brief description of the MMU registers is provided below. Details on their formats and functions are provided in the following sections.

PTB0, PTB1—Page Table Base Registers. They hold the physical memory addresses of the Page Tables referenced by the MMU for address translation. See Section 3.3.

EIA—Error/Invalidate Register. Dual-function register, used to display error addresses and also to purge cached translation information from the TLB. See Section 3.4.

BPR0, BPR1—Breakpoint Registers. Specify the conditions under which a breakpoint trap is generated. See Section 3.5.

BCNT—Breakpoint Counter Register. 24-bit counter used to count BPR0 events. Allows the breakpoint trap from the BPR0 register to be inhibited until a specified number of events have occurred. See Section 3.6.

MSR—Memory Management Status Register. Contains basic control and status fields for all MMU functions. See Section 3.7.

3.2 MEMORY MANAGEMENT FUNCTIONS

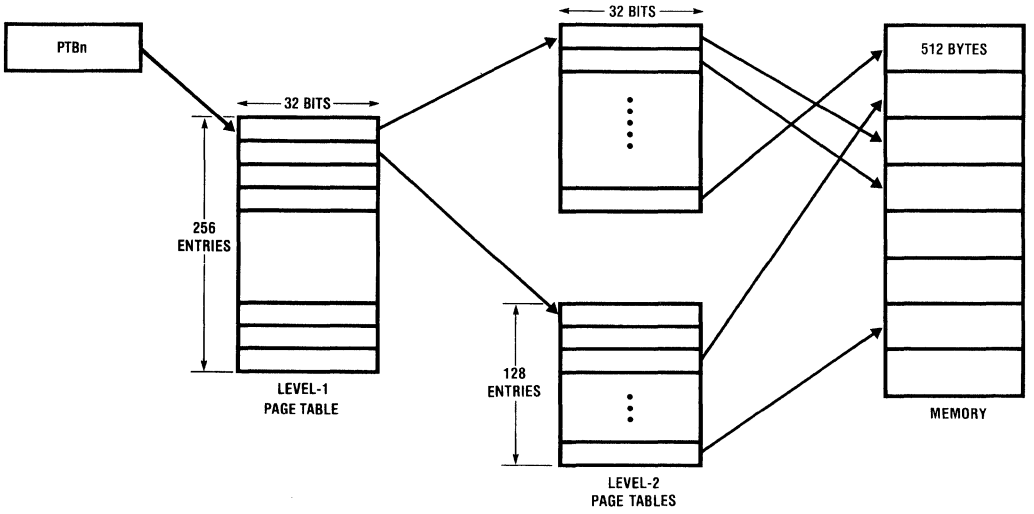
The NS32082 uses sets of tables in physical memory (the "Page Tables") to define the mapping from virtual to physical addresses. These tables are found by the MMU using one of its two Page Table Base registers: PTB0 or PTB1. Which register is used depends on the currently selected address space. See Section 3.2.2.

3.2.1. Page Table Structure

The page tables are arranged in a two-level structure, as shown in *Figure 3-1*. Each of the MMU's PTBn registers may point to a Level-1 page table. Each entry of the Level-1 page table may in turn point to a Level-2 page table. Each Level-2 page table entry contains translation information for one page of the virtual space.

The Level-1 page table must remain in physical memory while the PTBn register contains its address and translation is enabled. Level-2 Page Tables need not reside in physical memory permanently, but may be swapped into physical memory on demand as is done with the pages of the virtual space.

The Level-1 Page Table contains 256 32-bit Page Table Entries (PTE'S) and therefore occupies 1 Kbyte. Each entry of the Level-1 Page Table contains fields used to construct the physical base address of a Level-2 Page Table. These fields are a 15-bit PFN field, providing bits 9-23 of the physical address, and an MS bit providing bit 24. The remaining bits (0–8) are assumed zero, placing a Level-2 Page Table always on a 512-byte (page) boundary.



TL/EE/8692–18

FIGURE 3-1. Two-Level Page Tables

3.0 Architectural Description (Continued)

Level-2 Page Tables contain 128 32-bit Page Table entries, and so occupy 512 bytes (1 page). Each Level-2 Page Table Entry points to a final 512-byte physical page frame. In other words, its PFN and MS fields provide the Page Frame Number portion (bits 9-24) of the translated address (*Figure 3-3*). The OFFSET field of the translated address is taken directly from the corresponding field of the virtual address.

3.2.2 Virtual Address Spaces

When the Dual Space option is selected for address translation in the MSR (Sec. 3.7) the MMU uses two maps: one for translating addresses presented to it in Supervisor Mode and another for User Mode addresses. Each map is referenced by the MMU using one of the two Page Table Base registers: PTB0 or PTB1. The MMU determines the CPU's current mode by monitoring the state of the U/$\overline{S}$ pin and applying the following rules.

1) While the CPU is in Supervisor Mode (U/$\overline{S}$ pin = 0), the CPU is said to be presenting addresses belonging to Address Space 0, and the MMU uses the PTB0 register as its reference for looking up translations from memory.
2) While the CPU is in User Mode (U/$\overline{S}$ pin = 1), and the MSR DS bit is set to enable Dual Space translation, the CPU is said to be presenting addresses belonging to Address Space 1, and the MMU uses the PTB1 register to look up translations.
3) If Dual Space translation is not selected in the MSR, there is no Address Space 1, and all addresses presented in both Supervisor and User modes are considered by the MMU to be in Address Space 0. The privilege level of the CPU is used then only for access level checking.

Note: When the CPU executes a Dual-Space Move instruction (MOVUSi or MOVSUi), it temporarily enters User Mode by switching the state of the U/$\overline{S}$ pin. Accesses made by the CPU during this time are treated by the MMU as User-Mode accesses for both mapping and access level checking. It is possible, however, to force the MMU to assume Supervisor-Mode privilege on such accesses by setting the Access Override (AO) bit in the MSR (Sec. 3.7).

3.2.3 Page Table Entry Formats

Figure 3-2 shows the formats of Level-1 and Level-2 Page Table Entries (PTE's). Their formats are identical except for the "M" bit, which appears only in a Level-2 PTE.

The bits are defined as follows:

V Valid. The V bit is set and cleared only by software.

V=1=> The PTE is valid and may be used for translation by the MMU.

V=0=> The PTE does not represent a valid translation. Any attempt to use this PTE will cause the MMU to generate an Abort trap. While V=0, the operating system may use all other bits except the PL field for any desired function.

PL Protection Level. This two-bit field establishes the types of accesses permitted for the page in both User Mode and Supervisor Mode, as shown in Table 3-1.

The PL field is modified only by software. In a Level-1 PTE, it limits the maximum access level allowed for all pages mapped through that PTE.

TABLE 3-1. Access Protection Levels

Mode	U/$\overline{S}$	Protection Level Bits (PL)			
		00	01	10	11
User	1	no access	no access	read only	full access
Supervisor	0	read only	full access	full access	full access

R Referenced. This is a status bit, set by the MMU and cleared by the operating system, that indicates whether the page mapped by this PTE has been referenced within a period of time determined by the operating system. It is intended to assist in implementing memory allocation strategies. In a Level-1 PTE, the R bit indicates only that the Level-2 Page Table has been referenced for a translation, without necessarily implying that the translation was successful. In a Level-2 PTE, it indicates that the page mapped by the PTE has been successfully referenced.

R=1=> The page has been referenced since the R bit was last cleared.

R=0=> The page has not been referenced since the R bit was last cleared.

Note: The RDVAL and WRVAL instructions set the Level-1 and Level-2 bits for the page whose protection level is tested. See Sections 2.5.2 and 3.11.

M Modified. This is a status bit, set by the MMU whenever a write cycle is successfully performed to the page mapped by this PTE. It is initialized to zero by the operating system when the page is brought into physical memory.

M=1=> The page has been modified since it was last brought into physical memory.

M=0=> The page has not been modified since it was last brought into physical memory.

In Level-1 Page Table Entries, this bit position is undefined, and is altered in an undefined manner by the MMU while the V bit is 1.

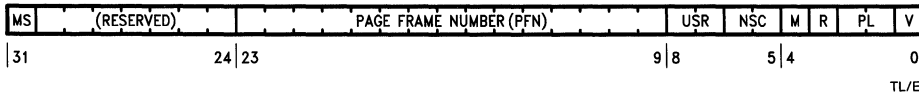
Note: The WRVAL instruction sets the M bit for the page whose protection level is tested. See Sections 2.5.2 and 3.11.

NSC Reserved. These bits are ignored by the MMU and their values are not changed.

They are reserved by National, and therefore should not be used by the user software.

USR User bits. These bits are ignored by the MMU and their values are not changed.

They can be used by the user software.



TL/EE/8692–19

FIGURE 3-2. A Page Table Entry

3.0 Architectural Description (Continued)

PFN Page Frame Number. This 15-bit field provides bits 9-23 of the Page Frame Number of the physical address. See *Figure 3-3.*

MS Memory System. This bit represents the most significant bit of the physical address, and is presented by the MMU on pin A24. This bit is treated by the MMU no differently than any other physical address bit, and can be used to implement a 32-Mbyte physical addressing space if desired.

3.2.4 Physical Address Generation

When a virtual address is presented to the MMU by the CPU and the translation information is not in the TLB, the MMU performs a page table lookup in order to generate the physical address.

The Page Table structure is traversed by the MMU using fields taken from the virtual address. This sequence is diagrammed in *Figure 3-3.*

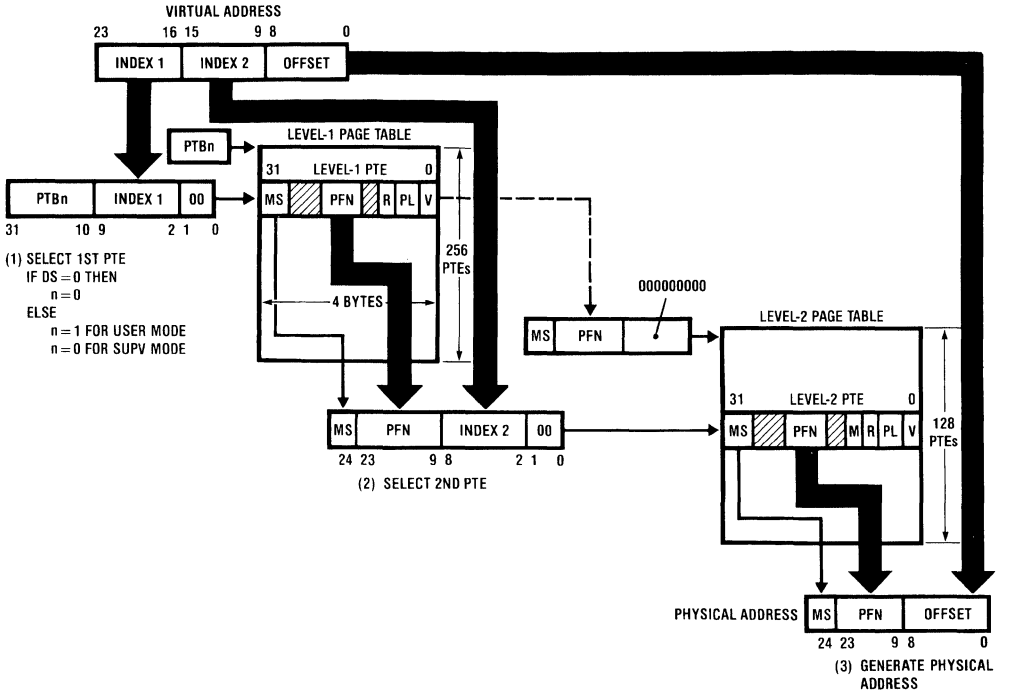
Bits 9–23 of the virtual address hold the 15-bit Page Number, which in the course of the translation is replaced with the 16-bit Page Frame Number of the physical address. The virtual Page Number field is further divided into two fields, INDEX 1 and INDEX 2.

Bits 0–8 constitute the OFFSET field, which identifies a byte's position within the accessed page. Since the byte position within a page does not change with translation, this value is not used, and is simply echoed by the MMU as bits 0–8 of the final physical address.

The 8-bit INDEX 1 field of the virtual address is used as an index into the Level-1 Page Table, selecting one of its 256 entries. The address of the entry is computed by adding INDEX 1 (scaled by 4) to the contents of the current Page Table Base register. The PFN and MS fields of that entry give the base address of the selected Level-2 Page Table.

The INDEX 2 field of the virtual address (7 bits) is used as the index into the Level-2 Page Table, by adding it (scaled by 4) to the base address taken from the Level-1 Page Table Entry. the PFN and MS fields of the selected entry provide the entire Page Frame Number of the translated address.

The offset field of the virtual address is then appended to this frame number to generate the final physical address.



TL/EE/8692–20

FIGURE 3-3. Virtual to Physical Address Translation

3.0 Architectural Description (Continued)

3.3 PAGE TABLE BASE REGISTERS (PTB0, PTB1)

The PTBn registers hold the physical addresses of the Level-1 Page Tables.

The format of these registers is shown in *Figure 3-4.* The least-significant 10 bits are permanently zero, so that each register always points to a 1 Kbyte boundary in memory.

The PTBn registers may be loaded or stored using the MMU Slave Processor instructions LMR and SMR (Section 3.11).

3.4 ERROR/INVALIDATE ADDRESS REGISTER (EIA)

The Error/Invalidate Address register is a dual-purpose register.

1) When it is read using the SMR instruction, it presents the virtual address which last generated an address translation error.
2) When a virtual address is written into it using the LMR instruction, the translation for that virtual address is purged, if present, from the TLB. This must be done whenever a Page Table Entry has been changed in memory, since the TLB might otherwise contain an incorrect translation value.

The format of the EIA register is shown in *Figure 3-5.* When a translation error occurs, the cause of the error is reported by the MMU in the appropriate fields of the MSR register (Section 3.7). The ADDRESS field of the EIA register holds the virtual address at which the error occurred, and the AS bit indicates the address space that was in use.

In writing a virtual address to the EIA register, the virtual address is specified in the low-order 24 bits, and the AS bit specifies the address space. A TLB entry is purged only if it matches both the ADDRESS and AS fields.

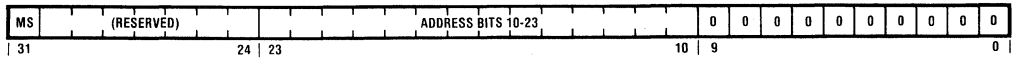
Another technique for purging TLB entries is to load a PTBn register. This automatically purges all entries associated with the addressing space mapped by that register. Turning off translation (clearing the MSR TU and/or TS bits) does not purge any entries from the TLB.

3.5 BREAKPOINT REGISTERS (BPR0, BPR1)

The Breakpoint registers BPR0 and BPR1 specify the addresses and conditions on which a Breakpoint trap will be generated. They are each 32 bits in length and have the format shown in *Figure 3-6.* All implemented bits of BPR0 and BPR1 are readable and writable.

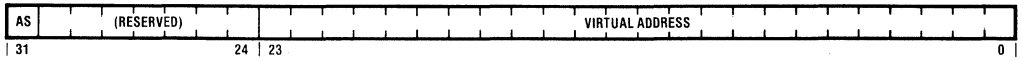
Bits 0 through 23 and bit 31 (AS) specify the breakpoint address. This address may be either virtual or physical, as specified in the VP bit.

Bits 24 and 25 are not implemented. Bit 26 (CE) is not implemented in register BPR1.



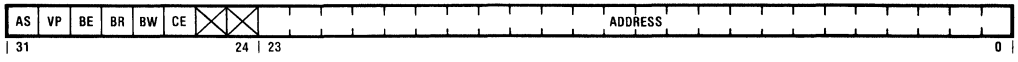
TL/EE/8692–21

FIGURE 3-4. Page Table Base Registers (PTB0, PTB1)



TL/EE/8692–22

FIGURE 3-5. EIA Register



TL/EE/8692–23

FIGURE 3-6. Breakpoint Registers (BPR0, BPR1)

3.0 Architectural Description (Continued)

Bits 26 through 30 specify the breakpoint conditions. Breakpoint conditions define how the breakpoint address is compared and which conditions permit a break to be generated. A Breakpoint register can be selectively disabled by setting all of these bits to zero.

AS Address Space. This bit depends on the setting of the VP bit. For virtual addresses, this bit contains the AS (Address Space) qualifier of the virtual address (Section 3.2.2). For physical addresses, this bit contains the MS (Memory System) bit of the physical address.

VP Virtual/Physical. If VP is 0, the breakpoint address is compared against each referenced virtual address. If VP is 1, the breakpoint address is compared against each physical address that is referenced by the CPU (i.e. after translation).

BE Break on Execution. If BE is 1, a break is generated immediately before the instruction at the breakpoint address is executed. While this option is enabled, the breakpoint address must be the address of the first byte of an instruction. If BE is 0, this condition is disabled.

Note: This option cannot be used in systems based on any CPU with a 32-bit wide bus.

The BE bit should only be set when the CPU has a 16-bit bus (i.e. NS32016, NS32C016). In other systems, use instead the BPT instruction placed in memory, to signal a break.

BR Break on Read. If BR is 1, a break is generated when data is read from the breakpoint address. Instruction fetches do not trigger a Read breakpoint. If BR is 0, this condition is disabled.

BW Break on Write. If BW is 1, a break is generated when data is written to the breakpoint address or when data is read from the breakpoint address as the first part of a read-modify-write access. If BW is 0, this condition is disabled.

CE Counter Enable. This bit is implemented only in the BPR0 register. If CE is 1, no break is generated unless the Breakpoint Count register (BCNT, see below) is zero. The BCNT register decrements when the condition for the breakpoint in register BPR0 is met and the BCNT register is not already zero. If CE is 0, the BCNT register is disabled, and breaks from BPR0 occur immediately.

Note 1: The bits BR, BW and CE should not all be set. The counting performed by the MMU becomes inaccurate, and in Abort Mode (MSR AI bit set), it can trap a program in such a way as to make it impossible to retry the breakpointed instruction correctly.

Note 2: An execution breakpoint should not be counted (BE and CE bits both set) if it is placed at an address that is the destination of a branch, or if it follows a queue-flushing instruction. See Table 3-2. The counting performed by the MMU will be inaccurate if interrupts occur during the fetch of that address.

TABLE 3-2. Instructions Causing Non-Sequential Fetches

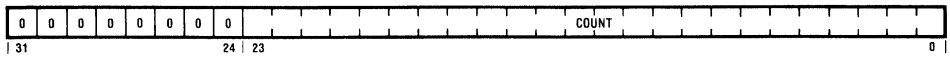
Branch	
ACBi	Add, Compare and Branch: unless result is zero
BR	Branch (Unconditional)
BSR	Branch to Subroutine
Bcond	Branch (Conditional): only if condition is met
CASEi	Case Branch
CXP	Call External Procedure
CXPD	Call External Procedure with Descriptor
DIA	Diagnose
JSR	Jump to Subroutine
JUMP	Jump
RET	Return from Subroutine
RXP	Return from External Procedure
BPT	Breakpoint Trap
FLAG	Trap on Flag
RETI	Return from Interrupt: if MSR loaded properly by supervisor
RETT	Return from Trap: if MSR loaded properly by supervisor
SVC	Supervisor Call

Also all traps or interrupts not generated by the MMU.

Branch to Following Instruction	
BICPSRi	Bit Clear in PSR
BISPSRi	Bit Set in PSR
LMR	Load Memory Management Register
LPRi	Load Processor Register: unless UPSR is the register specified
MOVSUi	Move Value from Supervisor to User Space
MOVUSi	Move Value from User to Supervisor Space
WAIT	Wait: fetches next instruction before waiting

3.6 BREAKPOINT COUNT REGISTER (BCNT)

The Breakpoint Count register (BCNT) permits the user to specify the number of breakpoint conditions given by register BPR0 that should be ignored before generating a Breakpoint trap. The BCNT register is 32 bits in length, containing a counter in its low-order 24 bits, as shown in *Figure 3-7.* The high-order eight bits are not used.



TL/EE/8692–24

FIGURE 3-7. Breakpoint Count Register (BCNT)

3.0 Architectural Description (Continued)

The BCNT register affects the generation of Breakpoint traps only when it is enabled by the CE bit in the BPR0 register. When the BPR0 breakpoint condition is encountered, and the BPR0 CE bit is 1, the contents of the BCNT register are checked against zero. If the BCNT contents are zero, a breakpoint trap is generated. If the contents are not equal to zero, no breakpoint trap is generated and the BCNT register is decremented by 1.

If the CE bit in the BPR0 register is 0, the BCNT register is ignored and the BPR0 condition breaks the program execution regardless of the BCNT register's contents. The BCNT register contents are unaffected.

3.7 MEMORY MANAGEMENT STATUS REGISTER (MSR)

The Memory Management Status Register (MSR) provides overall control and status fields for both address translation and debugging functions. The format of the MSR register is shown in *Figure 3-8*.

The MSR fields relevant to either of the above functions are described in the following sub-sections.

3.7.1 MSR Fields for Address Translation.

Control Functions

The address translation control bits in the MSR, ad exception of the R bit, are both readable (using the SMR instruction) and writable (using LMR).

R Reset. When read, this bit's contents are undefined. Whenever a "1" is written into it, MSR status fields TE, B, TET, ED, BD, EST and BST are cleared to all zeroes. (The BN bit is not affected.)

TU Translate User-Mode Addresses. While this bit is "1", the MMU translates all addresses presented while the CPU is in User Mode. While it is "0", the MMU echoes all User-Mode virtual addresses without performing translation or access level checking. This bit is cleared by a hardware Reset.

Note: Altering the TU bit has no effect on the contents of the TLB.

TS Translate Supervisor-Mode Addresses. While this bit is "1", the MMU translates all addresses presented while the CPU is in Supervisor Mode. While it is "0", the MMU echoes all Supervisor-Mode virtual addresses without translation or access level checking. This bit is cleared by a hardware Reset.

Note: Altering the TS bit has no effect on the contents of the TLB.

DS Dual-Space Translation. While this bit is "1", Supervisor Mode addresses and User Mode addresses are translated independently of each other, using separate mappings. While it is "0", both Supervisor Mode addresses and User Mode addresses are translated using the same mapping. See Section 3.2.2.

AO Access Level Override. This bit may be set to temporarily cause User Mode accesses to be given Supervisor Mode privilege. See Section 3.10.

Status Fields

The MSR status fields may be read using the MSR instruction, but are not writable. Instead, all status fields (except the BN bit) may be cleared by loading a "1" into the R bit using the LMR instruction.

TE Translation Error. This bit is set by the MMU to indicate that an address translation error has occurred. This bit is cleared by a hardware reset.

TET Translation Error Type. This three-bit field shows the reason(s) for the last address translation error reported by the MMU. The format of the TET field is shown below.

IL2	IL1	PL

PL Protection Level error. The access attempted by the CPU was not allowed by the protection level assigned to the page it attempted to access (forbidden by either of the Page Table Entry PL fields).

IL1 Invalid Level 1. The Level-1 Page Table Entry was invalid (V bit = 0).

IL2 Invalid Level 2. The Level-2 Page Table Entry was invalid (V bit = 0).

These error indications are not mutually exclusive. A protection level error and an invalid translation error can be reported simultaneously by the MMU.

ED Error Direction. This bit indicates the direction of the transfer that the CPU was attempting on the most recent address translation error.

ED = 0 => Write cycle.

ED = 1 => Read cycle.

EST Error Status. This 3-bit field is set on an address translation error to the low-order three bits of the CPU status bus. Combinations appearing in this field are summarized below.

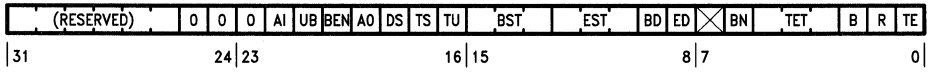
000 Sequential instruction fetch

001 Non-sequential instruction fetch

010 Operand transfer (read or write)

011 The Read action of a read-modify-write transfer (operands of access class "rmw" only: See the Series 32000 Instruction Set Reference Manual for further details).

100 A read transfer which is part of an effective address calculation (Memory Relative or External mode)



TL/EE/8692-25

Note: In some Series 32000 documentation, the bits TE, R and B are jointly referenced with the keyword "ERC".

FIGURE 3-8. Memory Management Status Register (MSR)

3.0 Architectural Description (Continued)

3.7.2 MSR Fields for Debugging

Control Functions

Breakpoint control bits in the MSR are both readable (using the SMR instruction) and writable (using LMR).

BEN Breakpoint Enable. Setting this bit enables both Breakpoint Registers (BPR0, BPR1) to monitor CPU activity. This bit is cleared by a hardware reset or whenever a Breakpoint trap or an address translation error occurs. If only one breakpoint register must be enabled, the other register should be disabled by clearing all of its control bits (bits 26–31) to zeroes.

Note: When the BEN bit is set (using the LMR instruction), the MMU enables breakpoints only after two non-sequential instruction fetch cycles have been completed by the CPU. See Section 3.9.

UB User-Only Breakpointing. When this bit is set in conjunction with the BEN bit, it limits the Breakpoint Registers to monitor addresses only while the CPU is in User Mode.

AI Abort/Interrupt. This bit selects the action taken by the MMU on a breakpoint. While AI is "0" the MMU generates a pulse on the $\overline{INT}$ pin (this can be used to generate a non-maskable interrupt). While AI is "1" the MMU generates an Abort pulse instead.

Status Fields

The MSR status fields may be read using the SMR instruction, but are not writable. Instead, all status fields (except the BN bit) may be cleared by loading a "1" into the R bit using the LMR instruction. See Section 3.7.1.

B Break. This bit is set to indicate that a breakpoint trap has been generated by the MMU.

BN Breakpoint Number. The BN bit contains the register number for the most recent breakpoint trap generated by the MMU. If BN is 1, the breakpoint was triggered by the BPR1 register. If BN is 0, the breakpoint was triggered by the BPR0 register. If both registers trigger a breakpoint simultaneously, the BN bit is set to 1.

BD Break Direction. This bit indicates the direction of the transfer that the CPU was attempting on the access that triggered the most recent breakpoint trap. It is loaded from the complement of the $\overline{DDIN}$ pin.

BD = 0 => Write cycle.

BD = 1 => Read cycle.

BST Breakpoint Status. This 3-bit field is loaded on a Breakpoint trap from the low-order three bits of the CPU status bus. Combinations appearing in this field are summarized below.

000 No break has occurred since the field was last reset.

001 Instruction fetch

010 Operand transfer (read or write)

011 The Read action of a read-modify-write transfer (operands of access class "rmw" only: See the Series 32000 Instruction Set Reference Manual for further details).

100 A read transfer which is part of an effective address calculation (Memory Relative or External mode)

Note: The BST field encodings 000 and 001 differ from those of the EST field (Section 3.7.1) because the MMU inserts a DIA instruction into the instruction stream in implementing Execution breakpoints (Section 2.7.1). One side effect of this is that a breakpoint trap is never triggered directly by a sequential instruction fetch cycle.

3.8 TRANSLATION LOOKASIDE BUFFER (TLB)

The Translation Lookaside Buffer is an on-chip fully associative memory. It provides direct virtual to physical mapping for the 32 most recently used pages, requiring only one clock period to perform the address translation.

The efficiency of the MMU is greatly increased by the TLB, which bypasses the much longer Page Table lookup in over 97% of the accesses made by the CPU.

Entries in the TLB are allocated and replaced by the MMU itself; the operating system is not involved. The TLB entries cannot be read or written by software; however, they can be purged from it under program control.

Figure 3-9 models the TLB. Information is placed into the TLB whenever the MMU peforms a lookup from the Page Tables in memory. If the retrieved mapping is valid (V = 1 in both levels of the Page Tables), and the access attempted is permitted by the protection level, an entry of the TLB is loaded from the information retrieved from memory. The recipient entry is selected by an on-chip circuit that implements a Least-Recently-Used (LRU) algorithm. The MMU places the virtual page number (15 bits) and the Address Space qualifier bit into the Tag field of the TLB entry.

The Value portion of the entry is loaded from the Page Tables as follows:

The Translation field (16 bits) is loaded from the MS bit and PFN field of the Level-2 Page Table Entry.

The M bit is loaded from the M bit of the Level-2 Page Table Entry.

The PL field (2 bits) is loaded to reflect the net protection level imposed by the PL fields of the Level-1 and Level-2 Page Table Entries.

(Not shown in the figure are additional bits associated with each TLB entry which flag it as full or empty, and which select it as the recipient when a Page Table lookup is performed.)

When a virtual address is presented to the MMU for translation, the high-order 15 bits (page number) and the Address Space qualifier are compared associatively to the corresponding fields in all entries of the TLB. When the Tag portion of a TLB entry completely matches the input values, the Value portion is produced as output. If the protection level is not violated, and the M bit does not need to be changed, then the physical address Page Frame number is output in the next clock cycle. If the protection level is violated, the MMU instead activates the Abort output. If no TLB entry matches, or if the matching entry's M bit needs to be changed, the MMU performs a page-table lookup from memory.

Note that for a translation to be loaded into the TLB it is necessary that the Level-1 and Level-2 Page Table Entries be valid (V bit = 1). Also, it is guaranteed that in

3.0 Architectural Description (Continued)

the process of loading a TLB entry (during a Page Table lookup) the Level-1 and Level-2 R bits will be set in memory if they were not already set. For these reasons, there is no need to replicate either the V bit or the R bit in the TLB entries.

Whenever a Page Table Entry in memory is altered by software, it is necessary to purge any matching entry from the TLB, otherwise the MMU would be translating the corresponding addresses according to obsolete information. TLB entries may be selectively purged by writing a virtual address to the EIA register using the LMR instruction. The TLB entry (if any) that matches that virtual address is then purged, and its space is made available for another translation. Purging is also performed by the MMU whenever an address space is remapped by altering the contents of the PTB0 or PTB1 register. When this is done, the MMU purges all the TLB entries corresponding to the address space mapped by that register. Turning translation on or off (via the MSR TU and TS bits) does not affect the contents of the TLB.

Note: If the value in the PTB0 register must be changed, it is strongly recommended that the translation be disabled before loading the new value, otherwise the purge performed may be incomplete. This is due to instruction prefetches and/or memory read cycles occurring during the LMR instruction which may restore TLB entries from the old map.

3.9 ENTRY/RE-ENTRY INTO PROGRAMS UNDER DEBUGGING

Whenever the MSR is written, breakpoints are disabled. After two non-sequential instruction fetch cycles have completed, they are again enabled if the new BEN bit value is '1'. The recommended sequence for entering a program under test is:

```
LMR    MSR, New_Value
RETT   n        ; or RETI
```

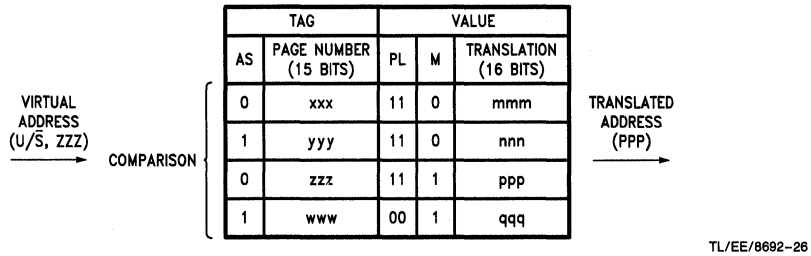
executed with interrupts disabled (CPU PSR I bit off).

This feature allows a debugger or monitor program to return control to a program being debugged without the risk of a false breakpoint trap being triggered during the return.

The LMR instruction performs the first non-sequential fetch cycle, in effect branching to the next sequential instruction. The RETT (or RETI) instruction performs the second non-sequential fetch as its last memory reference, branching to the first (next) instruction of the program under debug. The non-sequential fetch caused by the RETT instruction, which might not have occurred otherwise, is not monitored.

3.10 ADDRESS TRANSLATION ALGORITHM

The MMU either translates the 24-bit virtual address to a 25-bit physical address or reports a translation error. This process is described algorithmically in the following pages. See also *Figure 3-3*.



TL/EE/8692–26

FIGURE 3-9. TLB Model

MMU Page Table Lookup and Access Validation Algorithm

```
Legend:
x = y           x is assigned the value y
x == y          Comparison expression, true if x is equal to y
x AND y         Boolean AND expression, true only if assertions x and y are both true
x OR y          Boolean inclusive OR expression, true if either of assertions x and y is true
;               Delimiter marking end of statement
{ . . . }       Delimiters enclosing a statement block
item(i)         Bit number i of structure "item"
item(i:j)       The field from bit number i through bit number j of structure "item"
item.x          The bit or field named "x" in structure "item"
DONE            Successful end of translation; MMU provides translated address
ABORT           Unsuccessful end of translation; MMU aborts CPU access
This algorithm represents for all cases a valid definition of address translation.
Bus activity implied here occurs only if the TLB does not contain the mapping,
or if the reference requires that the MMU alter the M bit of the Page Table Entry.
Otherwise, the MMU provides the translated address in one clock period.
Input (from CPU):
     U (1 if U/S is high)
     W (1 if DDIN input is high)
     VA  Virtual address consisting of:
        INDEX_1 (from pins A23-A16)
        INDEX_2 (from pins AD15-AD9)
        OFFSET  (from pins AD8-AD0)
     ACCESS_LEVEL       The access level of a reference is a 2-bit value synthesized by the MMU from CPU status:
        bit 1 = U AND NOT MSR.A0  (U from U/S input pin)
        bit 0 = 1 for Write cycle, or Read cycle of an "rmw" class operand access
                0 otherwise.

Output:
     PA Physical Address on pins A24-A16, AD15-AD0;
        or
     Abort pulse on RST/ABT pin.
Uses:
     MSR        Status Register:
                        fields TU, TS and DS
```

MMU Page Table Lookup and Access Validation Algorithm (Continued)

```
     PTB0       Page Table Base Register 0
     PTB1       Page Table Base Register 1
     PTE_1      Level-1 Page Table Entry:
                        fields PFN, PL, V, R and MS
     PTEP_1     Pointer, holding address of PTE_1
     PTE_2      Level-2 Page Table Entry:
                        fields PFN, PL, V, M, R and MS
     PTEP_2     Pointer, holding address of PTE_2
IF ( (MSR.TU == 0) AND (U == 1) )  OR  ( (MSR.TS == 0) AND (U ==0) )  If translation not enabled then echo
     THEN  {  PA(0:23) = VA(0:23) ; PA(24) = 0 ; DONE  }  ;                 virtual address as physical address.

IF (MSR.DS == 1) AND (U == 1)                                          If Dual Space mode and CPU in User Mode
   THEN {  PTEP_1(24) = PTB1.MS ; PTEP_1(23:10) = PTB1(23:10) ;             then form Level-1 PTE address
           PTEP_1(9:2) = VA.INDEX_1 ; PTEP_1(1:0)=0  }                             from PTB1 register,
   ELSE {  PTEP_1(24) = PTB0.MS ; PTEP_1(23:10) = PTB0(23:10) ;             else form Level-1 PTE address
           PTEP_1(9:2) = VA.INDEX_1; PTEP_1(1:0) = 0                               from PTB0 register.
        } ;

                                              - - - LEVEL 1 PAGE TABLE LOOKUP - - -

IF ( ACCESS_LEVEL > PTE_1.PL ) OR (PTE_1.V == 0)                       If protection violation or invalid Level-2 page
   THEN ABORT ;                                                              table then abort the access.

IF PTE_1.R == 0 THEN PTE_1.R = 1 ;                                     Otherwise, set Reference bit if not already set,
PTE_1(4) = (undefined value);                                                  (the M bit position may be garbaged)

PTEP_2(24) = PTE_1.MS ; PTEP_2(23:9) = PTE_1.PFN ;                             and form Level-2 PTE address.
PTEP_2(8:2) = VA.INDEX_2 ; PTEP_2(1:0) = 0 ;

                                              - - - LEVEL 2 PAGE TABLE LOOKUP - - -

IF ( ACCESS_LEVEL > PTE_2.PL ) OR ( PTE_2.V ==0 )                          If protection violation or invalid page
   THEN ABORT ;                                                            then abort the access.

IF PTE_2.R == 0 THEN PTE_2.R == 1 ;                                    Otherwise, set Referenced bit if not already set,
IF ( W == 1) AND ( PTE_2.M == 0 ) THEN PTE_2.M = 1 ;                         if Write cycle set Modified bit if not
                                                                                  already set,
PA(24) = PTE_2.MS ; PA(23:9) = PTE_2.PFN ; PA(8:0) = VA.OFFSET ;                  and generate physical address.
DONE ;
```

3.0 Architectural Description (Continued)

3.11 INSTRUCTION SET

Four instructions of the Series 32000 instruction set are executed cooperatively by the CPU and MMU. These are:

LMR	Load Memory Management Register
SMR	Store Memory Management Register
RDVAL	Validate Address for Reading
WRVAL	Validate Address for Writing

The format of the MMU slave instructions is shown in *Figure 3-10*. Table 3-3 shows the encodings of the "short" field for selecting the various MMU internal registers.

TABLE 3-3. "Short" Field Encodings

"Short" Field	Register
0000	BPR0
0001	BPR1
1010	MSR
1011	BCNT
1100	PTB0
1101	PTB1
1111	EIA

Note: All other codes are illegal. They will cause unpredictable registers to be selected if used in an instruction.

For reasons of system security, all MMU instructions are privileged, and the CPU does not issue them to the MMU in User Mode. Any such attempt made by a User-Mode program generates the Illegal Operation trap, Trap (ILL). In addition, the CPU will not issue MMU instructions unless its CFG register's M bit has been set to validate the MMU instruction set. If this has not been done, MMU instructions are not recognized by the CPU, and an Undefined Instruction trap, Trap (UND), results.

The LMR and SMR instructions load and store MMU registers as 32-bit quantities to and from any general operand (including CPU General-Purpose Registers).

The RDVAL and WRVAL instructions probe a memory address and determine whether its current protection level would allow reading or writing, respectively, if the CPU were in User Mode. Instead of triggering an Abort trap, these instructions have the effect of setting the CPU PSR F bit if the type of access being tested for would be illegal. The PSR F bit can then be tested as a condition code.

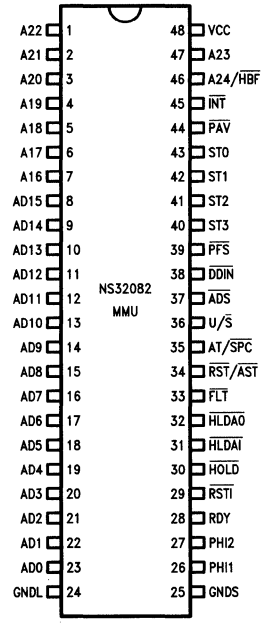
Note: The Series 32000 Dual-Space Move instructions (MOVSUi and MOVUSi), although they involve memory management action, are not Slave Processor instructions. The CPU implements them by switching the state of its U/$\overline{S}$ pin at appropriate times to select the desired mapping and protection from the MMU.

For full architectural details of these instructions, see the Series 32000 Instruction Set Reference Manual.

4.0 Device Specifications

4.1 NS32082 PIN DESCRIPTIONS

The following is a brief description of all NS32082 pins. The descriptions reference portions of the Functional Description, Section 2.0.



TL/EE/8692–28

Top View

Order Number NS16082D
See NS Package Number D48A

FIGURE 4-1. Dual-In-Line Package Connection Diagram

4.1.1 Supplies

Power (V_{CC}): +5V positive supply. Section 2.1.

Logic Ground (GNDL): Ground reference for on-chip logic. Section 2.1.

Buffer Ground (GNDB): Ground reference for on-chip drivers connected to output pins. Section 2.1.

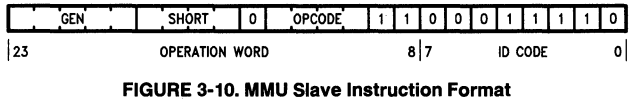
4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Section 2.2.

Ready (RDY): Active high. Used by slow memories to extend MMU originated memory cycles. Section 2.4.4.

Hold Request ($\overline{HOLD}$): Active low. Causes a release of the bus for DMA or multiprocessing purposes. Section 2.6.

Hold Acknowledge in ($\overline{HLDAI}$): Active low. Applied by the CPU in response to $\overline{HOLD}$ input, indicating that the CPU has released the bus for DMA or multiprocessing purposes. Section 2.6.



TL/EE/8692–27

FIGURE 3-10. MMU Slave Instruction Format

4.0 Device Specifications (Continued)

Reset Input ($\overline{RSTI}$): Active low. System reset. Section 2.3.

Status Lines (ST0–ST3): Status code input from the CPU. Active from T4 of previous bus cycle through T3 of current bus cycle. Section 2.4.

Program Flow Status ($\overline{PFS}$): Active low. Pulse issued by the CPU at the beginning of each instruction.

User/Supervisor Mode (U/$\overline{S}$): This signal is provided by the CPU. It is used by the MMU for protection and for selecting the address space (in dual address space mode only). Section 2.4.

Address Strobe Input ($\overline{ADS}$): Active low. Pulse indicating that a virtual address is present on the bus.

4.1.3 Output Signals

Reset Output/Abort ($\overline{RST}$/$\overline{ABT}$): Active Low. Held active longer than one clock cycle to reset the CPU. Pulsed low during T2 or TMMU to abort the current CPU instruction.

Interrupt Output ($\overline{INT}$): Active low. Pulse used by the debug functions to inform the CPU that a break condition has occurred.

Float Output ($\overline{FLT}$): Active low. Floats the CPU from the bus when the MMU accesses page table entries or performs a physical breakpoint check. Section 2.4.3.

Physical Address Valid ($\overline{PAV}$): Active low. Pulse generated during TMMU indicating that a physical address is present on the bus.

4.2 ABSOLUTE MAXIMUM RATINGS

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7V
Power Dissipation	1.5W

Hold Acknowledge Output ($\overline{HLDAO}$): Active low. When active, indicates that the bus has been released.

4.1.4 Input-Output Signals

Data Direction In ($\overline{DDIN}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Driven by the MMU during a page-table lookup.

Address Translation/Slave Processor Control ($\overline{AT}$/$\overline{SPC}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by the MMU to acknowledge completion of an MMU instruction. Section 2.3 and 2.5. Held low during reset to select the address translation mode on the CPU.

M.S. Bit of Physical Address/High Byte Float (A24/$\overline{HBF}$): Most significant bit of physical address. Sampled on the rising edge of the reset input to select 16 or 32-bit bus mode. This pin outputs a low level if address translation is not enabled. It is floated during T2–T4 if 32-bit bus mode is selected.

Address Bits 16–23 (A16–A23): High order bits of the address bus. These signals are floated by the MMU during T2–T4 if 32-bit bus mode is selected.

Address/Data 0–15 (AD0–AD15): Multiplexed Address/Data Information. Bit 0 is the least significant bit.

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS $T_A = 0$ to +70°C, $V_{CC} = 5V \pm 5\%$, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC} - 0.35$		$V_{CC} + 0.5$	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\ \mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2$ mA			0.45	V
I_{ILS}	$\overline{AT}$/$\overline{SPC}$ Input Current (low)	$V_{IN} = 0.4V$, $\overline{AT}$/$\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}$/$\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current (Output Pins In TRI-STATE Condition)	$0.4 \le V_{OUT} \le V_C$	−20		30	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		200	300	mA

4.0 Device Specifications (Continued)

4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 and PHI2, and 0.8V or 2.0V on all other signals as illustrated in *Figures 4-2* and *4-3*, unless specifically stated otherwise.

ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge
T.E. — trailing edge F.E. — falling edge

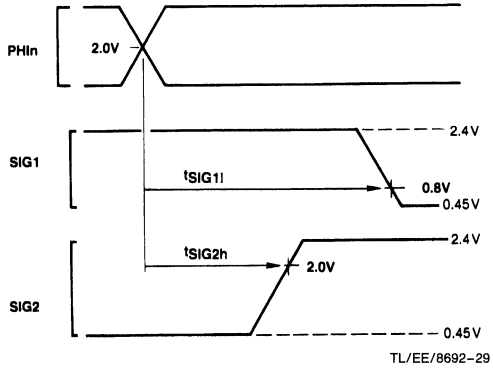


FIGURE 4-2. Timing Specification Standard (Signal Valid after Clock Edge)

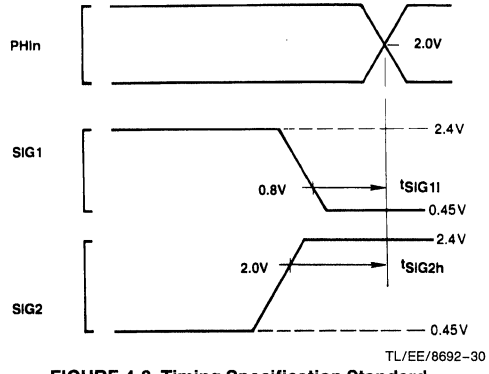


FIGURE 4-3. Timing Specification Standard (Signal Valid before Clock Edge)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32082-6, NS32082-8, NS32082-10.
Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
				Min	Max	Min	Max	Min	Max	
t_{ALv}	4–4	Address Bits 0–15 Valid	After R.E., PHI1 TMMU or T1		65		55		40	ns
t_{ALh}	4–4	Address Bits 0–15 Hold	After R.E., PHI1 T2	5		5		5		ns
t_{AHh}	4–4, 4–6	Address Bits 16–24 Valid	After R.E., PHI1 TMMU or T1		65		55		40	ns
t_{AHh}	4–4	Address Bits 16–24 Hold	After R.E., PHI1 T2	5		5		5		ns
t_{ALPAVs}	4–5	Address Bits 0–15 Set Up	Before $\overline{PAV}$ T.E.	25		25		25		ns
t_{AHPAVs}	4–5	Address Bits 16–24 Set Up	Before $\overline{PAV}$ T.E.	25		25		25		ns
t_{ALPAVh}	4–5	Address Bits 0–15 Hold	After $\overline{PAV}$ T.E.	20		20		15		ns
t_{AHPAVh}	4–5	Address Bits 16–24 Hold	After $\overline{PAV}$ T.E.	20		20		15		ns
t_{ALf}	4–10	AD0–AD15 Floating	After R.E., PHI1 T2		25		25		25	ns
t_{AHf}	4–7, 4–10	A16–A24 Floating	After R.E., PHI1 T2 or T1		25		25		25	ns
t_{ALz}	4–15, 4–16	AD0–AD15 Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 Ti		45		35		25	ns
t_{AHZ}	4–15, 4–16	A16–A24 Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 Ti		45		35		25	ns
t_{ALr}	4–15, 4–16	AD0–AD15 Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 T1		75		65		50	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32082-6, NS32082-8, NS32082-10. (Continued)

Name	Figure	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
				Min	Max	Min	Max	Min	Max	
t_{AHr}	4–15, 4–16	A16–A24 Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 T1		80		65		50	ns
t_{Dv}	4–6	Data Valid (Memory Write)	After R.E., PHI1 T2		80		65		50	ns
t_{Dh}	4–6	Data Hold (Memory Write)	After R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{Df}	4–11	Data Bits Floating (Slave Prccessor Read)	After R.E., PHI1 T1 or Ti		10		10		10	ns
t_{Dv}	4–11	Data Valid (Slave Processor Read)	After R.E., PHI1 T1		80		65		50	ns
t_{Dh}	4–11	Data Hold (Slave Processor Read)	After R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{DDINv}	4–5, 4–7	$\overline{DDIN}$ Signal Valid	After R.E., PHI1 T1 or T_{MMU}		80		65		50	ns
t_{DDINh}	4–5	$\overline{DDIN}$ Signal Hold	After R.E., PHI1 T1 or Ti	0		0		0		ns
t_{DDINf}	4–7	$\overline{DDIN}$ Signal Floating	After R.E., PHI1 T2						25	ns
t_{DDINz}	4–16	$\overline{DDIN}$ Signal Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 Ti		85		70		50	ns
t_{DDINr}	4–16	$\overline{DDIN}$ Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 T1 or Ti		85		70		50	ns
t_{DDINAf}	4–9	$\overline{DDIN}$ Floating after Abort ($\overline{FLT}$ = 0)	After R.E., PHI1 T2		25		25		25	ns
t_{PAVa}	4–4	$\overline{PAV}$ Signal Active	After R.E., PHI1 T_{MMU} or T1		55		45		35	ns
t_{PAVia}	4–4	$\overline{PAV}$ Signal Inactive	After R.E., PHI2 T_{MMU} or T1		60		55		45	ns
t_{PAVw}	4–4	$\overline{PAV}$ Pulse Width	At 0.8V (Both Edges)	50		40		30		ns
t_{PAVdz}	4–14, 4–15	$\overline{PAV}$ Floating Delay	After $\overline{HLDAI}$ F.E.		40		30		25	ns
t_{PAVdr}	4–14, 4–15	$\overline{PAV}$ Return from Floating	After $\overline{HLDAI}$ R.E.		40		30		25	ns
t_{PAVz}	4–16	$\overline{PAV}$ Floating (Caused by $\overline{HOLD}$)	After R.E., PHI2 T4		50		40		30	ns
t_{PAVr}	4–16	$\overline{PAV}$ Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI2 Ti		50		40		30	ns
t_{FLTa}	4–5, 4–10	$\overline{FLT}$ Signal Active	After R.E., PHI1 T_{MMU}		90		75		60	ns
t_{FLTia}	4–7, 4–10	$\overline{FLT}$ Signal Inactive	After R.E., PHI1 T_{MMU}, T_f or T2		55		45		35	ns
t_{ABTa}	4–8, 4–10	Abort Signal Active	After R.E., PHI1 T_{MMU} or T1		90		70		55	ns
t_{ABTia}	4–8, 4–10	Abort Signal Inactive	After R.E., PHI1 T2		80		65		50	ns
t_{ABTw}	4–8, 4–10	Abort Pulse Width	At 0.8V (Both Edges)	110		90		70		ns
t_{INTa}	4–4, 4–10	$\overline{INT}$ Signal Active	After R.E., PHI1 T_{MMU} or T_f						55	ns
t_{INTia}	4–4, 4–10	$\overline{INT}$ Signal Inactive	After R.E., PHI1 T2		80		65		50	ns
t_{INTw}	4–10	$\overline{INT}$ Pulse Width	At 0.8V (Both Edges)	110		90		70		ns
t_{SPCa}	4–13	$\overline{SPC}$ Signal Active	After R.E., PHI1 T1		55		45		35	ns
t_{SPCia}	4–13	$\overline{SPC}$ Signal Inactive	After R.E., PHI1 T4		55		45		35	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32082-6, NS32082-8, NS32082-10. (Continued)

Name	Figure	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
				Min	Max	Min	Max	Min	Max	
t_{SPCf}	4–13	$\overline{SPC}$ Signal Floating	After F.E., PHI1 T4		35		30		25	ns
t_{SPCw}	4–13	$\overline{SPC}$ Pulse Width	At 0.8V (Both Edges)	110		90		70		ns
t_{HLD0da}	4–14	$\overline{HLDA0}$ Assertion Delay	After $\overline{HLDAI}$ F.E.		60		55		50	ns
$t_{HLD0dia}$	4–14, 4–15	$\overline{HLDA0}$ Deassertion Delay	After $\overline{HLDAI}$ R.E.		60		55		50	ns
t_{HLD0a}	4–15, 4–16	$\overline{HLDA0}$ Signal Active	After R.E., PHI1 Ti		50		40		30	ns
t_{HLD0ia}	4–16	$\overline{HLDA0}$ Signal Inactive	After R.E, PHI1 Ti		50		40		30	ns
t_{ATa}	4–18	$\overline{AT}/\overline{SPC}$ Signal Active	After R.E., PHI1		50		40		35	ns
t_{ATia}	4–18	$\overline{AT}/\overline{SPC}$ Signal Inactive	After R.E., PHI1		50		40		35	ns
t_{ATf}	4–18	$\overline{AT}/\overline{SPC}$ Signal Floating	After F.E., PHI1		35		30		25	ns
t_{RST0a}	4–18	$\overline{RST}/\overline{ABT}$ Asserted (Low)	After R.E. PHI1		40		35		30	ns
t_{RST0ia}	4–18	$\overline{RST}/\overline{ABT}$ Deasserted (High)	After R.E. PHI1 Ti		40		35		30	ns

4.4.2.2 Input Signal Requirements: NS32082-6, NS32082-8, NS32082-10

Name	Figure	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
				Min	Max	Min	Max	Min	Max	
t_{DIs}	4–5	Data In Set Up (Memory Read)	Before F.E., PHI2 T3	25		20		15		ns
t_{DIh}	4–5	Data In Hold (Memory Read)	After R.E., PHI1 T4	0		0		0		ns
t_{DIs}	4–11	Data In Set Up (Slave Processor Write)	Before F.E., PHI2 T1	30		25		20		ns
t_{DIh}	4–11	Data In Hold (Slave Processor Write)	After R.E., PHI1 T4	0		0		0		ns
t_{RDYs}	4–5	RDY Signal Set Up	Before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	4–5	RDY Signal Hold	After F.E., PHI1 T3	0		0		0		ns
t_{USs}	4–4, 4-11	U/$\overline{S}$ Signal Set Up	Before F.E., PHI2 T4 or Ti	45		40		35		ns
t_{USh}	4–4, 4–11	U/$\overline{S}$ Signal Hold	After R.E., PHI1 Next T4	0		0		0		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements: NS32082-6, NS32082-8, NS32082-10 (Continued)

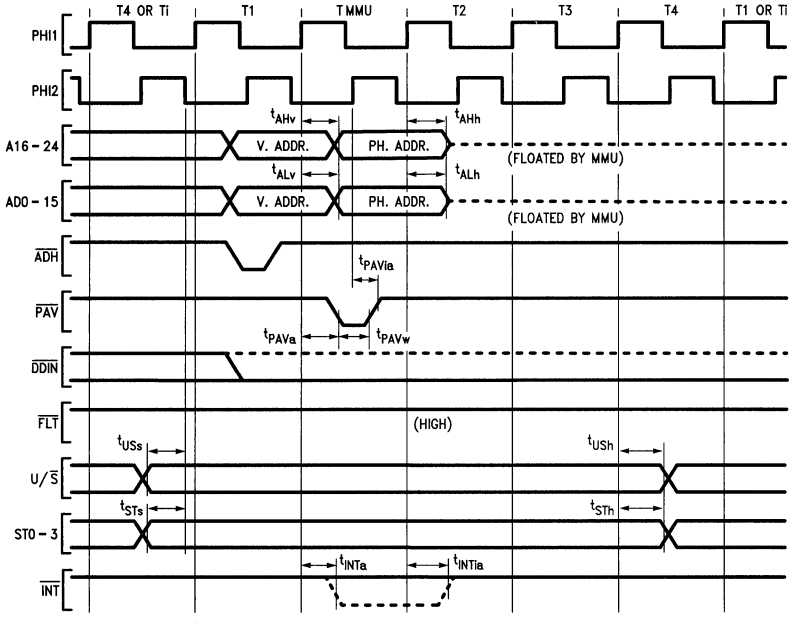
Name	Figure	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
				Min	Max	Min	Max	Min	Max	
t_{STs}	4–4, 4–11	Status Signals Set Up	Before F.E., PHI2 T4 or Ti	70		60		45		ns
t_{STh}	4–4, 4–11	Status Signals Hold	After R.E., PHI1 Next T4	0		0		0		ns
t_{SPCs}	4-11	$\overline{SPC}$ Input Set Up	Before F.E., PHI2 T1	70		55		45		ns
t_{SPCh}	4-11	$\overline{SPC}$ Input Hold	After R.E., PHI1 T4	0		0		0		ns
t_{HLDs}	4-16	$\overline{HOLD}$ Signal Set Up	Before F.E., PHI2 T4 or Ti	25		25		25		ns
t_{HLDh}	4-16	$\overline{HOLD}$ Signal Hold	After F.E., PHI2 T4 or Ti	0		0		0		ns
t_{HLDIs}	4-15	$\overline{HLDAI}$ Signal Set Up	Before F.E., PHI2 Ti	30		25		20		ns
t_{HLDIh}	4-15	$\overline{HLDAI}$ Signal Hold	After F.E., PHI2 Ti	0		0		0		ns
t_{HBFs}	4-18	A24/$\overline{HBF}$ Signal Set Up	Before F.E., PHI2	20		15		10		ns
t_{HBFh}	4-18	A24/$\overline{HBF}$ Signal Hold	After F.E., PHI2	0		0		0		ns
t_{RSTIs}	4-18	Reset Input Set Up	Before F.E., PHI1	20		20		20		ns
t_{PWR}	4-19	Power Stable to $\overline{RSTI}$ R.E.	After V_{CC} Reaches 4.5V	50		50		50		μs
t_{RSTIw}		$\overline{RSTI}$ Pulse Width	At 0.8V (Both Edges)	64		64		64		t_{cp}

4.4.2.3 Clocking Requirements: NS32082-6, NS32082-8, NS32082-10

Name	Figure	Description	Reference/ Conditions	NS32082-6		NS32082-8		NS32082-10		Units
				Min	Max	Min	Max	Min	Max	
t_{CLr}	4–17	PHI1, PHI2 Rise Time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	4–17	PHI1, PHI2 Fall Time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	4–17	Clock Period	R.E., PHI1, PHI2 to Next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4–17	PHI1, PHI2 Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	$0.5t_{Cp}-14$		$0.5t_{Cp}-12$		$0.5t_{Cp}-10$		ns
$t_{CLh(1,2)}$	4–17	PHI1, PHI2 High Time	At V_{CC} − 0.9V on PHI1, PHI2 (Both Edges)	$0.5t_{Cp}-18$		$0.5t_{Cp}-17$		$0.5t_{Cp}-15$		ns
$t_{nOVL(1,2)}$	4–17	Non-overlap Time	0.8V on F.E. PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}		Non-overlap Asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}		PHI1, PHI2 Asymmetry $t_{CLw(1)} - t_{CLw(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

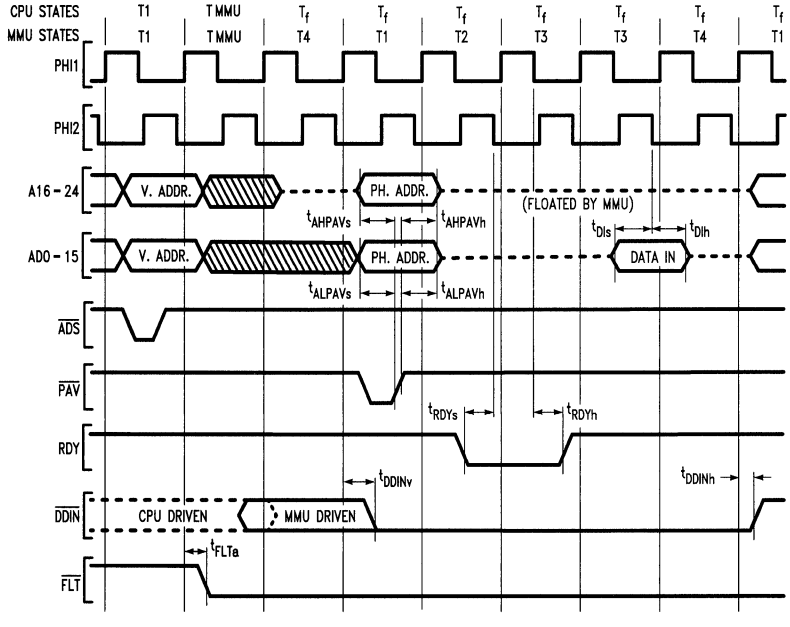
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



TL/EE/8692–31

FIGURE 4-4. CPU Read (Write) Cycle Timing (32-Bit Mode); Translation in TLB

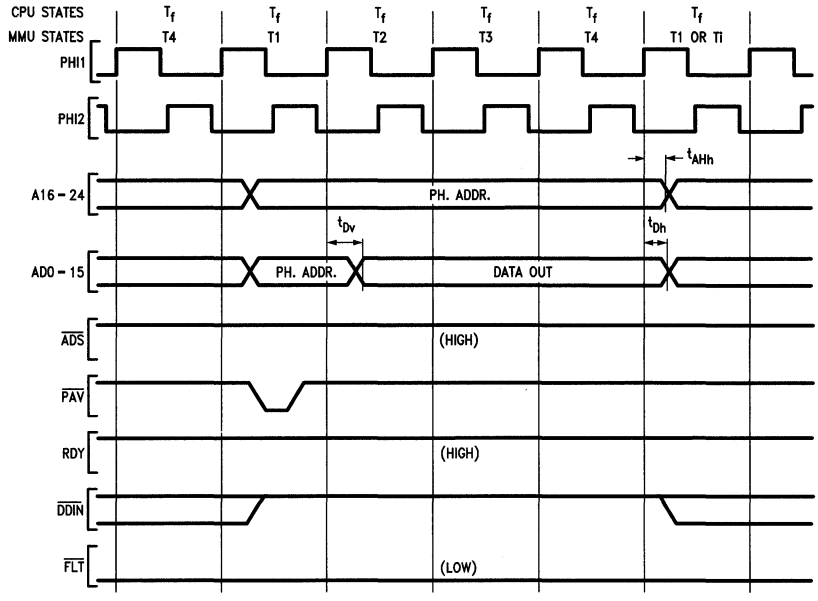


TL/EE/8692–32

FIGURE 4-5. MMU Read Cycle Timing (32-Bit Mode); After a TLB Miss

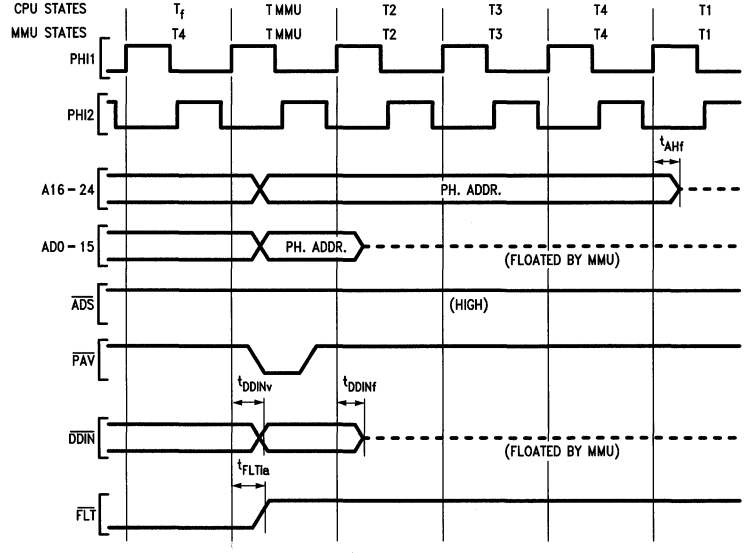
Note: After $\overline{FLT}$ is asserted, $\overline{DDIN}$ may be driven temporarily by both CPU and MMU. This, however, does not cause any conflict, since both CPU and MMU force $\overline{DDIN}$ to the same logic level.

4.0 Device Specifications (Continued)



TL/EE/8692–33

FIGURE 4-6. MMU Write Cycle Timing; after a TLB Miss

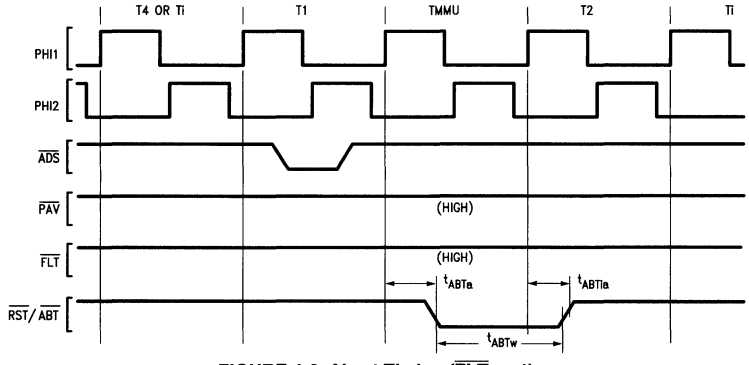


TL/EE/8692–34

FIGURE 4-7. $\overline{\text{FLT}}$ Deassertion Timing

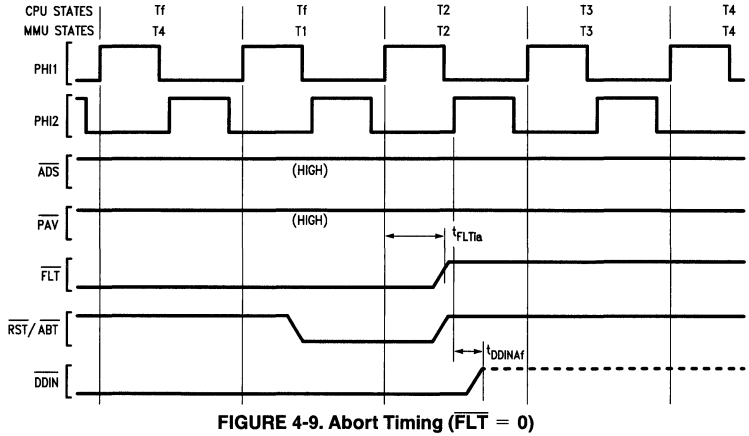
Note: After $\overline{\text{FLT}}$ is deasserted, $\overline{\text{DDIN}}$ may be driven temporarily by both CPU and MMU. This, however, does not cause any conflict. Since CPU and MMU force $\overline{\text{DDIN}}$ to the same logic level.

4.0 Device Specifications (Continued)



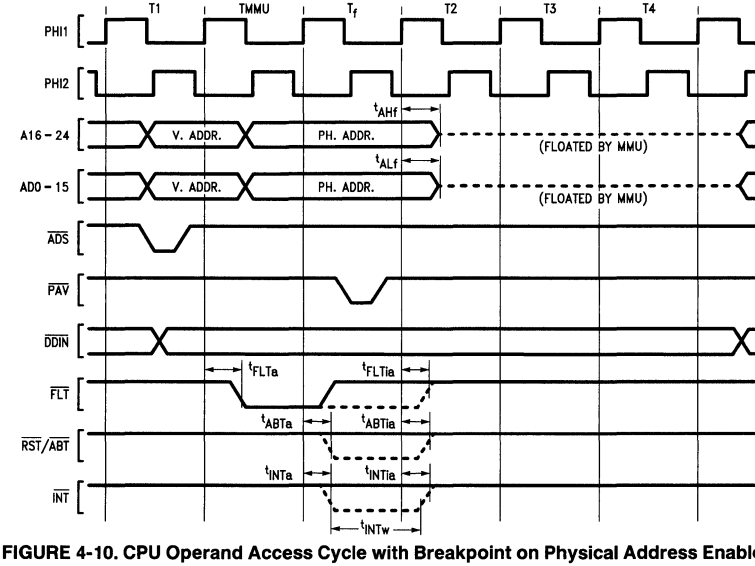
TL/EE/8692-35

FIGURE 4-8. Abort Timing ($\overline{FLT}$ = 1)



TL/EE/8692-36

FIGURE 4-9. Abort Timing ($\overline{FLT}$ = 0)



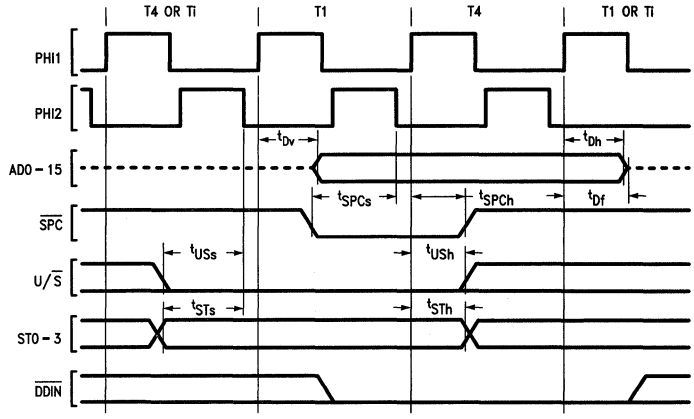
TL/EE/8692-37

FIGURE 4-10. CPU Operand Access Cycle with Breakpoint on Physical Address Enabled

Note: If a breakpoint condition is met and abort on breakpoint is enabled, the bus cycle is aborted. In this case $\overline{FLT}$ is stretched by one clock cycle.

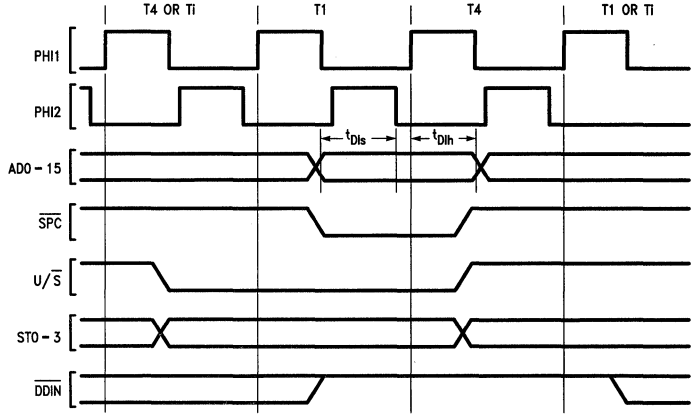
3

4.0 Device Specifications (Continued)



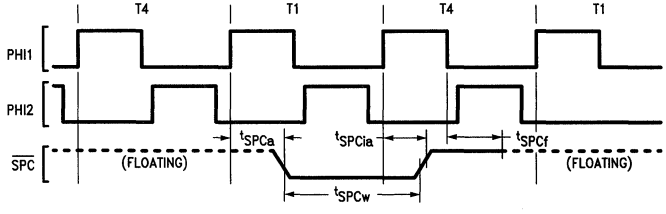
TL/EE/8692-38

FIGURE 4-11. Slave Access Timing; CPU Reading from MMU



TL/EE/8692-39

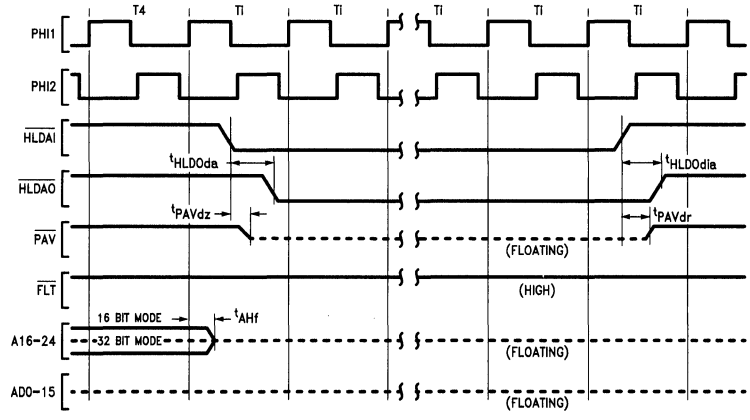
FIGURE 4-12. Slave Access Timing; CPU Writing to MMU



TL/EE/8692-40

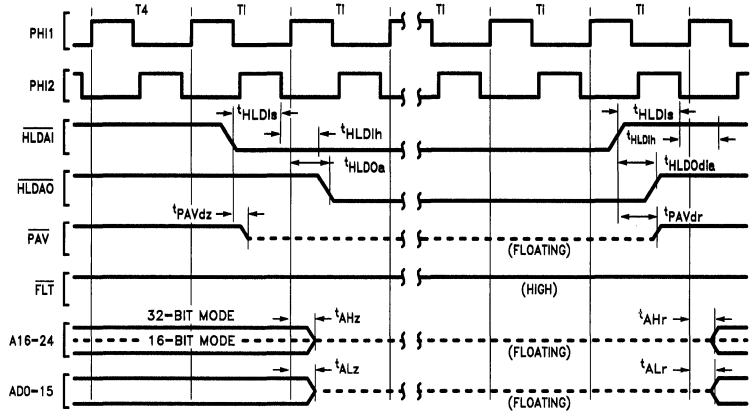
FIGURE 4-13. $\overline{\text{SPC}}$ Pulse from the MMU

4.0 Device Specifications (Continued)



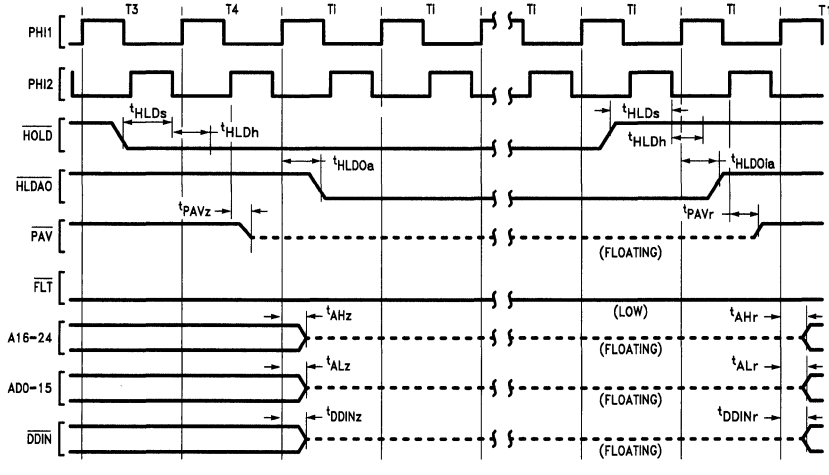
TL/EE/8692-41

FIGURE 4-14. Hold Timing ($\overline{FLT}$ = 1); SMR Instruction Not Being Executed



TL/EE/8692-42

FIGURE 4-15. Hold Timing ($\overline{FLT}$ = 1); SMR Instruction Being Executed

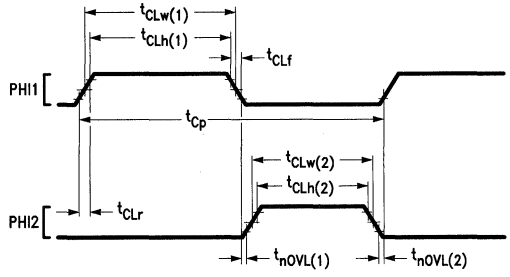


TL/EE/8692-43

FIGURE 4-16. Hold Timing ($\overline{FLT}$ = 0)

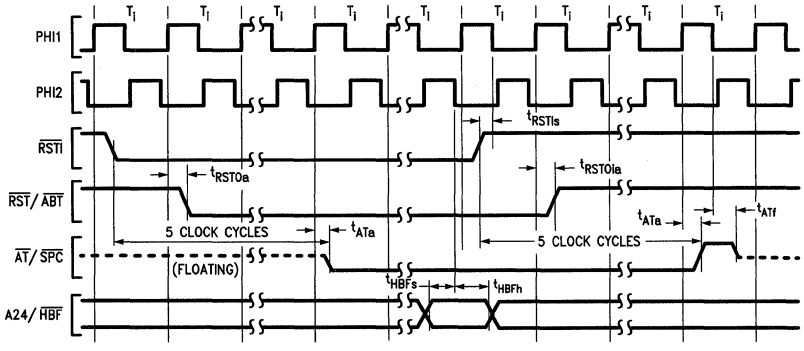
3

4.0 Device Specifications (Continued)



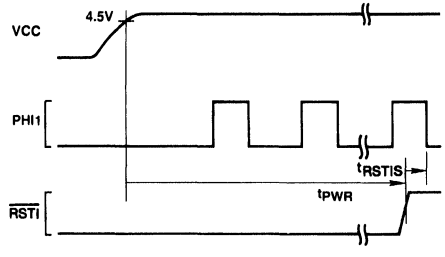
TL/EE/8692–49

FIGURE 4-17. Clock Waveforms



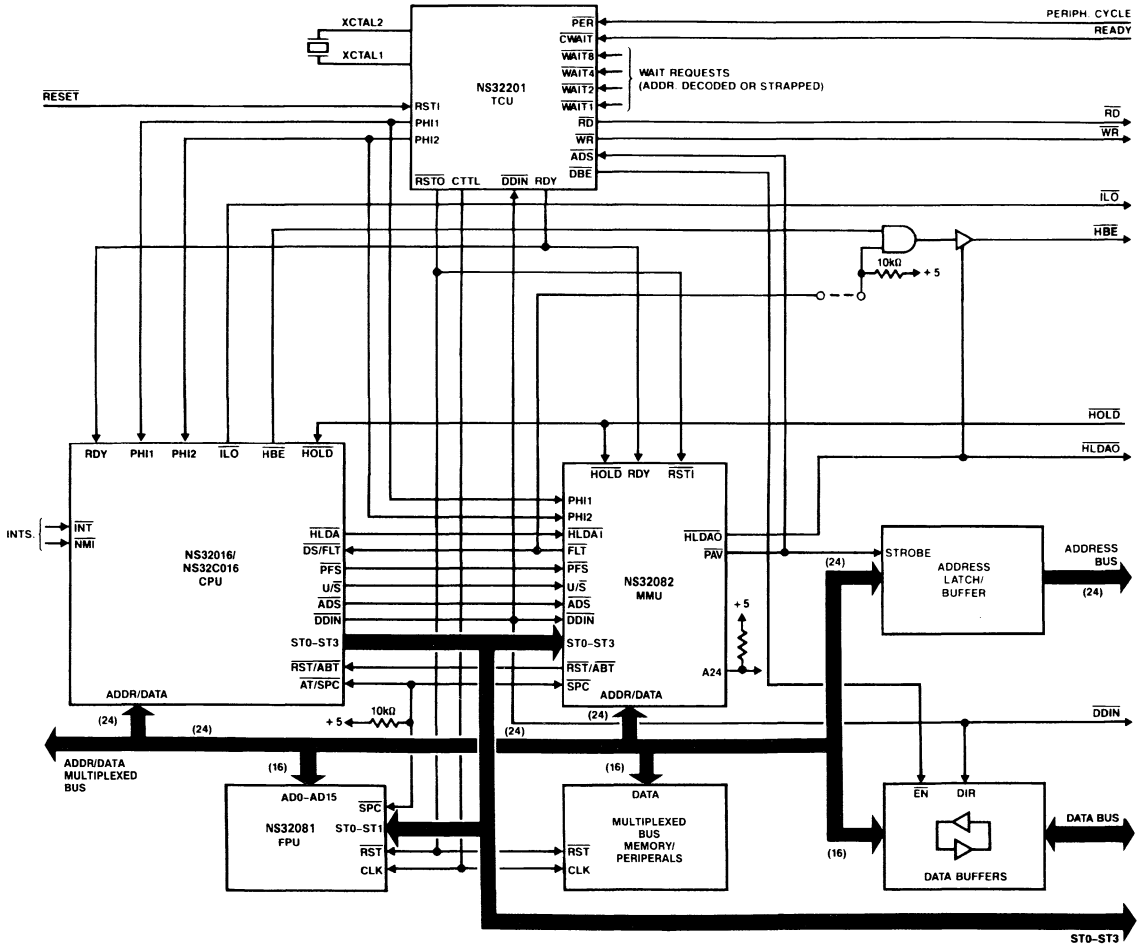
TL/EE/8692–45

FIGURE 4-18. Reset Timing



TL/EE/8692–46

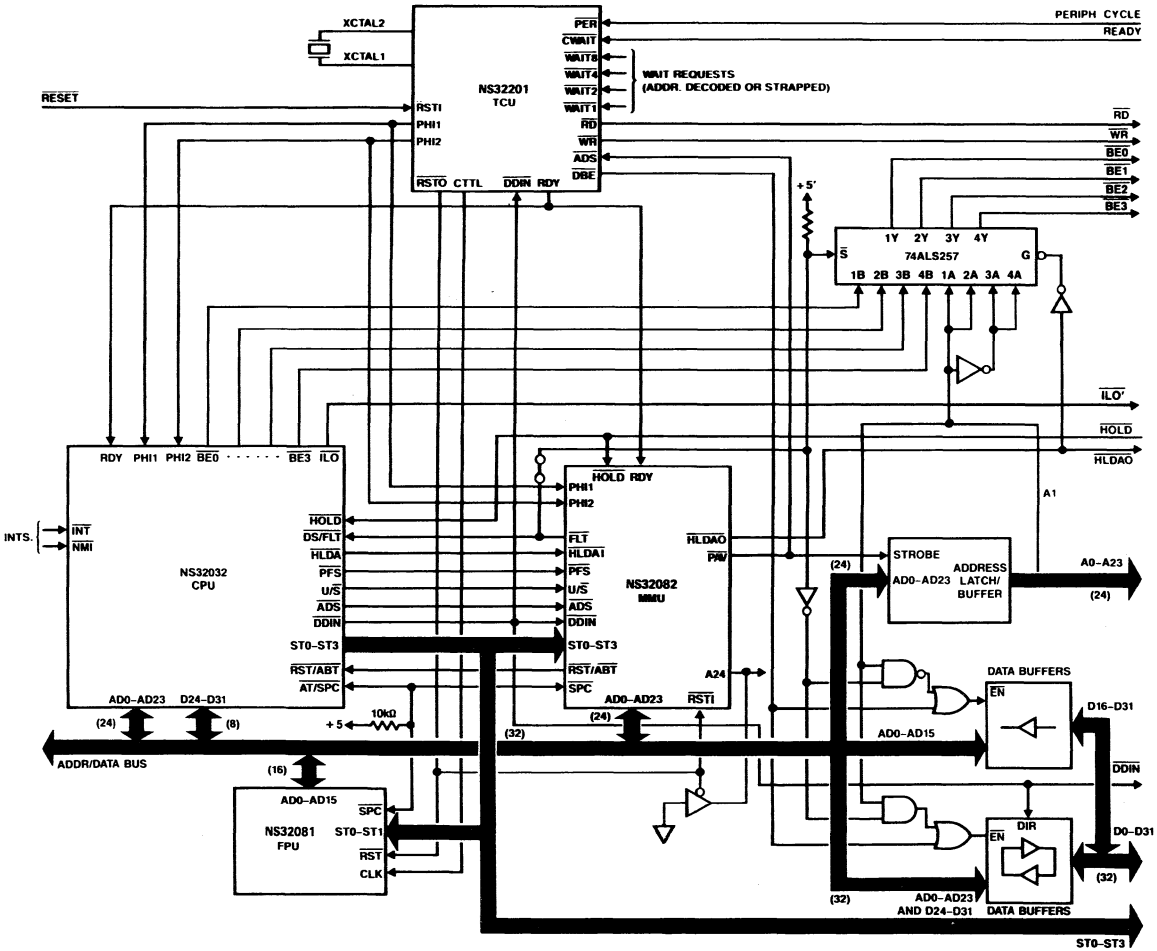
FIGURE 4-19. Power-On Reset



TL/EE/8692-47

Note: The "AND" gate on the $\overline{HBE}$ line is not needed when an NS32016 is used.

FIGURE A-1. System Connection Diagram



TL/EE/8692-48

FIGURE A-2. System Connection Diagram

National Semiconductor Corporation

ADVANCED INFORMATION

NS32382-10/NS32382-15 Memory Management Unit

1.0 General Description

The NS32382 Memory Management Unit (MMU) provides hardware support for demand paged virtual memory management for the NS32332 CPU. The MMU has a 32-bit data path and translates 32-bit virtual addresses from the CPU into 32-bit physical addresses. The physical address is carried on a dedicated 32-bit bus. Memory page size is 4 kbytes.

High-speed address translation is performed on-chip through a Translation Buffer (TB) which holds the address mappings for 32 pages. If the virtual address generated by the CPU has no corresponding entry in the translation buffer, the MMU will perform address translation on the virtual address using a two level page table algorithm.

Protection violations and page faults are automatically detected by the MMU, invoking the instruction abort feature of the CPU. This feature provides a mechanism to operate a demand paged virtual memory.

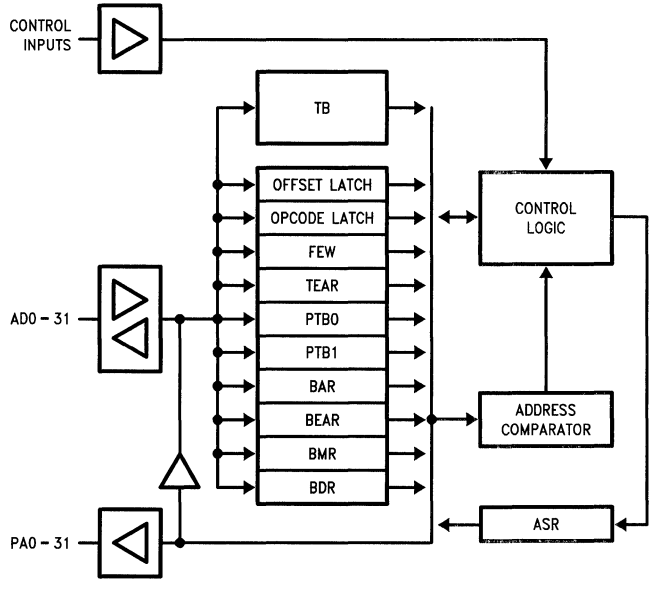
Pages can be declared non-cacheable through the CI bit in the second level Page Table Entry (PTE). This bit is available as an output from the MMU with the same timing as the physical address bits. Multiprocessor support is provided by incorporating features such as the running of interlocked bus cycles when updating the R (reference) and M (modify) bits in the PTE (Page Table Entry).

The architectural interface of the NS32382 MMU has been refined from that of the NS32082 MMU to provide a cleaner programming model with improved performance. The debug features have been simplified; a single virtual address breakpoint is provided.

2.0 Features

- NS32332 CPU compatible (not intended to be used with the NS32016 and NS32032 CPUs)
- 15 MHz maximum operating frequency
- Full hardware support for virtual memory and virtual machines
- 4k page size
- 32-bit virtual and physical addresses
- 32-bit data bus
- Dedicated 32-bit physical address bus
- 32-entry translation buffer
- Security mechanisms implemented via access level checking and dual space mapping
- Non-cacheable page support
- Improved architectural interface
- Dynamic address translation
- Single 5V Supply
- 125-pin PGA package
- High speed XMOS™ process

Block Diagram



TL/EE/8776-1

FIGURE 1

3.0 Pin Descriptions

The following are brief descriptions of the NS32382 pins.

3.1 SUPPLIES

VDD: Multiple pins, connected to the +5V power supply.

GND: Multiple pins, connected to ground.

VBB: 1 pin, output of on-chip substrate voltage generator.

3.2 INPUT SIGNALS

$\overline{\text{ADS}}$: 1 pin, Address Strobe Input, Active Low. This input is pulsed by the CPU during T1; used as a strobe to latch in the virtual address.

$\overline{\text{BER}}$: 1 pin, Bus Error, Active Low. Activation of this signal indicates that a hard error has occurred during the current bus cycle.

$\overline{\text{BRT}}$: 1 pin, Bus Retry, Active Low. Activation of this signal indicates the occurrence of a correctable error or a transient error.

PHI1, PHI2: 2 pins, Clocks, Active High. PHI1 and PHI2 are the two-phase non-overlapping clocks generated by the TCU.

$\overline{\text{HOLD}}$, $\overline{\text{HLDAI}}$: 2 pins, Hold Request and Hold Acknowledge In, Active Low. These signals in conjunction with the $\overline{\text{HLDAO}}$ output are used in DMA transfers. A DMA device requests the bus by asserting the $\overline{\text{HOLD}}$ line low. $\overline{\text{HLDAI}}$ input is connected to the $\overline{\text{HLDA}}$ output from the CPU and $\overline{\text{HLDAO}}$ from the MMU is connected to the DMA device.

RDY: 1 pin, Ready, Active High. Used by slower devices to extend the access cycle. RDY is sampled by the falling edge of PHI1 in T3.

$\overline{\text{RES}}$: 1 pin, Reset, Active Low. System reset.

ST0–3: 4 pins, System Status. These 4 status lines are generated by the CPU. They are activated one clock cycle before T1 and stay valid until the end of T3 of the current access.

$\overline{\text{MC}}$: 1 pin, Multiple Cycle, Active Low. The value of this pin will be loaded into the appropriate MC bit in the ASR when ASR is clocked.

$\overline{\text{SPC}}$: 1 pin, Slave Processor Control, Active Low. This signal is used by the CPU to strobe the slaves during a slave instruction. For more details, refer to the NS32332 CPU data sheet.

U/S: 1 pin, User/Supervisor Mode Indicator. This CPU generated signal reflects the state of the U bit (low = supervisor, high = user) inside the Program Status Register (PSR). It is used by the MMU to implement memory protection as well as the selection of address spaces in the dual space mode.

3.3 OUTPUT SIGNALS

CI: 1 pin, Cache Inhibit, Active High. The CI bit in the PTE is stored in the translation buffer (TB) with the upper physical address bits. This bit is available with the same timing as the physical address. During MMU generated bus cycles, the CI pin is held low. When the MMU is in No-Translation mode, the CI pin will be held low.

$\overline{\text{DONE}}$: 1 pin, Slave Done, Active Low. This signal is generated by the MMU to inform the CPU that the slave activities which the CPU requested have been completed. $\overline{\text{DONE}}$ is in TRI-STATE® when it is not active.

$\overline{\text{FLT}}$: 1 pin, Float, Active Low. $\overline{\text{FLT}}$ is used to float the CPU off the bus when the MMU has to access the page table entries in memory. It is sampled by the CPU during TMMU. During reset, this signal is driven high.

$\overline{\text{HLDAO}}$: 1 pin, Hold Acknowledge Out, Active Low. This signal in conjunction with the $\overline{\text{HOLD}}$ and $\overline{\text{HLDAI}}$ inputs is used in DMA operations.

$\overline{\text{MADS}}$: 1 pin, MMU Address Strobe, Active Low. This signal is asserted in T1 of an MMU initiated cycle. It is used to inform users that physical address is available on the physical address bus. $\overline{\text{MADS}}$ has the same timing as $\overline{\text{ADS}}$ from the CPU. (A separate pin is needed because $\overline{\text{ADS}}$ is not in a TRI-STATE condition during float.) $\overline{\text{MADS}}$ will be TRI-STATE during Hold Acknowledge.

$\overline{\text{MILO}}$: 1 pin, MMU Interlock, Active Low. This signal is asserted by the MMU when it performs a read-modify-write operation to update the R (reference) and/or the M (modify) bit in the Page Table Entry (PTE). $\overline{\text{MILO}}$ has the same timing as the CPU's NILO signal. $\overline{\text{MILO}}$ will be deactivated (high) during Hold Acknowledge.

PA0–PA31: 32 pins, Physical Address Bus, Active High. The physical addresses will be in a TRI-STATE condition during hold acknowledge.

$\overline{\text{PAV}}$: 1 pin, Physical Address Valid, Active Low. A pulse on the $\overline{\text{PAV}}$ line during TMMU indicates that the physical address presented by PA0–PA31 is valid. This signal will not be generated if an abort occurs. $\overline{\text{PAV}}$ will be in a TRI-STATE mode during hold acknowledge.

$\overline{\text{RST}}$/$\overline{\text{ABT}}$: 1 pin, Reset or Abort, Active Low. This is a dual function pin. When this line is pulsed during T2/TMMU of a CPU initiated access, the CPU will be aborted. Abort information is held in the MMU Abort Status Register (ASR). When the $\overline{\text{RES}}$ (reset) input is activated, the MMU will respond by holding the $\overline{\text{RST}}$/$\overline{\text{ABT}}$ output low. This causes the CPU to reset since a $\overline{\text{RST}}$/$\overline{\text{ABT}}$ signal held active longer than one clock cycle is interpreted by the CPU as a valid reset.

3.4 INPUT-OUTPUT SIGNALS

AD0–AD31: 32 pins, Multiplexed Address/Data Bus, Active High. During T1 of a CPU access, these lines carry the virtual address. During T2, T3, and T4, the bus contains data. When $\overline{\text{FLT}}$ is active, the CPU puts AD0–AD31 into a TRI-STATE condition and their signals are driven by the MMU.

$\overline{\text{DDIN}}$: 1 pin, Data Direction Indicator. This signals an input to the MMU during CPU cycles: low during read and high during write. In MMU cycles, the $\overline{\text{DDIN}}$ signal is driven by the MMU to indicate the type of access: low for read and high for write. $\overline{\text{DDIN}}$ will be in a TRI-STATE condition during Hold Acknowledge.

4.0 Functional Description

4.1 MMU REGISTERS

The NS32382 MMU provides the following interface registers:

FEW	Feature Enable Word, 32 bits
ASR	Abort Status Register, 32 bits
TEAR	Translation Exception Addr Reg, 32 bits
BEAR	Bus Error Address Register, 32 bits
PTB0	Page Table Base Register 0, 32 bits
PTB1	Page Table Base Register 1, 32 bits
IVAR0	Invalidate Virtual Addr Register 0, 32 bits
IVAR1	Invalidate Virtual Addr Register 1, 32 bits
BAR	Breakpoint Address Register, 32 bits
BMR	Breakpoint Mask Register, 32 bits
BDR	Breakpoint Data Register, 32 bits

4.0 Functional Description (Continued)

All registers except IVAR0 and IVAR1 can be read by the SMR instruction. IVAR0 and IVAR1 are write-only pseudo-registers. All registers except TEAR, BEAR, and BDR can be loaded by the LMR instruction. TEAR, BEAR, and BDR are read-only registers. Writing to a read-only register will have no effect on the MMU. However, reading a write-only register should be avoided since random data patterns may be returned.

4.1.1 Feature Enable Word (FEW)

The Feature Enable Word (FEW) controls the different features provided by the MMU. The FEW is 32 bits in length and has the format shown in *Figure 2*. All bits inside the FEW will be cleared on reset. Bits 8 to 31 are RESERVED for future use. When FEW is loaded, bits 8 to 31 must contain zeros. The control bits in the FEW are described below:

TU: TU is the Tranlsate User bit. If TU is 1, MMU will translate all addresses specified in the User mode. If TU is 0, MMU will interpret addresses specified in the User mode as physical addresses.

TS: TS is the Translate Supervisor bit. If TS is 1, MMU will translate all addresses specified in the Supervisor mode. If TS is 0, MMU will interpret addresses specified in the Supervisor mode as physical addresses.

DS: DS is the dual space bit. If DS is 1, then PTB1 contains the first level Page Table Base address of all addresses specified in the User mode and PTB0 contains the first level Page Table Base address of all addresses specified in the Supervisor mode. If DS is 0, then PTB0 contains the first level Page Table Base address of all addresses specified in both User and Supervisor modes.

AO: AO is the access override bit. If AO is 1, MMU will override the protection level of all addresses. This permits a program to access memory which is normally accessible only to the Supervisor while the system is in the User mode.

BR, BW, BX, BS: These are the four Breakpoint Address Enable (BAE) bits.

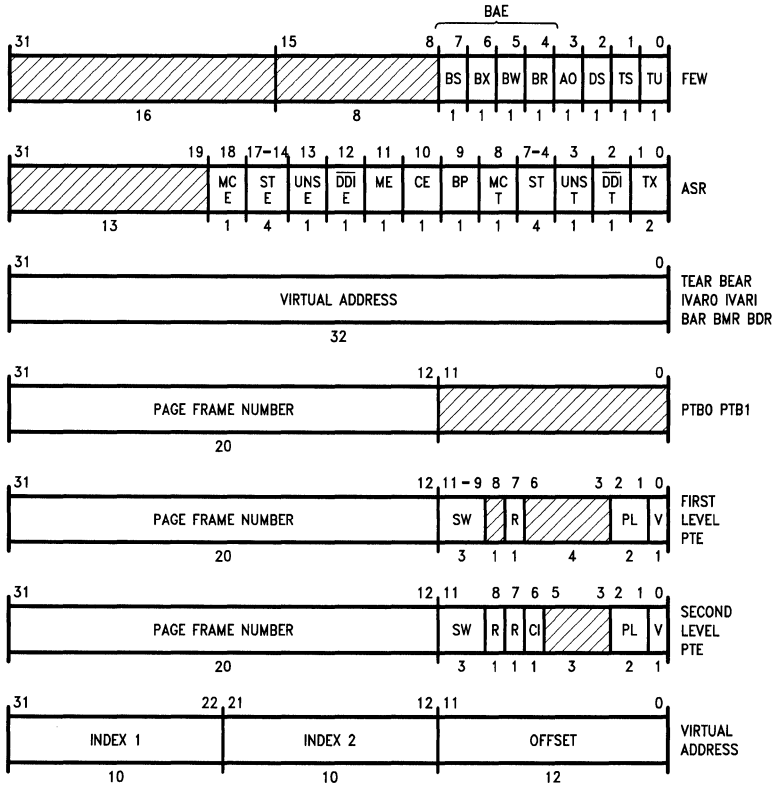
BR is the breakpoint enable bit for operand read operations. BR = 1 enables the address comparison logic for operand reads, effective address reads, and the dummy read before read-modify-write.

BW is the breakpoint enable bit for operand write operations. BW = 1 enables the address comparison logic for operand writes.

BX is the breakpoint enable bit for instruction read operations. BX = 1 enables the address comparison logic for instruction reads.

BS is the address space select bit for the Compare Feature. BS = 0 selects address space 0 (PTB0). BS = 1 selects address space 1 (PTB1).

The Address Compare feature will be disabled if BR, BW and BX are cleared (i.e., BR = BW = BX = 0).



TL/EE/8776-2

FIGURE 2. MMU Register Assignments

4.0 Functional Description (Continued)

4.1.2 Abort Status Register (ASR)

The Abort Status Register (ASR) contains the status of the MMU at the time a translation exception or a bus error was reported to the CPU. When a bus error is detected, the bus error (E) bits in the ASR will be clocked. In case of a translation exception, the translation exception (T) bits in the ASR will be updated. CPU error is recognized when a bus error occurs while CPU is controlling the bus. MMU error is recognized when a bus error occurs while MMU is controlling the bus. If a successful address comparison is found, only AB0 in the ASR will set, all other bits are not modified. The ASR is loadable via the LMR instruction.

ASR is 32 bits in length and has the format shown in *Figure 2.* Bits 19 to 31 are RESERVED for future use. When the ASR is read, bits 19 to 31 will be returned as zeros. All bits in the ASR are cleared to zero upon reset.

TX0–1: These are the two Translation Exception bits. They specify the cause of the current address translation exception.

TX1	TX0	
0	0	No Translation Exception
0	1	First Level PTE Invalid
1	0	Second Level PTE Invalid
1	1	Protection Exception

$\overline{\text{DDI}}$ (E): $\overline{\text{DDI}}$ (E) is the data direction when a bus error is recognized. If $\overline{\text{DDI}}$ = 0, a read operation or the first part of a read-modify-write was in progress when a bus error is detected. If $\overline{\text{DDI}}$ = 1, a write or the last part of a read-modify-write is in progress when a bus error is detected.

UNS (E): This is the state of the User/Supervisor pin from the CPU when the bus error is recognized.

ST0–3 (E): These four bits represent the state of the status bits from the CPU when the bus error is recognized.

MC (E): This is the Multiple Cycle signal from CPU when the bus error is recognized.

The next three bits in the ASR are the Abort Indicators:

BP: This is the breakpoint indicator bit. BP is set to 1 when a breakpoint is detected.

CE: This is the CPU error indicator bit. CE is set to 1 when a CPU error is detected.

ME: This is the MMU error indicator bit. ME is set to 1 when an MMU error is detected.

$\overline{\text{DDI}}$ (T): $\overline{\text{DDI}}$ (T) is the data direction when a translation exception is recognized. If $\overline{\text{DDI}}$ = 0, a read operation or the first part of a read-modify-write was in progress when the exception is detected. If $\overline{\text{DDI}}$ = 1, a write or the last part of a read-modify-write was in progress when the exception is detected.

UNS (T): This is the state of the User/Supervisor pin from the CPU when a translation exception is recognized.

ST0–3 (T): These four bits represent the state of the status bits from the CPU when a translation exception is recognized.

MC (T): This is the Multiple Cycle signal from the CPU when a translation exception is recognized.

4.1.3 Translation Exception Address Register (TEAR)

The TEAR is clocked when an abort due to a translation exception occurs. The format of the TEAR is shown in *Figure 2.* This register contains the 32-bit virtual address which caused the translation exception. TEAR is a read only register.

4.1.4 Bus Error Address Register for Bus Error (BEAR)

The BEAR is clocked when a CPU or MMU error occurs. The format of the BEAR is shown in *Figure 2.* This register contains the 32-bit virtual address which triggered the bus error. BEAR is a read only register.

4.1.5 Page Table Base Registers (PTB0, PTB1)

PTB0 and PTB1 are 32-bit registers. Their format is shown in *Figure 2.* Bits 0–11 of the PTB registers are RESERVED and will be returned as zeros upon read. The PFN field specifies the base address of the first table used by the MMU during address translation.

The current mode of system operation (User or Supervisor) and the Dual Space bit (DS) in the Feature Enable Word (FEW) specify which register (PTB0 or PTB1) will be used for address translation. If DS = 0, PTB0 will be used in either User or Supervisor mode. If DS = 1, PTB0 will be used in Supervisor mode and PTB1 will be used in User mode.

When PTB0 is loaded by the CPU, all entries in the Translation Buffer (TB) with AS = 0 are invalidated. Loading PTB1 invalidates entries with AS = 1. In this way, user entries in the Translation Buffer are automatically flushed on a context switch.

4.1.6 Invalidate Virtual Address Registers (IVAR0, IVAR1)

The Invalidate Address Registers are 32-bit write-only pseudo-registers. They are used to remove invalid Page Table Entries (PTEs) from the MMU's Translation Buffer (TB). These registers do not physically exist in hardware but a write to IVAR0 or IVAR1 will cause the execution of an address invalidation operation.

The translation buffer contains 20-bit physical page numbers and 20-bit virtual address tags of the 32 most recently used pages. A virtual address written into IVARx causes the MMU to invalidate the one entry, if that entry exists, in the TB which corresponds to the virtual address. IVAR0 is used to invalidate entries with AS = 0 while IVAR1 is used to invalidate entries with AS = 1.

Page Table Entries (PTEs) in the translation buffer must be invalidated whenever the corresponding entries in the page tables are modified. Entries can also be invalidated by loading the PTBx registers.

4.1.7 Breakpoint Address Register (BAR)

The Breakpoint Address Register (BAR) is 32 bits in length. Its format is shown in *Figure 2.* BAR is used to hold a virtual address for the purpose of address comparison during instruction and operand accesses.

4.1.8 Breakpoint Mask Register (BMR)

The Breakpoint Mask Register is 32 bits in length. Its format is shown in *Figure 2.* The content of BMR indicates which bit positions of the virtual address are to be compared when the Compare Function is enabled. Bits which are set are used for matching while bits which are cleared become "don't cares." This feature allows an abort to be generated upon an access to any location within a block of addresses.

4.1.9 Breakpoint Data Register (BDR)

The Breakpoint Data Register is 32 bits in length. Its format is shown in *Figure 2.* The BDR contains the virtual ad-

4.0 Functional Description (Continued)

dress on the multiplexed address/data bus from the CPU when a breakpoint is detected. BDR is a read only register.

4.2 TRANSLATION BUFFER

The Translation Buffer (TB) contains second level Page Table Entries (PTEs) of the 32 most recently used pages. Direct mapping from virtual to physical address for locations inside those pages are immediately available without going through the address translation process.

The 32-entry Translation Buffer is a Content-Addressable-Memory (CAM). The virtual page frame number represented by the upper 20 bits of the CPU address and the Address Space Bit (AS) are compared against entries in the buffer. If a match is found, the mapped physical address in the selected entry will be presented to the Physical Address Bus immediately. If a match is not found, the control block will be notified to perform address translation using page tables in memory and the new mapping will be stored into an entry selected by a sequential replacement scheme.

Each entry in the TB contains a 20-bit Virtual Address Tag (VA12–VA31), an address space bit (AS), a 20-bit Physical Page Number (PA12–PA31), a Presence bit (P) which indicates the validity of the entry, and the following PTE bits: CI, M, and PL0–1.

The CI (Cache Inhibit) bit determines whether the page associated with an entry is cacheable while the M (Modify) indicates whether the page has been modified.

The protection level field (PL0–1) contains the most restrictive combination of the level 1 and level 2 PTEs. The M bit and the PL0–1 bits are used by the MMU to implement the address translation and error handling algorithms described in the Address Translation Section.

4.3 FUNCTIONAL DESCRIPTION

4.3.1 Address Translation

In T1 of each access, the CPU presents a virtual address on the A/D bus. MMU strobes in the virtual address using the ADS signal from the CPU and checks the Translation Buffer (TB) entries for possible match. The 20-bit Virtual Address Tag and the AS bit in each entry are compared with virtual address bits VA12–31 and the current address space respectively. A TB miss will occur in the following cases:

1. No match is found after the Virtual Address Tag and the AS bit in each entry have been compared with virtual address bits VA12–31 and the current address space.
2. The Virtual Address Tag and the AS bit of an entry matches virtual address bits VA12–31 and the current address space. However, the Presence bit (P) of the entry is not set.
3. A match is found, but the current access is a write to an un-modified page (M = 0 and CPU access = write).

If a miss occurs in the TB, the MMU translates the address by referencing page tables contained in memory. The virtual address is partitioned into three (3) fields: INDEX1, INDEX2, and OFFSET.

The contents of the Page Table Base (PTBx) register point to the base of the first level page table. The INDEX1 field is used to index into the first level page table and select the first level Page Table Entry (PTE). The Page Frame Number field (PFN) inside this PTE points to the base of the second level page table. The INDEX2 field of the virtual address is used to index into the second level page table and select the second level Page Table Entry. The PFN field in the second level PTE is concatenated with the OFFSET field in the virtual address to form the physical address. *Figure 3* illustrates the translation process.

4.3.2 Page Table Entry Format

The Page Table Entry (PTE) format is shown in *Figure 2.* The individual fields in the entry are defined below:

V: The V bit is the Valid bit. If V = 1, the corresponding page is resident in memory. If V = 0, any reference to that page will cause the MMU to generate an abort to the CPU. When the V bit in a PTE is set to zero, it indicates that the page associated with that PTE is not resident in memory. However, the protection bits and the CI bit in the PTE are still valid.

PL0–1: These are protection bits. The PL field is used to control access to a page. The PL bits in the selected entry are checked during each access. The protection levels for both user and supervisor modes are shown in Table I.

R: R is the Reference bit. R is set to 1 whenever the page associated with the PTE is referenced.

M: M is the Modified bit. The M bit is used only in second level PTEs. M is set to 1 whenever the page associated with the PTE is modified. M and R bits are located in different bytes of the PTE to prevent PTE inconsistency in multiprocessor environments where pages tables are shared.

CI: Cache Inhibit bit is used only in second level PTEs. If a page is non-cacheable, the CI bit in its associated PTE will be set to 1. The CI bit is stored in the translation buffer and driven onto the CI pin with the same timing as the Page Frame Number of the physical address. In all MMU initiated bus accesses, the CI pin will be driven inactive (low).

SW0–2: These are three software bits. They are reserved for the operating system and will not be modified by the MMU.

PFN: PFN is the 20-bit Page Frame Number. It contains the higher address bits of the physical address.

BITS 3, 4, 5: Reserved for future use.

4.4 ADDRESS COMPARISON

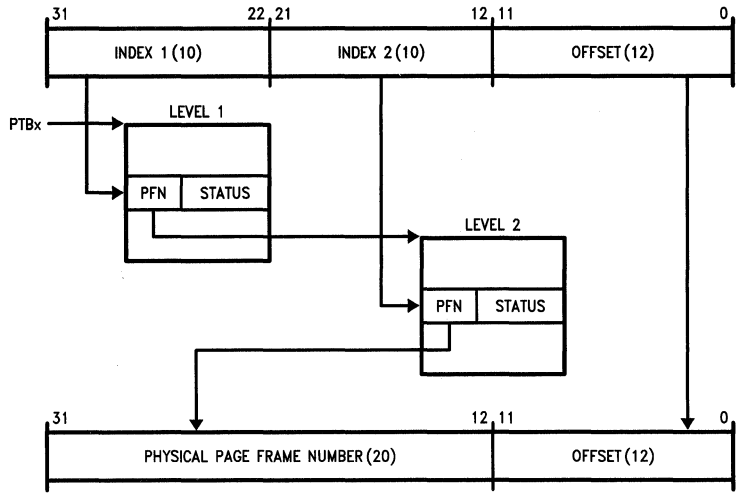
One 32-bit Address Comparison Register is provided. Address comparison is controlled by 4 enable bits (BAE0–3) in the Feature Enable Word (FEW).

When address comparison is enabled in the MMU, the Burst Feature in the NS32332 CPU should be turned off. The Burst Feature must be disabled because of two reasons: first, the CPU is capable of starting and ending a burst cycle anywhere inside a 16 byte boundary and second, addresses are not incremented during burst. It is therefore

TABLE I. Protection Levels

Protection Level	00	01	10	11
User	no access	no access	read only	full access
Supervisor	read only	full access	full access	full access

4.0 Functional Description (Continued)



TL/EE/8776-3

FIGURE 3. Address Translation

possible to skip over the 32-bit address specified in the Address Compare Register during a burst cycle.

The MMU offers users a high degree of flexibility in specifying address comparison conditions.

The following selections are provided through the 4 address compare control bits in the FEW:

BR, BW, BX, BS are the four Breakpoint Address Enable (BAE) bits.

BR is the breakpoint enable bit an operand read operations. BR = 1 enables the address comparison logic for operand reads, effective address reads, and the dummy read before read-modify-write.

BW is the breakpoint enable bit for operand write operations. BW = 1 enables the address comparison logic for operand writes.

BX is the breakpoint enable bit for instruction read operations (execution). BX = 1 enables the address comparison logic for instruction reads.

BS is the address space select bit for the Compare Feature. BS = 0 selects address space 0 (PTB0). BS = 1 selects address space 1 (PTB1).

The Address Compare feature will be disabled if BR, BW and BX are cleared (i.e., BR = BW = BX = 0).

The full 32-bit address specified in the BAR will be compared with the virtual address from the CPU. However, selected bits can be masked by the pattern stored in the Breakpoint Mask Register (BMR). Only those bit positions which are set in the BMR will be used in the comparison process, the other bit positions become "don't cares."

If an address match is detected, an abort will be issued to the CPU and the BP bit of the Abort Status Register will be set. The virtual address on the multiplexed address/data bus from the CPU will be latched into the Breakpoint Data Register (BDR).

If instruction address comparison is enabled, users should align the instruction address they wish to trap on double word boundaries since the NS32332 CPU fetches instructions on double word boundaries.

4.5 BUS OPERATION

The MMU supports the bus timing of the NS32332 CPU. Refer to the NS32332 specification for details. In particular, the MMU supports the READY timing, Bus Error, and Bus Retry features of the CPU.

4.5.1 Clock

The two phase non-overlap clocks PHI1 and PHI2 generated by the NS32301 TCU are connected to both CPU and MMU for bus timing.

4.5.2 Reset

In a typical system, the reset signal generated by system logic will be first synchronized by the TCU before it is fed to the $\overline{RES}$ input of the MMU. The MMU passes the reset signal to the CPU via its $\overline{RST}/\overline{ABT}$ output. When $\overline{RES}$ activates, MMU resets its internal states and asserts $\overline{RST}$ (low) until $\overline{RES}$ is released (high).

4.5.3 Address Translation Bus Timing

The NS32382 MMU bus timing matches the NS32332 CPU bus timing. All bus accesses contain four basic time states: T1—T2—T3—T4. The READY line is sampled in mid T3. If the sampled value is low, an additional time state will be inserted by repeating T3. During read accesses, data must be available to the MMU at the falling edge of PHI2 in T3.

In T1, the CPU places the virtual address on AD0–AD31 and strobes $\overline{ADS}$. Virtual address bits 0–11 are latched by the MMU on $\overline{ADS}$ and immediately placed onto bits 0–11 of the Physical Address Bus (PA0–11). These 12 address bits are the physical byte off-set inside the 4k page and will not be translated by the MMU. The rest of the virtual address (VA12–31) is used by the MMU to generate physical address bits PA12–31. At the beginning of T2, the virtual address lines from the CPU are deactivated.

4.0 Functional Description (Continued)

The translation buffer (TB) is implemented as a Content-Addressable Memory (CAM). Virtual address bits VA12–31 from the CPU and the current address space are compared with the virtual address tag field and the AS bit in each entry of the buffer. If a match is found and the Presence (P) bit in that entry is set, the 20-bit Physical Page Frame Number in the buffer entry will be placed onto bits 12–31 of the Physical Address Bus (PA12–31) and $\overline{PAV}$ will be activated. The Cache Inhibit bit (CI) in the buffer entry will be driven onto the CI pin with the same timing as PA12–31.

If a TB miss occurs, $\overline{PAV}$ will stay inactive and the $\overline{FLT}$ line will be asserted (low) in T2 of the CPU cycle to float the CPU off the bus. Before floating the CPU, the $\overline{DDIN}$ signal is latched by the MMU. This signal will be used when the MMU restarts the original CPU access. After gaining control of the bus, the MMU executes interlocked memory cycles to fetch the first and second level page table entries needed to perform the translation algorithm. All MMU accesses contain four time states: T1—T2—T3—T4.

In T1 of an MMU generated access the MMU puts the physical address on PA0–PA31 and strobes $\overline{MADS}$. $\overline{PAV}$ is pulsed in T2 with the same timing as in the case of a TB hit. Addresses are put out in T1 because some cache designs take advantage of the lower 12 address bits available in the T1 time state of a CPU access to perform directory search and qualify their match logic with PAV in T2. By strobing $\overline{MADS}$ in T1 and $\overline{PAV}$ in T2, MMU accesses would appear identical to CPU accesses.

After the MMU has completed the address translation process, it restarts the original bus cycle by putting the physical address on PA0–PA31 in T1 and pulses $\overline{MADS}$. $\overline{PAV}$ is then strobed in T2 with the same timing as in the case of a TB hit. The latched CPU $\overline{DDIN}$ value is driven onto the $\overline{DDIN}$ line and will remain driven until the rising edge of PHI1 in T3. $\overline{FLT}$ is released in T2, allowing the memory access initiated originally by the CPU to resume from the point where it was stopped.

Cycle extension in both CPU and MMU accesses are achieved via the READY signal. READY is generated by the TCU and sampled by the processors at the falling edge of PHI1 in T3. If READY is sampled low, both processors will insert one wait state into the access by repeating T3. READY is tested in each wait state and as long as READY remains low, additional cycles will be added. A sampled high on READY breaks the wait loop and the processors proceed to T4 at the next clock.

4.5.4 Interlocked Bus Transfers

The NS32382 MMU is capable of executing interlocked cycles to access a stream of data from memory without intervention from other devices. NS32382 MMU executes interlocked Read-Modify-Write memory cycles to access Page Table Entries (PTEs) and update the Reference (R) and Modify (M) bit in the PTEs when necessary. During interlock access cycles, the $\overline{MILO}$ signal from the MMU will be asserted. $\overline{MILO}$ is asserted in the clock cycle immediately before the Read-Modify-Write access and deactivated in the clock cycle following T4 of the write cycle. If a Bus Error is detected in T4 of the write cycle. If a Bus Error is detected in T4 of the read or the write portion of an interlocked access, the interlocked access will be terminated and $\overline{MILO}$ will be deactivated in the clock cycle following the T4 state where Bus Error is recognized.

4.5.5 Bus Retry

The Bus Retry signal ($\overline{BRT}$) will be asserted on the bus when a soft, or correctable error occurs. If the MMU gets a Bus Retry when it is controlling the bus, it will re-run the bus cycle until $\overline{BRT}$ is deactivated.

If Retry is to occur, $\overline{BRT}$ must be asserted (low) by the falling edge of PHI1 in T3. This signal is sampled again at the falling edge of PHI1 in T4. The sampled value of $\overline{BRT}$ in both T3 and T4 must be low before a valid bus retry is recognized by the MMU.

During Hold Acknowledge, the MMU will not recognize the Bus Retry signal.

4.5.6 Bus Error

The Bus Error signal ($\overline{BER}$) will be asserted (low) when a hard, or uncorrectable error occurs (e.g., bus timeout, double ECC error). This signal is sampled at the falling edge of PHI1 in T4. If the MMU detects Bus Error while it is controlling the bus, it will store the virtual address which caused the error in the BEAR (Bus Error Address Register), set the ME bit in the ASR to indicate MMU ERROR. An abort signal ($\overline{ABT}$) will be generated and further memory accesses by the MMU will be inhibited. The NS32382 then returns bus control to the CPU by releasing the $\overline{FLT}$ signal ($\overline{FLT}$ returns high). If the Bus Error signal is received when the CPU is controlling the bus, the MMU will store the virtual address in BEAR, and set the CE bit in the ASR to indicate CPU ERROR. During Hold Acknowledge, the MMU will not recognize the Bus Error signal.

4.5.7 Hold

An external DMA device asserts the $\overline{HOLD}$ line (low) to request the CPU cluster bus. This line is an input to both the CPU and the MMU. If the MMU is not floating CPU ($\overline{FLT}$ line high), it will transfer acknowledge signal, from the CPU directly to its own $\overline{HLDAO}$ output through its $\overline{HLDAI}$ input.

If the CPU is floated off the bus by the MMU when the hold request is received, it (the CPU) will not acknowledge the request. In this case, the $\overline{HLDA}$ input of the MMU remains high. MMU, being the bus master, will grant the bus to the requesting device by pulling its $\overline{HLDAO}$ output low at the end of a write access provided a bus error or bus retry has not occurred.

When the DMA device has completed its access, it will release $\overline{HOLD}$ (returns high) and return bus control to the current bus master.

4.5.8 Slave Instruction Bus Operation

For slave instructions, the bus operation follows a different protocol. A slave bus cycle contains only two clock cycles: T1 and T4. Data transfer between CPU and MMU is controlled by one clock cycle wide pulse on the Slave Protocol Control line ($\overline{SPC}$).

Slave operations are specified in the 4-bit status code from the CPU. CPU starts its slave operation by putting a slave I.D. on the data bus with a status code of 1111. It then pulses the $\overline{SPC}$ line for a one clock cycle. All slaves match their own I.D. with the one on the data bus and only the selected slave will respond to the following sequence of slave transfers. The NS32382 MMU responds to a slave I.D. of 1E(hex). A slave write is accomplished by a one clock $\overline{SPC}$ pulse with status code 1101 while a slave read is accomplished by a one clock $\overline{SPC}$ pulse with status code 1110.

4.0 Functional Description (Continued)

When the slave completes the operation specified by the CPU, it will activate the $\overline{DONE}$ line to signal completion.

4.6 ABORTS AND ERRORS

4.6.1 Aborts

In normal accesses (not RDVAL or WRVAL cycles), the abort signal is asserted by the MMU under the following conditions:

1. A protection exception has occurred.
2. During address translation, the first or second level PTE is invalid.
3. A breakpoint is detected.
4. An MMU bus error has occurred.

In RDVAL or WRVAL cycles, the abort signal is asserted by the MMU only if the first level PTE is invalid and no protection violation is detected.

When a translation exception causes an abort, the TX field in the Abort Status Register (ASR) will be encoded to reflect the type of translation exception which has occurred and the Translation Exception Address Register (TEAR) will be loaded with the current virtual address. The format of the TX field is as follows:

TX1	TX0	
0	0	No Translation Exception
0	1	First Level PTE Invalid
1	0	Second Level PTE Invalid
1	1	Protection Exception

If the abort is triggered by a breakpoint detection, the BP bit in the ASR is set. The actual virtual address which caused the abort is found in the Breakpoint Datas Register (BDR).

4.6.2 Errors

There are two types of errors, CPU error and MMU error. A CPU error occurs when the Bus Error signal ($\overline{BER}$) is sampled low at mid T4 of a CPU access. An MMU error happens when the same signal is found to be low at mid T4 of an MMU access. Since the MMU monitors the $\overline{BER}$ signal continuously, it will set the CE bit or the ME bit in the ASR (depending on a CPU or an MMU error) when an error is detected. The virtual address which caused the error will be recorded in the Bus Error Address Register (BEAR). An abort signal will be generated if an MMU error is detected.

4.7 SLAVE INSTRUCTION EXECUTION

The NS32382 MMU suppports the 32-bit fast slave protocol of the NS32332 CPU. This protocol is optimized for 32-bit transfers. The old 16-bit protocol is not supported by the NS32382 MMU.

There are two types of MMU slave instructions. The first type contains register read and write instructions (LMR and SMR). LMR allows the CPU to load the internal registers in the MMU while SMR enables the CPU to read the contents of those registers. The second type contains address validation instructions (RDVAL and WRVAL) which the CPU uses to detect possible access violations associated with the address.

All MMU instructions are privileged. If they are executed in the user mode, the CPU will trap on all MMU instructions.

4.7.1 MMU Slave Instruction Format

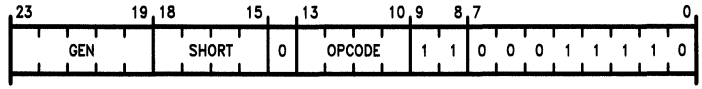
The 3-byte general format of an MMU instruction is shown in *Figure 4.* Bit 0 to bit 7 contains the hex number 1E which corresponds to the slave I.D. of the MMU. Bit 10 to bit 13 is the 4-bit opcode field which specifies one of the four MMU operations. Bit 15 to bit 18 is the short code which the CPU uses to select MMU registers. Bit 19 to bit 23 is the general operand field which is not used in MMU instructions. *Figure 4* shows the format of MMU instructions as they are stored in memory. When the CPU sends a slave instruction to the MMU (with status 1111), the byte order is reversed. For example, the LMR instruction stored in memory has the following format:

GGGGGSSS Byte 2	S0001011 Byte 1	00011110 Byte 0
GGGGG = general operand SSSS = short code 0010 in byte 1 = opcode of LMR instruction 00011110 = slave ID of MMU		

The same instruction, when transferred to the MMU over the 32-bit data bus, will take on the following format:

00011110 Byte 3	S0001011 Byte 2	GGGGGSSS Byte 1	XXXX0000 Byte 0

Figure 5 illustrates the format of the 4 MMU instructions. *Figure 6* contains the possible combinations of the 4-bit MMU short code.



TL/EE/8776–4

FIGURE 4. MMU Slave Instruction Format

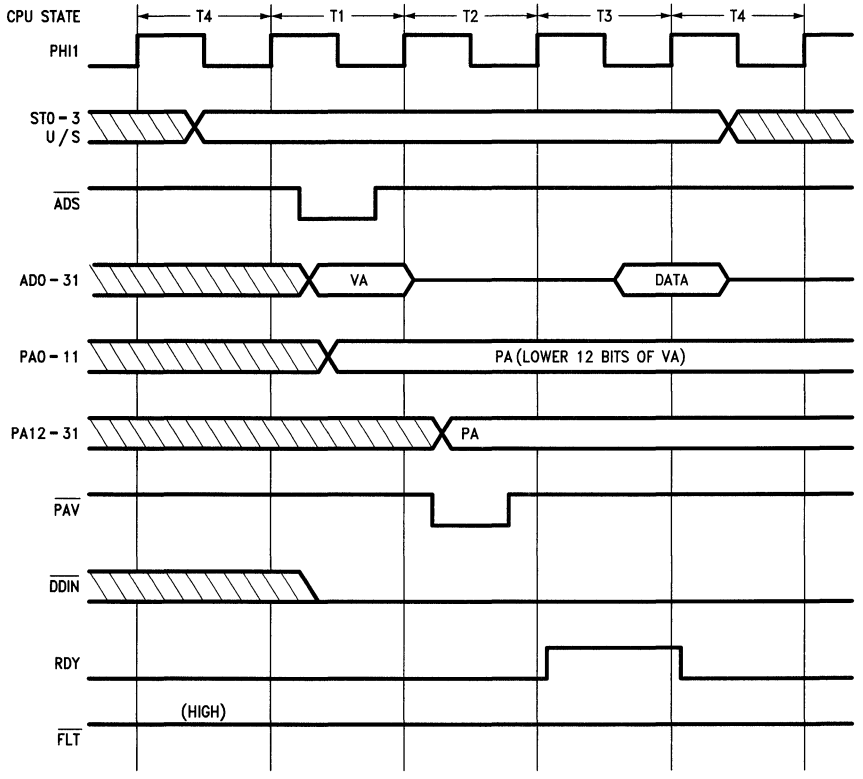
Gen	Short	0	Opcode	D	Slave ID	Instruction
GGGGG	SSSS	0	0010	11	00011110	LMR mreg, src
GGGGG	SSSS	0	0011	11	00011110	SMR mreg, src
GGGGG	0000	0	0000	11	00011110	RDVAL loc
GGGGG	0000	0	0001	11	00011110	WRVAL loc

FIGURE 5. MMU Instruction

4.0 Functional Description (Continued)

Short Code	Register	Description
0000	BAR	Breakpoint Address Register
0001		Reserved
0010	BMR	Breakpoint Mask Register
0011	BDR	Breakpoint Data Register
1110	IVAR0	Invalidate Virtual Address Register 0
1111	IVAR1	Invalidate Virtual Address Register 1
1001	FEW	Feature Enable Work
1010	ASR	Abort Status Register
1011	TEAR	Translation Exception Address Register
0110	BEAR	Bus Error Address Register
1100	PTB0	Page Table Base 0
1101	PTB1	Page Table Base 1

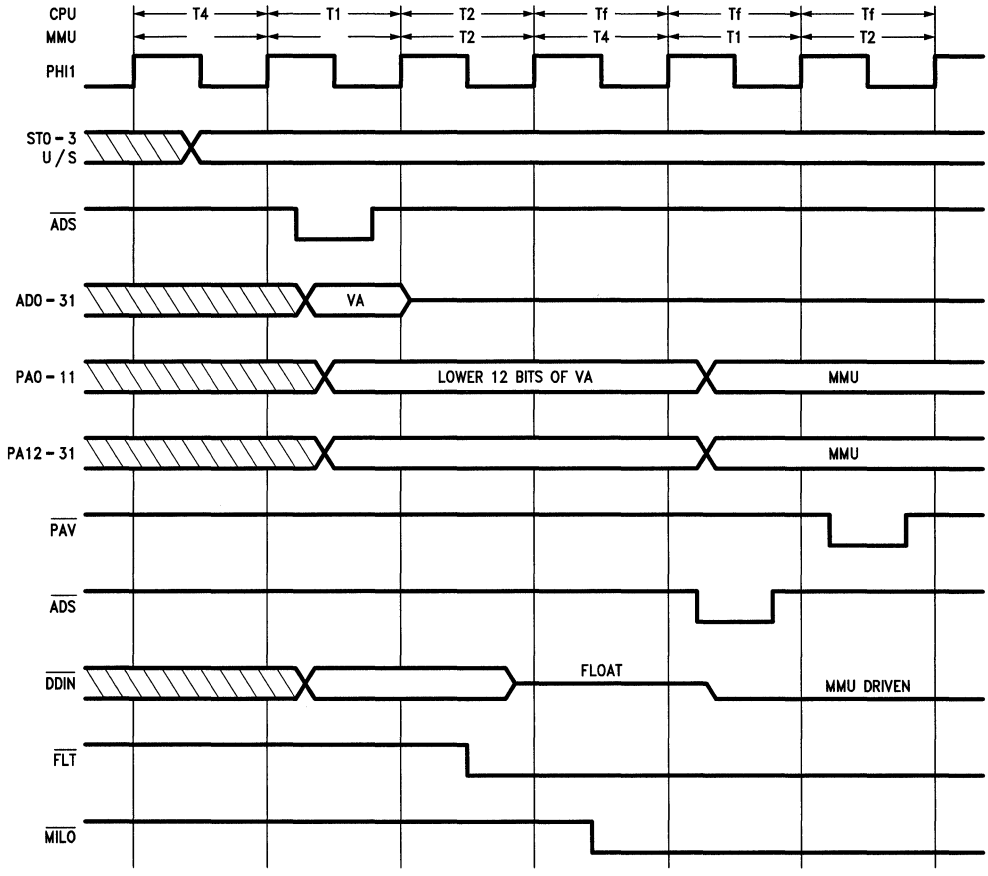
FIGURE 6. MMU Short Code



TL/EE/8776–5

FIGURE 7. Address Translation: TLB Hit CPU Read

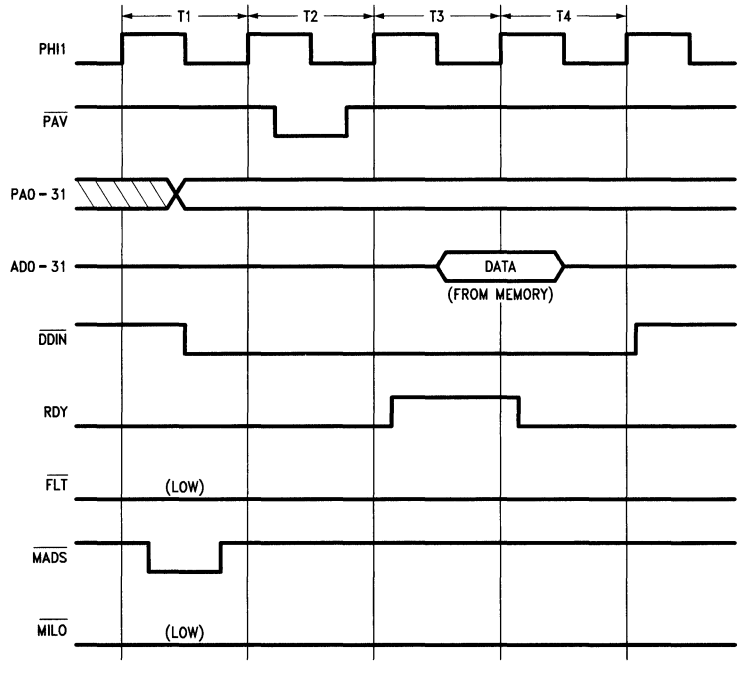
4.0 Functional Description (Continued)



TL/EE/8776–6

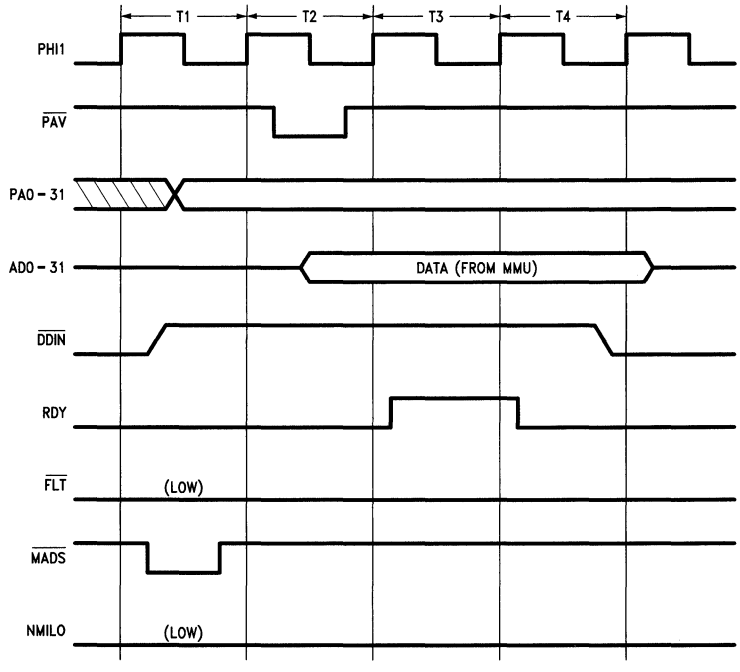
FIGURE 8. FLT Assertion on TLB Miss

4.0 Functional Description (Continued)



TL/EE/8776–7

FIGURE 9. MMU Generated Read Cycle

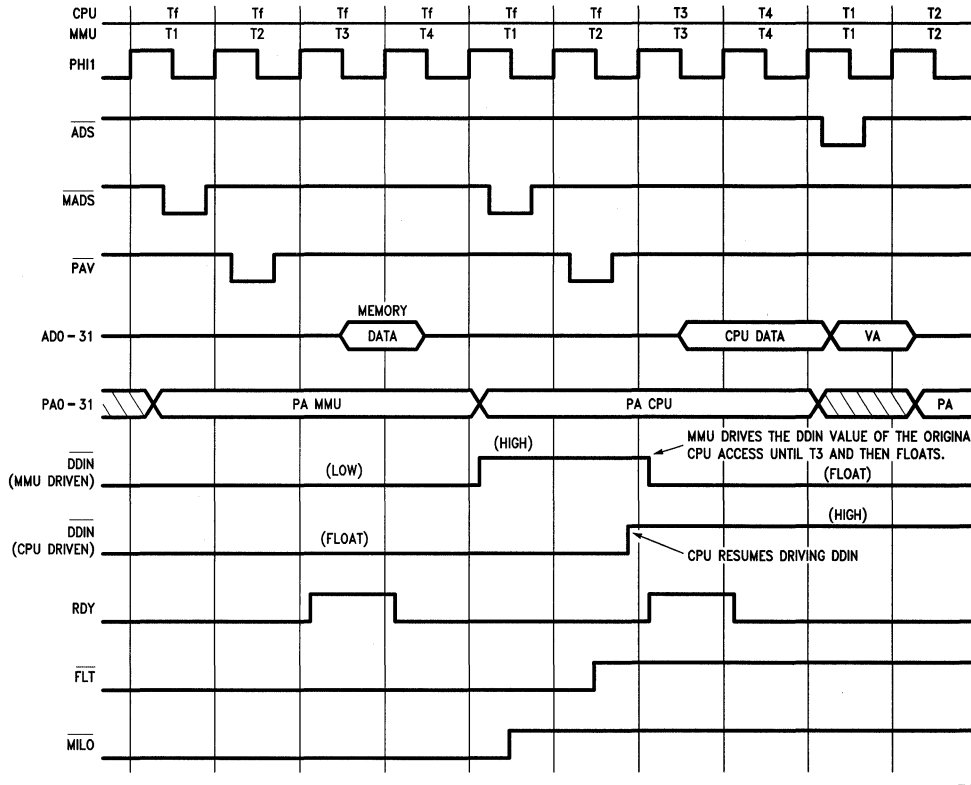


TL/EE/8776–8

FIGURE 10. MMU Generated Write Cycle

3

4.0 Functional Description (Continued)



TL/EE/8776–9

FIGURE 11. FLT Release: MMU Read, CPU Write Originally

4.0 Functional Description (Continued)

NS32082 vs NS32382 Register Comparison

		NS32082 MSR Register	NS32382 MSR Register
ERC	Bit 0	Translation Exception	ASR: TX Code (TX = 00 Means No Translation Exception)
ERC	Bit 1 Bit 2	Not Used 1 = Address Compare Trap 0 = Branch Trace Trap	 ASR: BP Not Implemented
TET	Bit 3 Bit 4 Bit 5	Protection Exception First Level PTE Invalid Second Level PTE Invalid	ASR: TX Code 11 ASR: TX Code 01 ASR: TX Code 10
	Bit 6	BN, Breakpoint Register Number	Only 1 Breakpoint Register
	Bit 7	Not Used	
	Bit 8	ED, Data Direction Bit	ASR: DDI
	Bit 9	BD, Breakpoint Direction	FEW: BR, BW, BX
	Bit 10–12	EST, Error Status Flag (ST0: 2 from CPU)	FEW: ST0–ST3 (ST0: 3 from CPU)
	Bit 13–15	BST, Breakpoint Status Flag	FEW: ST0–ST3
	Bit 16	TU, Translate User Bit	FEW: TU
	Bit 17	TS, Translate Supervisor Bit	FEW: TS
	Bit 18	DS, Dual Space Bit	FEW: DS
	Bit 19	AO, Access Override Bit	FEW: AO
	Bit 20	BEN, Breakpoint Enable Bit	FEW: BR, BW, BX
	Bit 21	UB, User Break Bit	FEW: BS
	Bit 22	AI, Abort/Interrupt Select	Always Abort, No Interrupt
	Bit 23	FT, Flow Trace Bit Trace	Not Supported
	Bit 24	UT, User Trace Bit Trace	Not Supported
	Bit 25	NT, Nonsequential Trace Bit Trace	Not Supported

4.0 Functional Description (Continued)

NS32082 vs. NS32382 Register Comparison (Continued)

NS32082	NS32382
PTB0/PTB1 Bit 31 M = Memory Space Bit Used in System Emulation	No Emulation Support
PF0/PF1 Bit 31 AS = Address Space Bit	No Program Flow Support
EIA Bit 32 AS Bit —During exception report indicates which PTB is active for the current translation. —During TB invalidation, indicates the address space of the entry to be invalidated.	For exception report. The DS field in FEW in conjunction with the UNS field in ASR yield the same information. For invalidation. Address space is selected by loading into either IVAR0 or INVAR1.
BPR0/BPR1 Bit 16 CE = Breakpoint Counter Enable	Breakpoint Counter Not Supported
BPR0/BPR1 Bit 27 BW = Break on Write Bit	FEW: BW
BPR0/BPR1 Bit 28 BR = Break on Read Bit	FEW: BR
BPR0/BPR1 Bit 29 BE = Breakpoint on Execution Bit	FEW: BR, BW, BX BR, BW → Operand BX → Instruction
BPR0/BPR1 Bit 30 VP = Virtual/Physical Address Compare	Virtual Address Compare Only
BPR0/BPR1 Bit 31 AS = PTB0 or PTB1 Pointer for Breakpoint	IVAR0 for AS = 0 IVAR1 for AS = 1
PTEII Bit 32 25th Bit of PA BS = Bank Select Bit	Bank Select Not Supported 32-Bit Physical Address
PIEI Bit 4 Modified Bit	Not Used

Section 4
Peripherals

Section 4 Contents

NS32C201-6, -10, -15 Timing Control Unit (TCU) 4-3
NS32201-6, -8, -10 Timing Control Unit (TCU) 4-25
NS32301-10, -15 Advanced Timing Control Unit (TCU) 4-49
NS32202-6, -8, -10 Interrupt Control Unit (ICU) 4-64
NS16450/INS8250A Asynchronous Communications Element 4-89
NS16550 Asynchronous Communications Element with FIFOs 4-104

National Semiconductor Corporation

PRELIMINARY

NS32C201-6/NS32C201-10/NS32C201-15 Timing Control Units

General Description

The NS32C201 Timing Control Unit (TCU) is a 24-pin device fabricated using National's microCMOS technology. It provides a two-phase clock, system control logic and cycle extension logic for the Series 32000® microprocessor family. The TCU input clock can be provided by either a crystal or an external clock signal whose frequency is twice the system clock frequency.

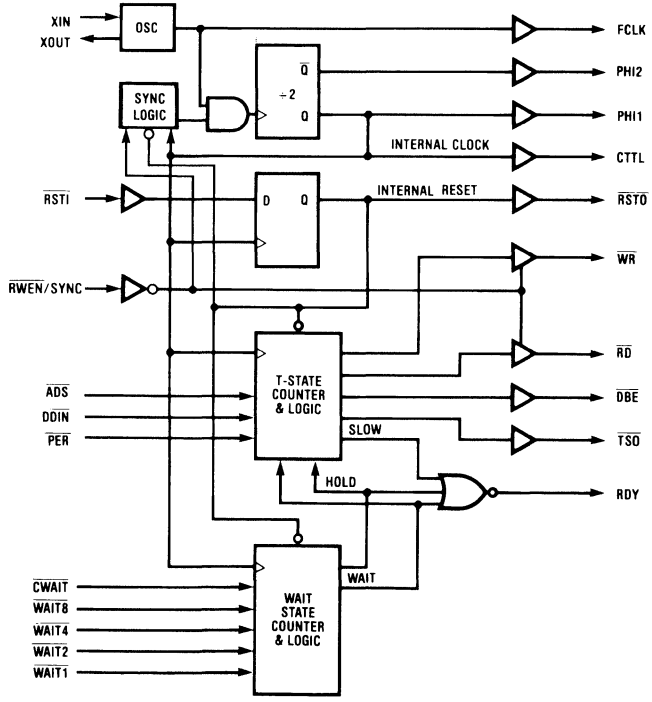
In addition to the two-phase clock for the CPU and MMU (PHI1 and PHI2), it also provides two system clocks for general use within the system (FCLK and CTTL). FCLK is a fast clock whose frequency is the same as the input clock, while CTTL is a replica of PHI1 clock.

The system control logic and cycle extension logic make the TCU very attractive by providing extremely accurate bus control signals, and allowing extensive control over the bus cycle timing.

Features

- Oscillator at twice the CPU clock frequency
- 2 phase full V_{CC} swing clock drivers (PHI1 and PHI2)
- 4-bit input ($\overline{WAITn}$) allowing precise specification of 0 to 15 wait states
- Cycle Hold for system arbitration and/or memory refresh
- System timing (FCLK, CTTL) and control ($\overline{RD}$, $\overline{WR}$, and $\overline{DBE}$) outputs
- General purpose Timing State Output ($\overline{TSO}$) that identifies internal states
- Peripheral cycle to accommodate slower MOS peripherals
- Provides "ready" (RDY) output for the Series 32000 CPUs
- Synchronous system reset generation from Schmitt trigger input
- Phase synchronization to a reference signal
- High-speed CMOS technology
- TTL compatible inputs
- Single 5V power supply
- 24-pin dual-in-line package

Block Diagram



TL/EE/8524–1

4

Table of Contents

1.0 FUNCTIONAL DESCRIPTION

1.1 Power and Grounding
1.2 Crystal Oscillator Characteristics
1.3 Clocks
1.4 Resetting
1.5 Synchronizing Two or More TCUs
1.6 Bus Cycles
1.7 Bus Cycle Extension
1.7.1 Normal Wait States
1.7.2 Peripheral Cycle
1.7.3 Cycle Hold
1.8 Bus Cycle Extension Combinations
1.9 Overriding $\overline{\text{WAIT}}$ Wait States

2.0 DEVICE SPECIFICATIONS

2.1 Pin Descriptions
2.1.1 Supplies
2.1.2 Input Signals
2.1.3 Output Signals
2.2 Absolute Maximum Ratings
2.3 Electrical Characteristics
2.4 Switching Characteristics
2.4.1 Definitions
2.4.2 Output Loading
2.4.3 Timing Tables
2.4.4 Timing Diagrams

List of Illustrations

Crystal Connection 1-1
PHI1 and PHI2 Clock Signals 1-2
Recommended Reset Connections (Non Memory-Managed System) 1-3a
Recommended Reset Connections (Memory-Managed System) 1-3b
Slave TCU does not use $\overline{\text{RWEN}}$ during Normal Operation 1-4a
Slave TCU Uses Both SYNC and $\overline{\text{RWEN}}$ 1-4b
Synchronizing Two TCUs 1-5
Synchronizing One TCU to an External Pulse 1-6
Basic TCU Cycle (Fast Cycle) 1-7
Wait State Insertion Using $\overline{\text{CWAIT}}$ (Fast Cycle) 1-8
Wait State Insertion Using $\overline{\text{WAITn}}$ (Fast Cycle) 1-9
Peripheral Cycle 1-10
Cycle Hold Timing Diagram 1-11
Fast Cycle with 12 Wait States 1-12
Peripheral Cycle with Six Wait States 1-13
Cycle Hold with Three Wait States 1-14
Cycle Hold of a Peripheral Cycle 1-15
Overriding WAITn Wait States 1-16
Connection Diagram 2-1
Clock Signals (a) 2-2
Clock Signals (b) 2-3
Control Inputs 2-4
Control Outputs (Fast Cycle) 2-5
Control Outputs (Peripheral Cycle) 2-6
Control Outputs (TRI-STATE Timing) 2-7
Cycle Hold 2-8
Wait States (Fast Cycle) 2-9
Wait States (Peripheral Cycle) 2-10
Synchronization Timing 2-11

1.0 Functional Description

1.1 POWER AND GROUNDING

The NS32C201 requires a single +5V power supply, applied to pin 24 (V_{CC}). See Electrical Characteristics. The Logic Ground on pin 12 (GND), is the common pin for the TCU.

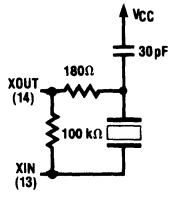
A 0.1 μF, ceramic decoupling capacitor must be connected across V_{CC} and GND, as close to the TCU as possible.

1.2 CRYSTAL OSCILLATOR CHARACTERISTICS

The NS32C201 has an internal oscillator that requires connections of the crystal and bias components to XIN and XOUT as shown in *Figure 1-1.* It is important that the crystal and the RC components be mounted in close proximity to the XIN, XOUT and V_{CC} pins to keep printed circuit trace lengths to an absolute minimum.

Typical Crystal Specifications:

Type ..... At-Cut
Tolerance ..... 0.005% at 25°C
Stability ..... 0.01% from 0° to 70°C
Resonance ..... Fundamental (parallel)
Capacitance ..... 20 pF
Maximum Series Resistance 50Ω



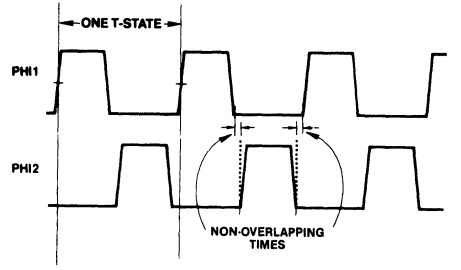
TL/EE/8524-3

CRYSTAL FREQUENCY (MHz)	R (OHM)
6-12	470
12-18	220
18-24	100
24-30	47

FIGURE 1-1. Crystal Connection Diagram

1.3 CLOCKS

The NS32C201 TCU has four clock output pins. The PHI1 and PHI2 clocks are required by the Series 32000 CPUs. These clocks are non-overlapping as shown in *Figure 1-2.*



TL/EE/8524-4

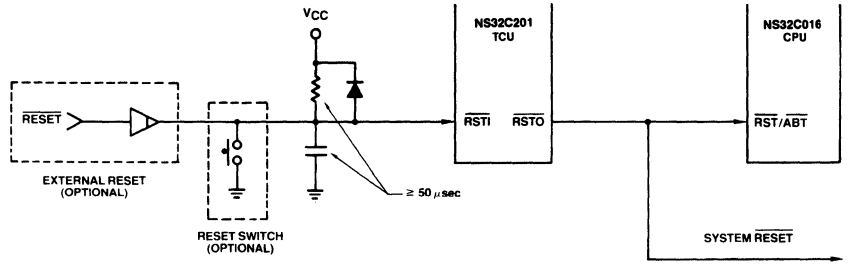
FIGURE 1.2. PHI1 and PHI2 Clock Signals

Each rising edge of PHI1 defines a transition in the timing state of the CPU.

As the TCU generates the various clock signals with very short transition timings, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible. It is also recommended that only the Series 32000 CPU and, if used, the MMU (Memory Management Unit) be connected to the PHI1 and PHI2 clocks.

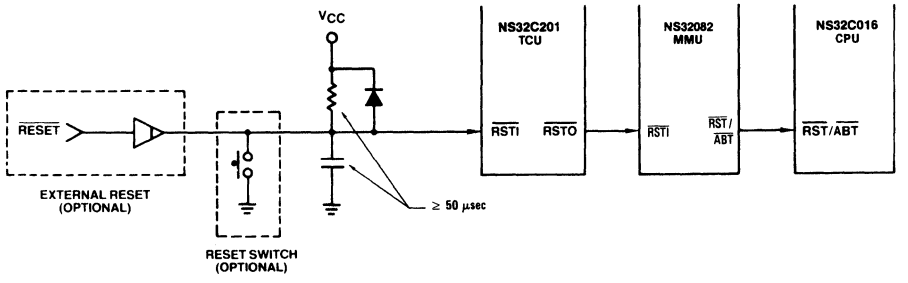
CTTL is a clock signal which runs at the same frequency as PHI1 and is closely balanced with it.

FCLK is a clock, running at the frequency of XIN input. This clock has a frequency that is twice the CTTL clock frequency. The exact phase relationship between PHI1, PHI2, CTTL and FLCK can be found in Section 2.



TL/EE/8524-5

FIGURE 1-3a. Recommended Reset Connections (Non Memory-Managed System)



TL/EE/8524-6

FIGURE 1-3b. Recommended Reset Connections (Memory-Managed System)

1.0 Functional Description (Continued)

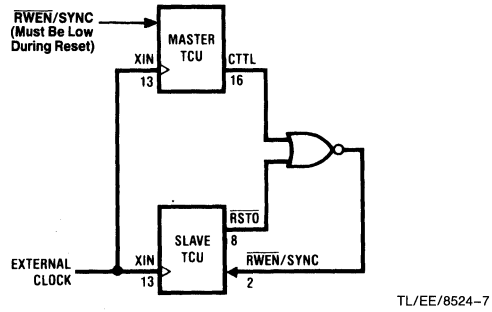
1.4 RESETTING

The NS32C201 TCU provides circuitry to meet the reset requirements of the Series 32000 CPUs. If the Reset Input line, $\overline{RSTI}$ is pulled low, the TCU asserts $\overline{RSTO}$ which resets the Series 32000 CPU. This Reset Output may also be used as a system reset signal. *Figure 1-3a* illustrates the reset connections for a non Memory-Managed system. *Figure 1-3b* illustrates the reset connections for a Memory-Managed system.

1.5 SYNCHRONIZING TWO OR MORE TCUs

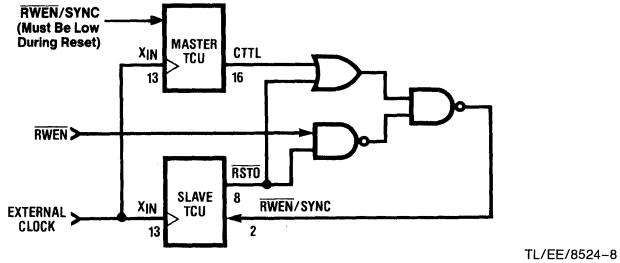
During reset, (when $\overline{RSTO}$ is low), one or more TCUs can be synchronized with a reference (Master) TCU. The $\overline{RWEN}$/SYNC input to the slave TCU(s) is used for synchronization. The Slave TCU samples the $\overline{RWEN}$/SYNC input on the rising edge of XIN. When $\overline{RSTO}$ is low and CTTL is high (see *Figure 1-5*), if $\overline{RWEN}$/SYNC is sampled high, the phase of CTTL of the Slave TCU is shifted by one XIN clock cycle.

Two possible circuits for TCU synchronization are illustrated in *Figures 1-4a* and *1-4b*. It should be noted that when $\overline{RWEN}$/SYNC is high, the $\overline{RD}$ and $\overline{WR}$ signals will be TRI-STATE on the slave TCU.



TL/EE/8524–7

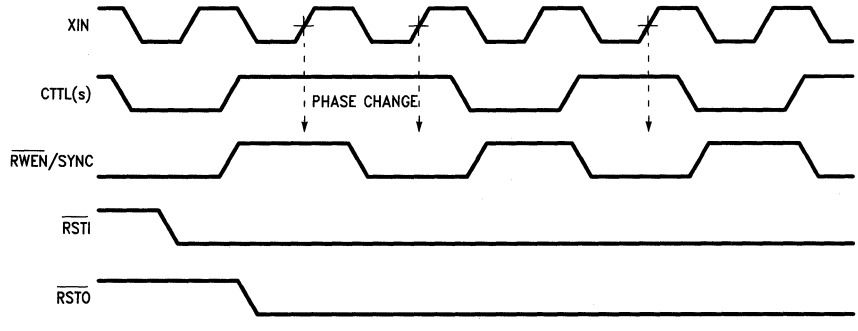
FIGURE 1-4a. Slave TCU Does Not Use $\overline{RWEN}$ During Normal Operation



TL/EE/8524–8

FIGURE 1-4b. Slave TCU Uses Both SYNC and $\overline{RWEN}$

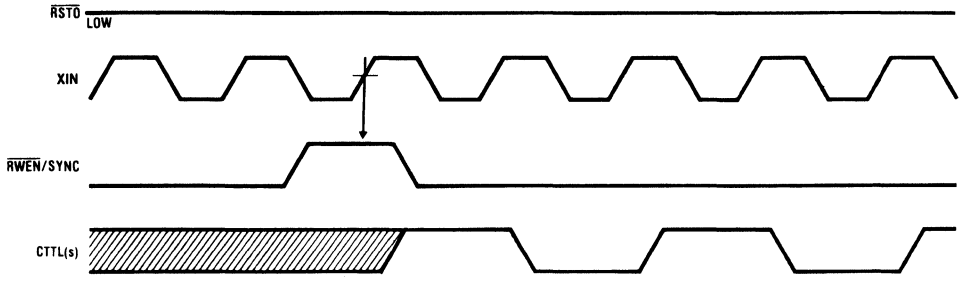
Note: When two or more TCUs are to be synchronized, the XIN of all the TCUs should be connected to an external clock source. For details on the external clock, see Switching Specifications in Section 2.



TL/EE/8524–9

FIGURE 1-5. Synchronizing Two TCUs

1.0 Functional Description (Continued)



TL/EE/8524–10

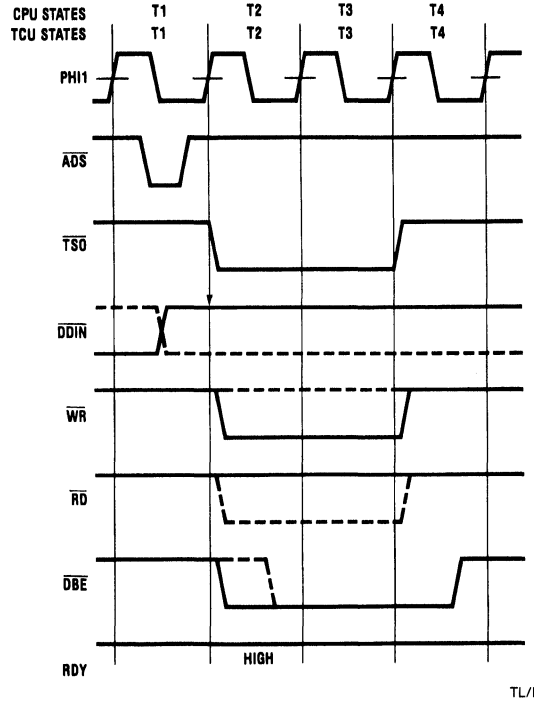
FIGURE 1-6. Synchronizing One TCU to An External Pulse

In addition to synchronizing two or more TCUs, the RWEN/SYNC input can be used to "fix" the phase of one TCU to an external pulse. The pulse to be used must be high for only one rising edge of XIN. Independent of CTTL's state at the XIN rising edge, the CTTL state following the XIN rising edge will be high. *Figure 1-6* shows the timing of this sequence.

1.6 BUS CYCLES

In addition to providing all the necessary clock signals, the NS32C201 TCU provides bus control signals to the system. The TCU senses the ADS signal from the CPU or MMU to start a bus cycle. The DDIN input signal is also sampled to determine whether a Read or Write cycle is to be generated. In addition to RD and WR, other signals are provided: DBE and TSO. DBE is used to enable data buffers. The leading edge of DBE is delayed a half clock period during Read cycles to avoid bus conflicts between data buffers and either the CPU or the MMU. This is shown in *Figure 1-7*.

The Timing State Output (TSO) is a general purpose signal that may be used by external logic for synchronizing to a System cycle. TSO is activated at the beginning of state T2 and returns to the high level at the beginning of state T4 of the CPU cycle. TSO can be used to gate the CWAIT signal when continuous waits are required. Another application of TSO is the control of interface circuitry for dynamic RAMs.



Notes:

1. The CPU and TCU view some timing states (T-states) differently. For clarity, references to T-states will sometimes be followed by (TCU) or (CPU). (CPU) also implies (MMU).
2. Arrows indicate when the TCU samples the input.
3. RWEN is assumed low (RD and WR enabled) unless specified differently.
4. For clarity, T-states for both the TCU and CPU are shown above the diagrams. (See Note 1.)

TL/EE/8524–11

FIGURE 1-7. Basic TCU Cycle (Fast Cycle)

1.0 Functional Description (Continued)

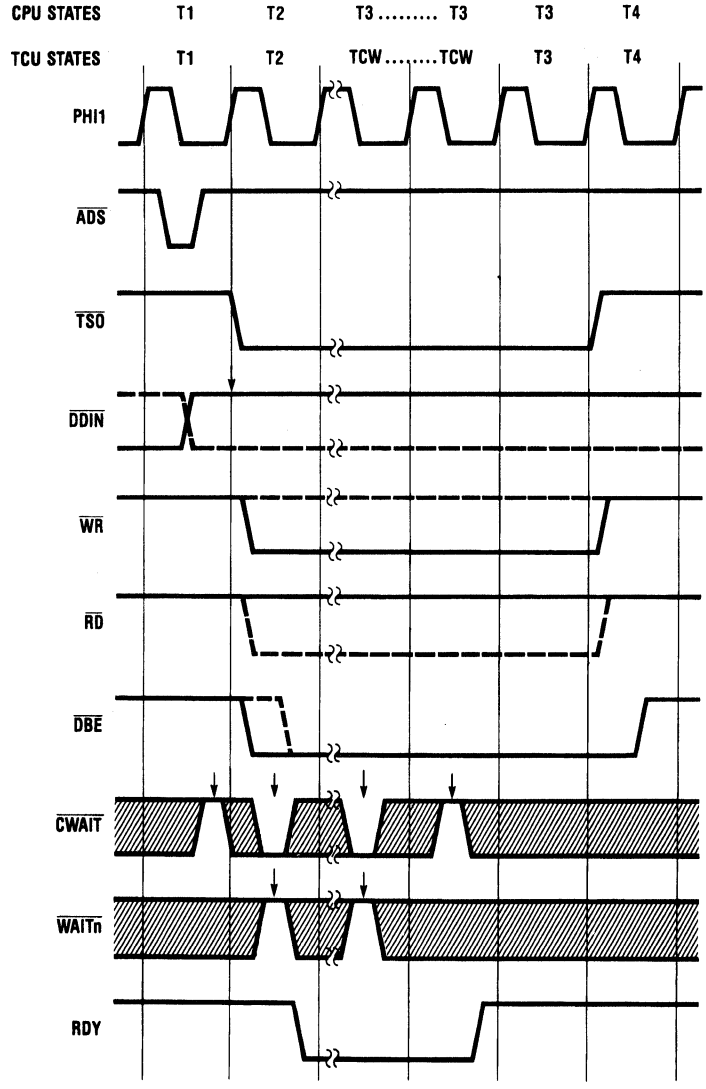
1.7 BUS CYCLE EXTENSION

The NS32C201 TCU uses the Wait input signals to extend normal bus cycles. A normal bus cycle consists of four PHI1 clock cycles. Whenever one or more Wait inputs to the TCU are activated, a bus cycle is extended by at least one PHI1 clock cycle. The purpose is to allow the CPU to access slow memories or peripherals. The TCU responds to the Wait signals by pulling the RDY signal low as long as Wait States are to be inserted in the Bus cycle.

There are three basic cycle extension modes provided by the TCU, as described below.

1.7.1 Normal Wait States

This is a normal Wait State insertion mode. It is initiated by pulling $\overline{\text{CWAIT}}$ or any of the $\overline{\text{WAITn}}$ lines low in the middle of T2. *Figure 1-8* shows the timing diagram of a bus cycle when $\overline{\text{CWAIT}}$ is sampled high at the end of T1 and low in the middle of T2.



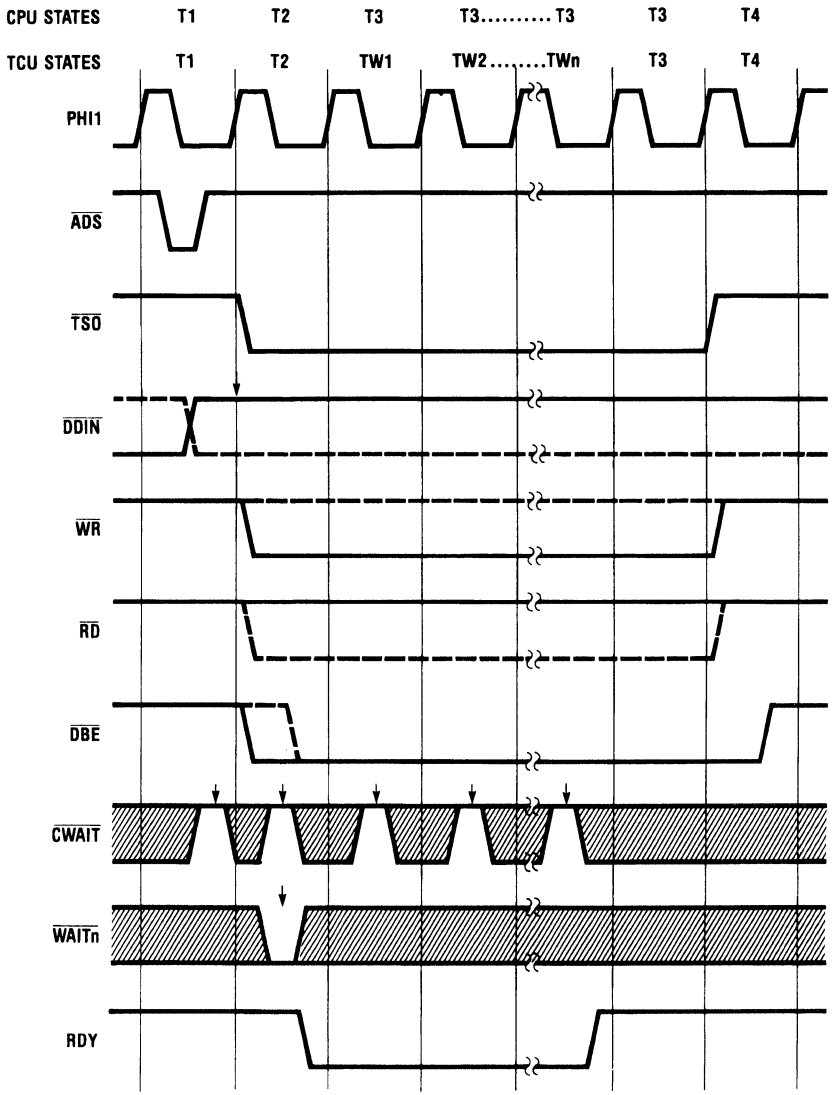
TL/EE/8524–12

FIGURE 1-8. Wait State Insertion Using $\overline{\text{CWAIT}}$ (Fast Cycle)

1.0 Functional Description (Continued)

The RDY signal goes low during T2 and remains low until $\overline{CWAIT}$ is sampled high by the TCU. RDY is pulled high by the TCU during the same PHI1 cycle in which the $\overline{CWAIT}$ line is sampled high.

If any of the $\overline{WAITn}$ signals are sampled low during T2 and $\overline{CWAIT}$ is high during the entire bus cycle, then the RDY line goes low for 1 to 15 clock cycles, depending on the binary weighted value of $\overline{WAITn}$. If, for example, $\overline{WAIT1}$ and $\overline{WAIT4}$ are sampled low, then five wait states will be inserted. This is shown in *Figure 1-9*.



TL/EE/8524–13

FIGURE 1-9. Wait State Insertion Using $\overline{WAITn}$ (Fast Cycle)

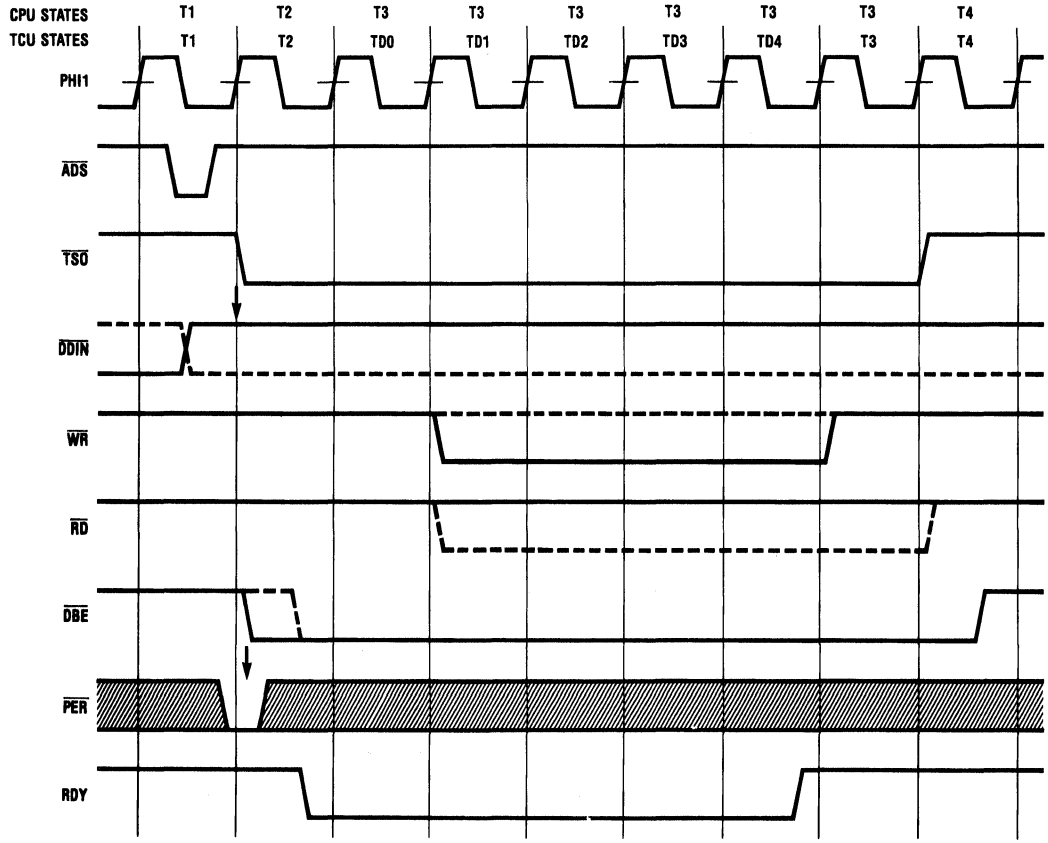
1.0 Functional Description (Continued)

1.7.2 Peripheral Cycle

This cycle is entered when the $\overline{PER}$ signal line is sampled low at the beginning of T2. The TCU adds five wait states identified as TD0–TD4 into a normal bus cycle. The $\overline{RD}$ and $\overline{WR}$ signals are also re-shaped so the setup and hold times for address and data will be increased.

This may be necessary when slower peripherals must be accessed.

Figure 1-10 shows the timing diagram of a peripheral cycle.



TL/EE/8524–14

FIGURE 1-10. Peripheral Cycle

1.0 Functional Description (Continued)

1.7.3 Cycle Hold

If the $\overline{CWAIT}$ input is sampled low at the end of state T1, the TCU will go into cycle hold mode and stay in this mode for as long as $\overline{CWAIT}$ is kept low. During this mode the control signals $\overline{RD}$, $\overline{WR}$, $\overline{TSO}$ and $\overline{DBE}$ are kept inactive; RDY is pulled low, thus causing wait states to be inserted into the bus cycle. The cycle hold feature can be used in applications involving dynamic RAMs. A timing diagram showing the cycle hold feature is shown in *Figure 1-11*.

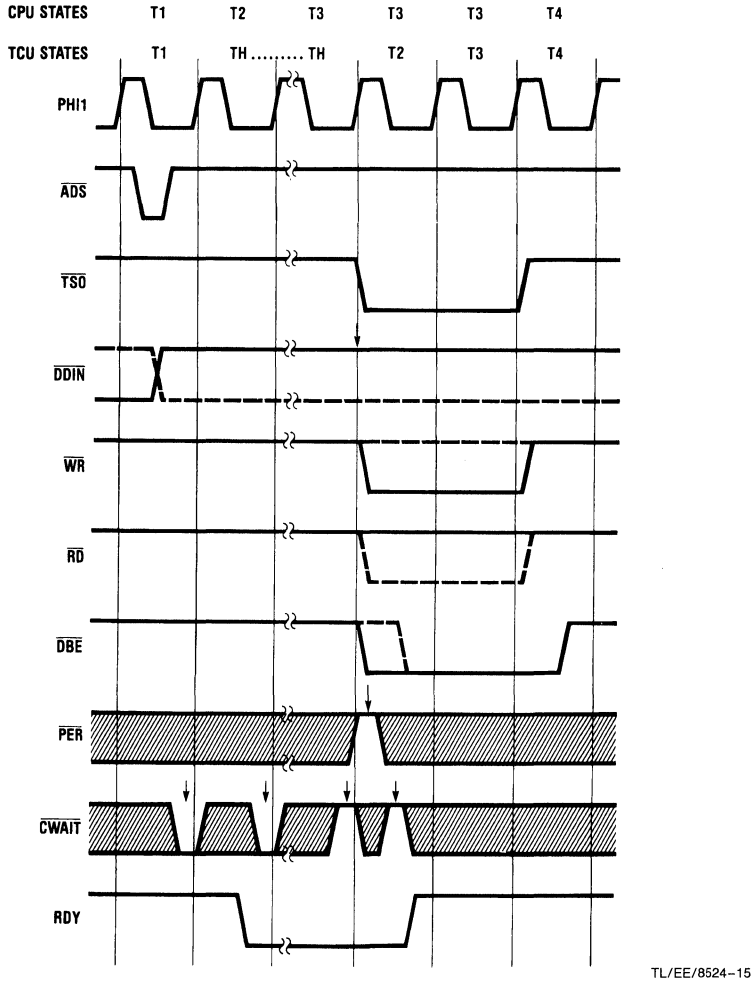


FIGURE 1-11. Cycle Hold Timing Diagram

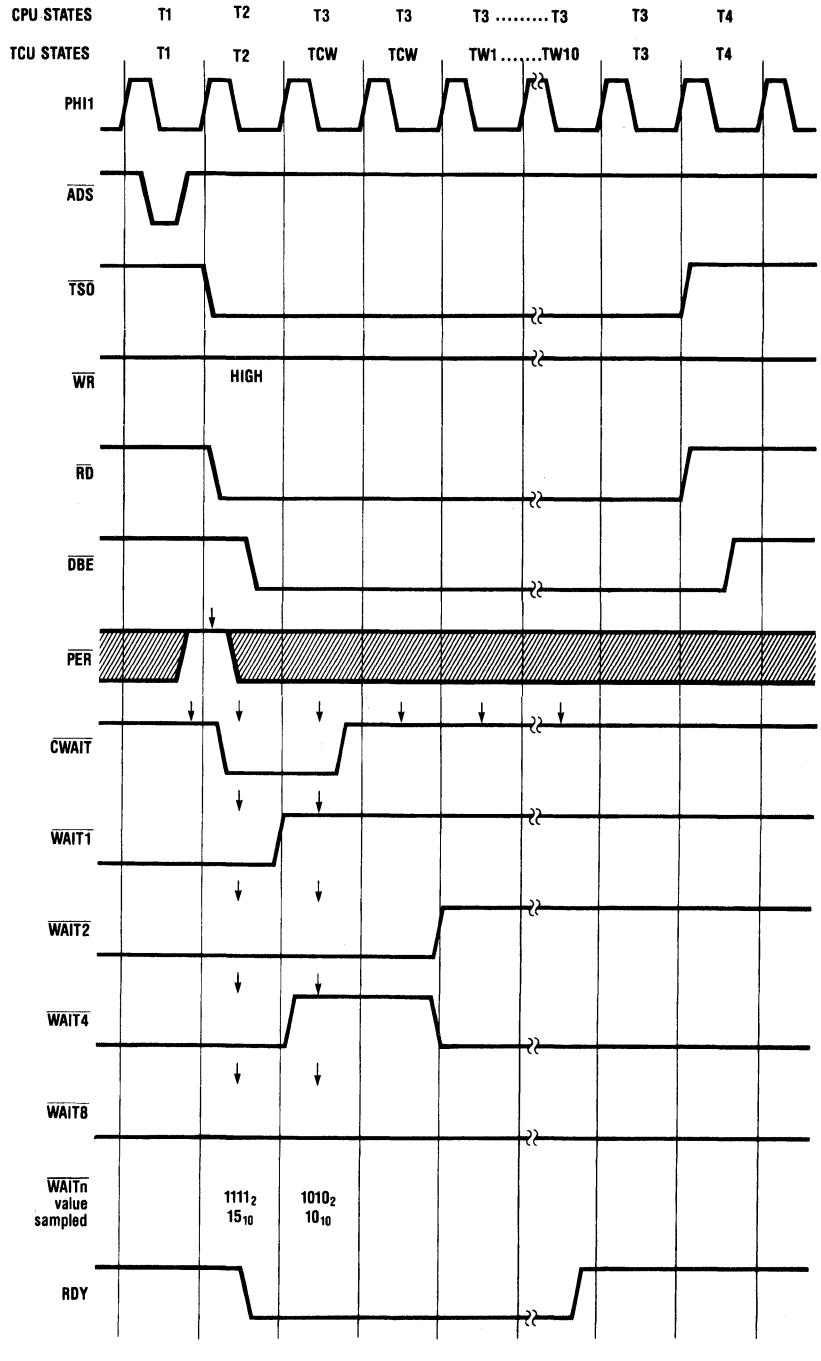
1.8 BUS CYCLE EXTENSION COMBINATIONS

Any combination of the TCU input signals used for extending a bus cycle can be activated at one time. The TCU will honor all of the requests according to a certain priority scheme. A cycle hold request is assigned top priority. It follows a peripheral cycle request, and then $\overline{CWAIT}$ and $\overline{WAITn}$ respectively.

If, for example, all the input signals $\overline{CWAIT}$, $\overline{PER}$ and $\overline{WAITn}$ are asserted at the beginning of the cycle, the TCU will enter the cycle hold mode. As soon as $\overline{CWAIT}$ goes high, the input signal $\overline{PER}$ is sampled to determine whether a peripheral cycle is requested.

Next, the TCU samples $\overline{CWAIT}$ again and $\overline{WAITn}$ to check whether additional wait states have to be inserted into the bus cycle. This sampling point depends on whether $\overline{PER}$ was sampled high or low. If $\overline{PER}$ was sampled high, then the sampling point will be in the middle of the TCU state T2, (*Figure 1-14*), otherwise it will occur three clock cycles later (*Figure 1-15*). *Figures 1-12* to *1-15* show the timing diagrams for different combinations of cycle extensions.

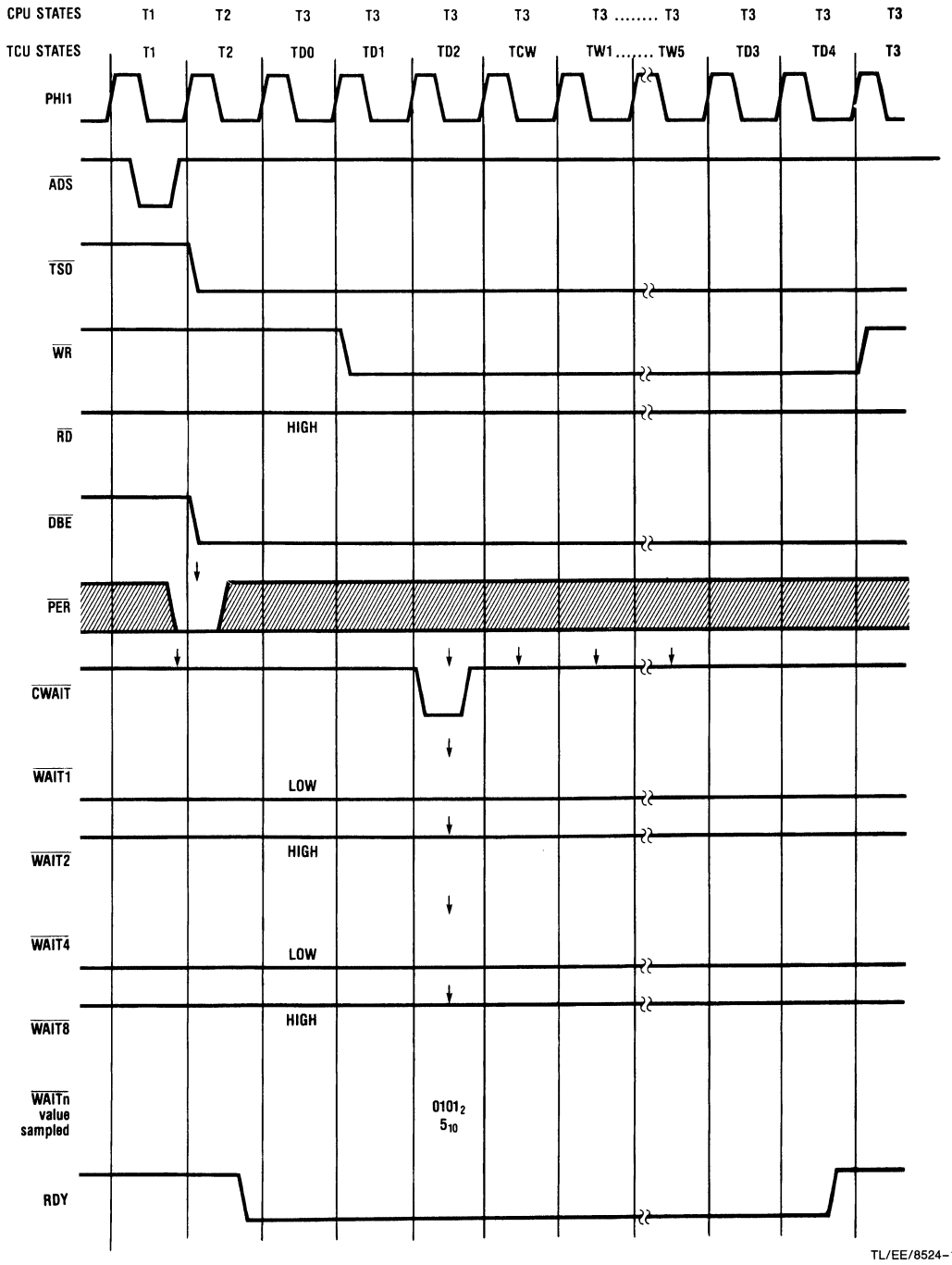
1.0 Functional Description (Continued)



TL/EE/8524–16

FIGURE 1-12. Fast Cycle With 12 Wait States (2 $\overline{\text{CWAIT}}$ and $\overline{\text{WAIT10}}$) (Read Cycle)

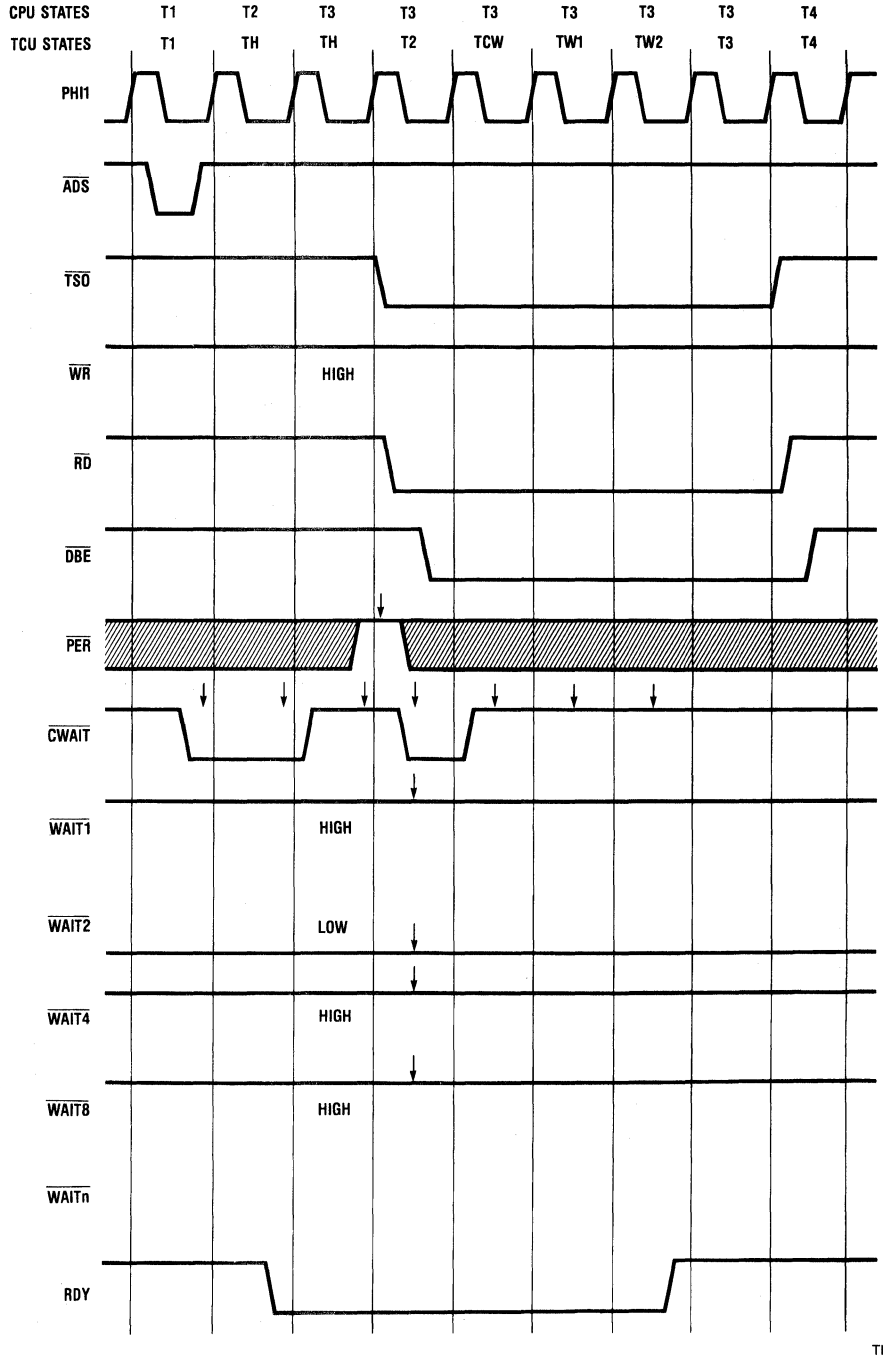
1.0 Functional Description (Continued)



TL/EE/8524–17

FIGURE 1-13. Peripheral Cycle with Six Wait States (1 CWAIT and WAIT5) (Write Cycle)

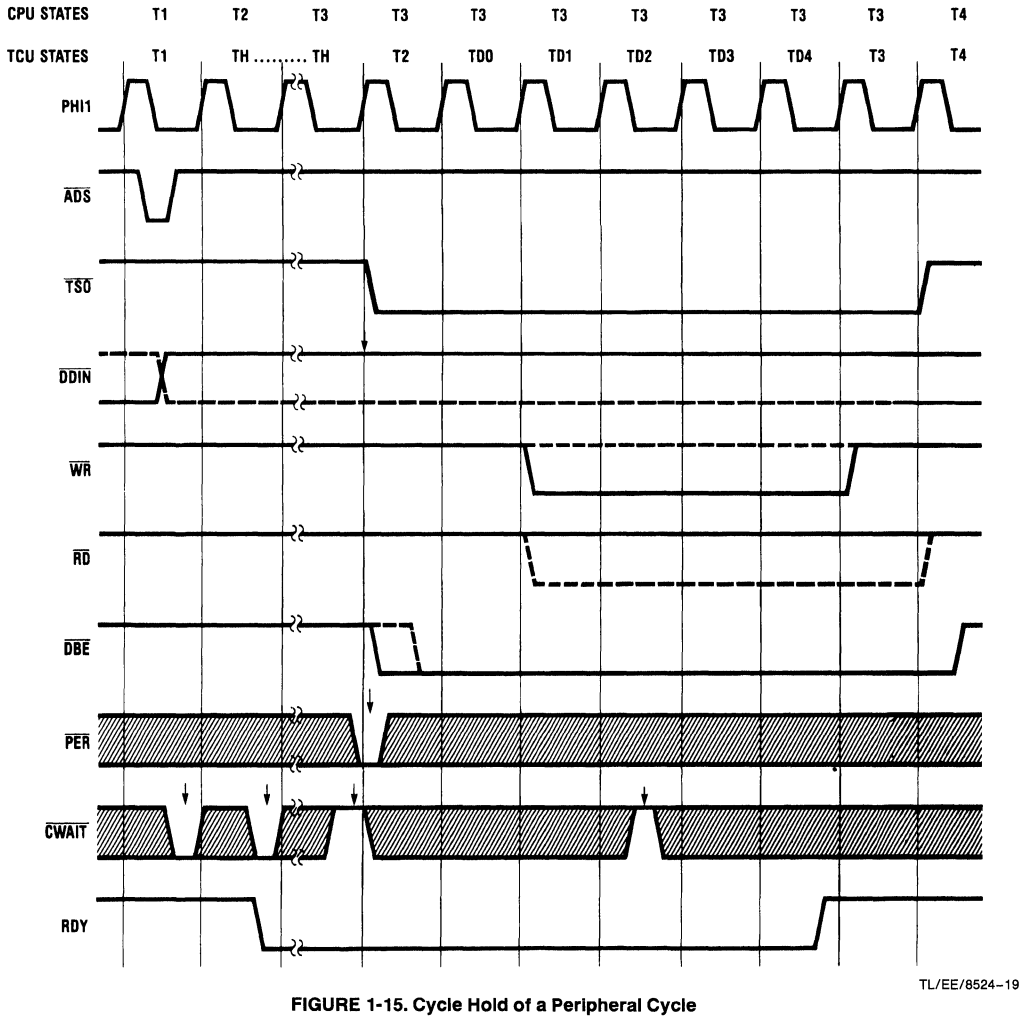
1.0 Functional Description (Continued)



TL/EE/8524–18

FIGURE 1-14. Cycle Hold with Three Wait States (1 $\overline{\text{CWAIT}}$ and $\overline{\text{WAIT2}}$) (Read Cycle)

1.0 Functional Description (Continued)



TL/EE/8524–19

FIGURE 1-15. Cycle Hold of a Peripheral Cycle

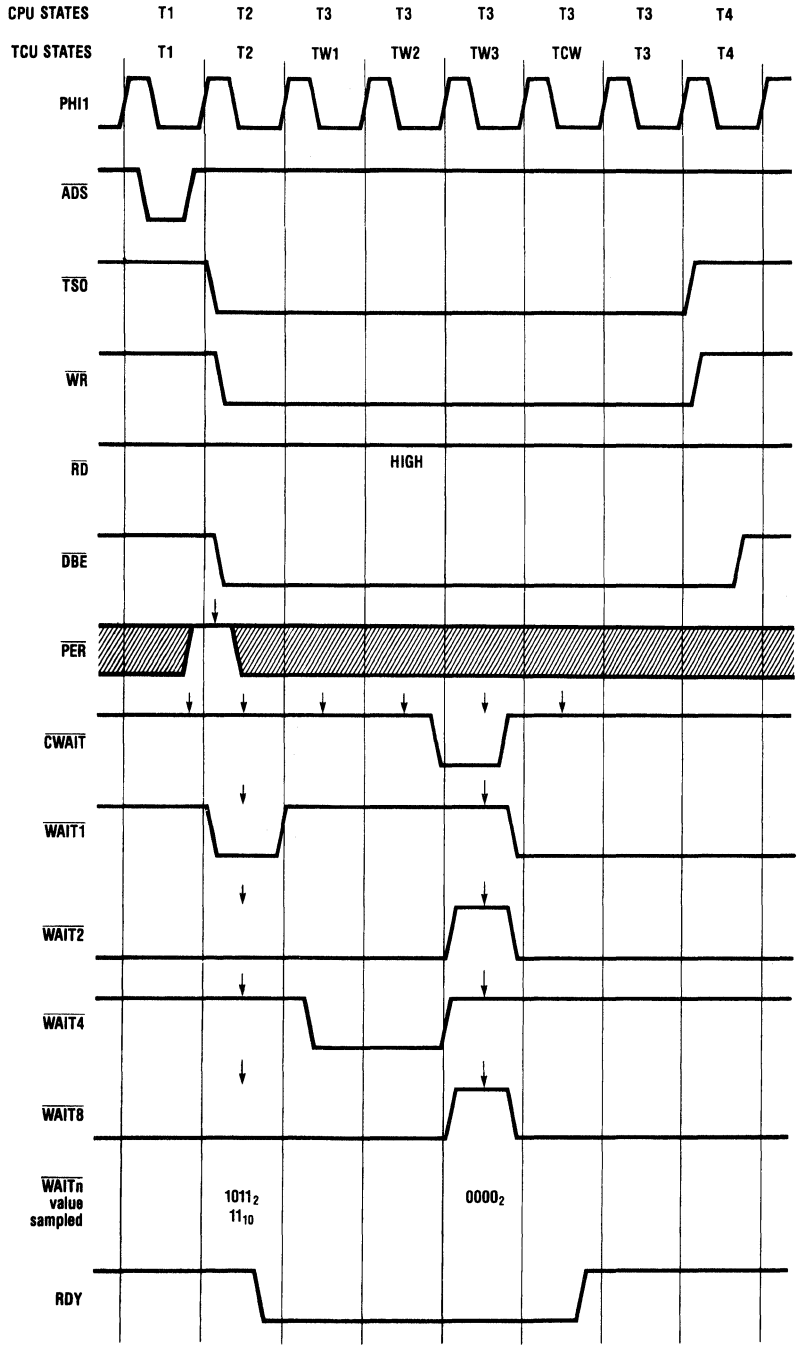
1.9 OVERRIDING WAITn WAIT STATES

The TCU handles the WAITn Wait States by means of an internal counter that is reloaded with the binary value corresponding to the state of the WAITn inputs each time CWAIT is sampled low, and is decremented when CWAIT is high.

This allows to either extend a bus cycle of a predefined number of clock cycles, or prematurely terminate it. To terminate a bus cycle, for example, CWAIT must be asserted for at least one clock cycle, and the WAITn inputs must be forced to their inactive state.

At least one wait state is always inserted when using this procedure as a result of CWAIT being sampled low. *Figure 1-16* shows the timing diagram of a prematurely terminated bus cycle where eleven wait states were being inserted.

1.0 Functional Description (Continued)



TL/EE/8524–20

FIGURE 1-16. Overriding $\overline{\text{WAITn}}$ Wait States (Write Cycle)

2.0 Device Specifications

2.1 PIN DESCRIPTIONS

The following is a description of all NS32C201 pins. The descriptions reference portions of the Functional Description, Section 1.

2.1.1 Supplies

Power (V_{CC}): +5V positive supply. Section 1.1.

Ground (GND): Power supply return. Section 1.1.

2.1.2 Input Signals

Reset Input ($\overline{RSTI}$): Active low. Schmitt triggered, asynchronous signal used to generate a system reset. Section 1.4.

Address Strobe ($\overline{ADS}$): Active low. Identifies the first timing state (T1) of a bus cycle.

Data Direction Input ($\overline{DDIN}$): Active low. Indicates the direction of the data transfer during a bus cycle. Implies a Read when low and a Write when high.

Note: In Rev. A of the NS32C201 this signal is CMOS compatible. In later revisions it is TTL compatible.

Read/Write Enable and Synchronization ($\overline{RWEN}$/SYNC): TRI-STATE® the $\overline{RD}$ and the $\overline{WR}$ outputs when high and enables them when low. Also used to synchronize the phase of the TCU clock signals, when two or more TCUs are used. Section 1.5.

Crystal or External Clock Source (XIN): Input from a crystal or an external clock source. Section 1.3.

Continuous Wait ($\overline{CWAIT}$): Active low. Initiates a continuous wait if sampled low in the middle of T2 during a Fast cycle, or in the middle of TD2, during a peripheral cycle. If $\overline{CWAIT}$ is low at the end of T1, it initiates a Cycle Hold. Section 1.7.1.

Four-Bit Wait State Inputs ($\overline{WAIT1}$, $\overline{WAIT2}$, $\overline{WAIT4}$ and $\overline{WAIT8}$): Active low. These inputs, (collectively called $\overline{WAITn}$), allow from zero to fifteen wait states to be specified. They are binary weighted. Section 1.7.1.

Peripheral Cycle ($\overline{PER}$): Active low. If active, causes the TCU to insert five wait states into a normal bus cycle. It also causes the Read and Write signals to be re-shaped to meet the setup and hold timing requirement of slower MOS peripherals. Section 1.7.2.

2.1.3 Output Signals

Reset Output ($\overline{RSTO}$): Active low. This signal becomes active when $\overline{RSTI}$ is low, initiating a system reset. $\overline{RSTO}$ goes high on the first rising edge of PHI1 after $\overline{RSTI}$ goes high. Section 1.4.

Read Strobe ($\overline{RD}$): (TRI-STATE) Active low. Identifies a Read cycle. It is decoded from $\overline{DDIN}$ and TRI-STATE by $\overline{RWEN}$/SYNC. Section 1.6.

Write Strobe ($\overline{WR}$): (TRI-STATE) Active low. Identifies a Write cycle. It is decoded from $\overline{DDIN}$ and TRI-STATE by $\overline{RWEN}$/SYNC. Section 1.6.

Note: $\overline{RD}$ and $\overline{WR}$ are mutually exclusive in any cycle. Hence they are never low at the same time.

Data Buffer Enable ($\overline{DBE}$): Active low. This signal is used to control the data bus buffers. It is low when the data buffers are to be enabled. Section 1.6.

Timing State Output ($\overline{TSO}$): Active low. The falling edge of $\overline{TSO}$ signals the beginning of state T2 of a bus cycle. The rising edge of $\overline{TSO}$ signals the beginning of state T4. Section 1.6.

Ready (RDY): Active high. This signal will go low and remain low as long as wait states are to be inserted in a bus cycle. It is normally connected to the RDY input of the CPU. Section 1.7.

Fast Clock (FCLK): This is a clock running at the same frequency as the crystal or the external source. Its frequency is twice that of the CPU clocks. Section 1.3.

CPU Clocks (PHI1 and PHI2): These outputs provide the Series 32000 CPU with two phase, non-overlapping clock signals. Their frequency is half that of the crystal or external source. Section 1.3.

System Clock (CTTL): This is a system version of the PHI1 clock. Hence, it operates at the CPU clock frequency. Section 1.3.

Crystal Output (XOUT): This line is used as the return path for the crystal (if used). It must be left open when an external clock source is used to drive XIN. Section 1.2.

2.0 Device Specifications (Continued)

2.2 ABSOLUTE MAXIMUM RATINGS (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage	7V
Input Voltages	$-0.5V$ to $V_{CC} + 0.5V$
Output Voltages	$-0.5V$ to $V_{CC} + 0.5V$
Storage Temperature	$-65°C$ to $+150°C$
Lead Temperature (Soldering, 10 sec.)	300°C
Continous Power Dissipation	1W

Note: *Absolute maximum ratings indicate limits beyond which permament damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

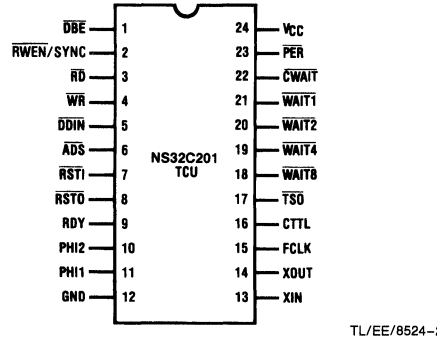
2.3 ELECTRICAL CHARACTERISTICS $T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	All Inputs Except $\overline{RSTI}$ & XIN			0.8	V
V_{IH}	Input High Voltage	All Inputs Except $\overline{RSTI}$ & XIN	2.0			V
V_{T+}	$\overline{RSTI}$ Rising Threshold Voltage	$V_{CC} = 5.0V$	2.5		3.5	V
V_{HYS}	$\overline{RSTI}$ Hysteresis Voltage	$V_{CC} = 5.0V$	0.8		1.8	V
V_{XL}	XIN Input Low Voltage				0.20 V_{CC}	V
V_{XH}	XIN Input High Voltage		0.80 V_{CC}			V
I_{IL}	Input Low Current	$V_{IN} = 0V$			-10	μA
I_{IH}	Input High Current	$V_{IN} = V_{CC}$			10	μA
V_{OL}	Output Low Voltage All Outputs Except XOUT	$I = 2$ mA			0.10 V_{CC}	V
V_{OH}	Output High Voltage All Outputs Except XOUT	$I = -1$ mA	0.90 V_{CC}			V
$I_{O(OFF)}$	Output Leakage Current on $\overline{RD}/\overline{WR}$	$0.4V \le V_{OUT} \le V_{CC}$	-20		$+20$	μA
I_{CC}	Supply Current	$f_{xin} = 20$ MHz		100	12011	mA

Note 1: All typical values are for $V_{CC} = 5V$ and $T_A = 25°C$.

Connection Diagram

Dual-In-Line Package



TL/EE/8524–2

Top View

Order Number NS32C201D or NS32C201N
See NS Package Number D24C or N24A

FIGURE 2.1

2.0 Device Specifications (Continued)

2.4 SWITCHING CHARACTERISTICS

2.4.1 Definitions

All the timing specifications given in this section refer to 15% or 85% of V_{CC} on the rising or falling edges of the clock phases PHI1 and PHI2, and all output signals; to 30% or 70% of V_{CC} on all the CMOS input signals, and to 0.8V or 2.0V on all the TTL input signals, unless specifically stated otherwise.

These specifications are guaranteed under the following conditions:

$T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V

2.4.2 Output Loading

Capacitive loading on output pins for the NS32C201.

RDY, $\overline{DBE}$, $\overline{TSO}$ ..... 50 pF

$\overline{RD}$, $\overline{WR}$ ..... 75 pF

CTTL ..... 50 ÷ 100 pF

FCLK ..... 100 pF

PHI1, PHI2 ..... 170 pF

ABBREVIATIONS

L.E.—Leading Edge

T.E.—Trailing Edge

R.E.—Rising Edge

F.E.—Falling Edge

2.4.3 Timing Tables

Symbol	Figure	Description	Reference/Conditions	NS32C201-6		NS32C201-10		NS32C201-15		Units
				Min	Max	Min	Max	Min	Max	
CLOCK-SIGNALS (XIN, FCLK, PHI1 & PHI2) TIMING										
t_{Cp}	2.2	Clock Period	PHI1 R.E. to Next PHI1 R.E.	160		100		66		ns
t_{CLh}	2.2	Clock High Time	At 90% V_{CC} on PHI1 (Both Edges)	0.5 t_{Cp} −17 ns	0.5 t_{Cp} −7 ns	0.5 t_{Cp} −15 ns	0.5 t_{Cp} −7 ns	0.5 t_{Cp} −10 ns	0.5 t_{Cp} −3 ns	
t_{CLl}	2.2	Clock Low Time	At 10% V_{CC} on PHI1	0.5 t_{Cp} −3 ns	0.5 t_{Cp} +12 ns	0.5 t_{Cp} −3 ns	0.5t_{CP} +10 ns	0.5 t_{Cp} −3 ns	0.5 t_{Cp} +6 ns	
$t_{CLw(1,2)}$	2.2	Clock Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	0.5 t_{CP} −14 ns	0.5 t_{CP} −4 ns	0.5 t_{CP} −10 ns	0.5 t_{CP} −4 ns	0.5 t_{CP} −6 ns	0.5 t_{CP} −4 ns	
t_{CLwas}		PHI1, PHI2 Asymmetry (t_{CLw} (Note 1) − t_{CLw} (Note 2))	At 2.0V on PHI1, PHI2	−5	5	−5	5	−3	3	ns
t_{CLR}	2.2	Clock Rise Time	10% to 90% V_{CC} on PHI1 R.E.		9		7		5	ns
t_{CLF}	2.2	Clock Fall Time	90% to 10% V_{CC} on PHI1 F.E.		7		7		6	ns
t_{nOVL} (Notes 1, 2)	2.2	Clock Non-Overlap Time	At 10% V_{CC} on PHI1, PHI2	0	5	0	5	0	5	ns
t_{nOVLas}		Non-Overlap Asymmetry (t_{nOVL} (Note 1) − t_{nOVL} (Note 2))	At 10% V_{CC} on PHI1, PHI2	−4	4	−4	4	−3	3	ns
t_{Xh}	2.2	XIN High Time (External Input)	At 80% V_{CC} on XIN (Both Edges)	25		16		10		ns
t_{Xl}	2.2	XIN Low Time (External Input)	At 20% V_{CC} on XIN (Both Edges)	25		16		10		ns
t_{XFr}	2.2	XIN to FCLK R.E. Delay	80% V_{CC} on XIN R.E. to FCLK R.E.	6	31	6	29	6	29	ns
t_{XFf}	2.2	XIN to FCLK F.E. Delay	20% V_{CC} on XIN F.E. to FCLK F.E.	6	31	6	29	6	29	ns
t_{XCr}	2.2	XIN to CTTL R.E. Delay	80% V_{CC} on XIN R.E. to CTTL R.E.	6	40	6	34		18	ns
t_{XPr}	2.2	XIN to PHI1 R.E. Delay	80% V_{CC} on XIN R.E. to PHI1 R.E.	6	40	6	32		18	ns
t_{FCr}	2.2	FCLK to CTTL R.E. Delay	FCLK R.E. to CTTL R.E.	0	10	0	6	0	6	ns
t_{FCf}	2.2	FCLK to CTTL F.E. Delay	FCLK R.E. to CTTL F.E.	−1	6	−1	4	−1	4	ns
t_{FPr}	2.3	FCLK to PHI1 R.E. Delay	FCLK R.E. to PHI1 R.E.	−1	6	−1	4	−1	4	ns
t_{FPf}	2.3	FCLK to PHI1 F.E. Delay	FCLK R.E. to PHI1 F.E.	−5	4	−5	2	−5	2	ns
t_{Fh}	2.3	FCLK High Time with Crystal	At 85% V_{CC} on FCLK (Both Edges)	0.25 t_{Cp} −7 ns	0.25 t_{Cp} +7 ns	0.25 t_{Cp} −5 ns	0.25 t_{Cp} +5 ns	0.25 t_{Cp} −5 ns	0.25 t_{Cp} +5 ns	
t_{PCf}	2.3	PHI2 R.E.to CTTL F.E. Delay	PHI2 R.E. to CTTL F.E.	−3	6	−3	4	−3	3	ns
t_{CTh}	2.3	CTTL High Time	At 85% V_{CC} on CTTL (Both Edges)	0.5 t_{Cp} −14 ns	0.5 t_{Cp} +2 ns	0.5 t_{Cp} −12 ns	0.5 t_{Cp} +1 ns	0.5 t_{Cp} −10 ns	0.5 t_{Cp} +1 ns	

Note 1: t_{XCr}, t_{FCr}, t_{FCf}, t_{PCf}, t_{CTh} are measured with 100 pF load on CTTL.

Note 2: PHI1 and PHI2 are interchangeable for the following parameters: t_{Cp}, t_{CLh}, t_{CLl}, t_{CLw}, t_{CLR}, t_{CLF}, t_{nOVL}, t_{XPr}, t_{FPr}, t_{FPf}.

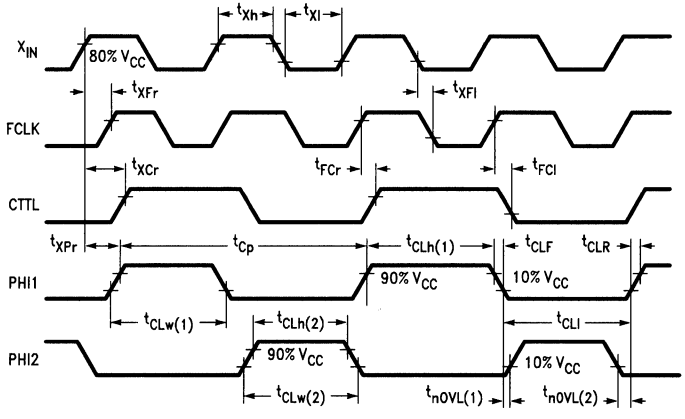
2.0 Device Specifications (Continued)

2.4.3 Timing Tables (Continued)

Symbol	Figure	Description	Reference/Conditions	NS32C201-6		NS32C201-10		NS32C201-15		Units
				Min	Max	Min	Max	Min	Max	
CTTL TIMING (CL = 50 pF)										
t_{PCr}	2.3	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	−2	7	−2	5	−2	2	ns
t_{CTR}	2.3	CTTL Rise Time	10% to 90% V_{CC} on CTTL R.E.		6		5		3	ns
t_{CTF}	2.3	CTTL Fall Time	90% to 10% V_{CC} on CTTL F.E.		6		5		3	ns
CTTL TIMING (CL = 100 pF)										
t_{PCr}	2.3	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	−2	8	−2	6		3	ns
t_{CTR}	2.3	CTTL Rise Time	10% to 90% V_{CC} on CTTL R.E.		8		7		5	ns
t_{CTF}	2.3	CTTL Fall Time	90% to 10% V_{CC} on CTTL F.E.		7		7		5	ns
CONTROL INPUTS ($\overline{RST1}$, $\overline{RSTO}$, $\overline{ADS}$, $\overline{DDIN}$) TIMING										
t_{RSTr}	2.4	$\overline{RSTO}$ R.E. Delay	After PHI1 R.E.		25		21		10	ns
t_{RSTs}	2.4	$\overline{RST1}$ Setup Time	Before PHI1 R.E.	15		15		10		
t_{ADs}	2.4	$\overline{ADS}$ Setup Time	Before PHI1 R.E.	35		30		20		ns
t_{ADw}	2.4	$\overline{ADS}$ Pulse Width	$\overline{ADS}$ L.E. to $\overline{ADS}$ T.E.	25		25		20		ns
t_{DDs}	2.4	$\overline{DDIN}$ Setup Time	Before PHI1 R.E.	10		10		10		ns
CONTROL OUTPUTS ($\overline{TSD}$, $\overline{RD}$, $\overline{WR}$, $\overline{DBE}$ & $\overline{RWEN}$/SYNC) TIMING										
t_{Tf}	2.5	$\overline{TSO}$ L.E. Delay	After PHI1 R.E.		15		12		6	ns
t_{Tr}	2.5	$\overline{TSO}$ T.E. Delay	After PHI1 R.E.		20		18		10	ns
$t_{RWf(F)}$	2.5	$\overline{RD}$/$\overline{WR}$ L.E. Delay (Fast Cycle)	After PHI1 R.E.		50		35		21	ns
$t_{RWf(S)}$	2.6	$\overline{RD}$/$\overline{WR}$ L.E. Delay (Peripheral Cycle)	After PHI1 R.E.		30		25		15	ns
t_{RWr}	2.5/6	$\overline{RD}$/$\overline{WR}$ T.E. Delay	After PHI1 R.E.		25		25		15	ns
$t_{DBf(W)}$	2.5/6	$\overline{DBE}$ L.E. Delay (Write Cycle)	After PHI1 R.E.		28		28		15	ns
$t_{DBf(R)}$	2.5/6	$\overline{DBE}$ L.E. Delay (Read Cycle)	After PHI2 R.E.		25		21		11	ns
t_{DBr}	2.5/6	$\overline{DBE}$ T.E. Delay	After PHI2 R.E.		23		23		20	ns
t_{pLZ}	2.7	$\overline{RD}$,$\overline{WR}$ Low Level to TRI-STATE	After $\overline{RWEN}$/SYNC R.E.		20		20		15	ns
t_{pHZ}	2.7	$\overline{RD}$,$\overline{WR}$ High Level to TRI-STATE	After $\overline{RWEN}$/SYNC R.E.		20		20		15	ns
t_{pZL}	2.7	$\overline{RD}$,$\overline{WR}$ TRI-STATE to Low Level	After $\overline{RWEN}$/SYNC F.E.		25		25		18	ns
t_{pZH}	2.7	$\overline{RD}$,$\overline{WR}$ TRI-STATE to High Level	After $\overline{RWEN}$/SYNC F.E.		25		25		18	ns
WAIT STATES & CYCLE HOLD ($\overline{CWAIT}$, $\overline{WAITn}$, $\overline{PER}$ & $\overline{RDY}$) TIMING										
$t_{CWs(H)}$	2.8	$\overline{CWAIT}$ Setup Time (Cycle Hold)	Before PHI1 R.E.	35		30		20		ns
$t_{CWh(H)}$	2.8	$\overline{CWAIT}$ Hold Time (Cycle Hold)	After PHI1 R.E.	0		0		0		ns
$t_{CWs(W)}$	2.8/9	$\overline{CWAIT}$ Setup Time (Wait States)	Before PHI2 R.E.	10		10		6		ns
$t_{CWh(W)}$	2.9	$\overline{CWAIT}$ Hold Time (Wait States)	After PHI2 R.E.	20		20		15		ns
t_{Ws}	2.9	$\overline{WAITn}$ Setup Time	Before PHI2 R.E.	7		7		5		ns
t_{Wh}	2.9	$\overline{WAITn}$ Hold Time	After PHI2 R.E.	25		20		15		ns
t_{Ps}	2.10	$\overline{PER}$ Setup Time	Before PHI1 R.E.	0		0		0		ns
t_{Ph}	2.10	$\overline{PER}$ Hold Time	After PHI1 R.E.	30		30		30		ns
t_{Rd}	2.8/9/10	$\overline{RDY}$ Delay	After PHI2 R.E.		30		25		12	ns
SYNCHRONIZATION (SYNC) TIMING										
t_{Sys}	2.11	SYNC Setup Time	Before XIN R.E.	6		6		6		ns
t_{Syh}	2.11	SYNC Hold Time	After XIN R.E.	3		0		0		ns
t_{CS}	2.11	CTTL/SYNC Inversion Delay	CTTL (master) to $\overline{RWEN}$/SYNC (slave)		15		10		7	ns

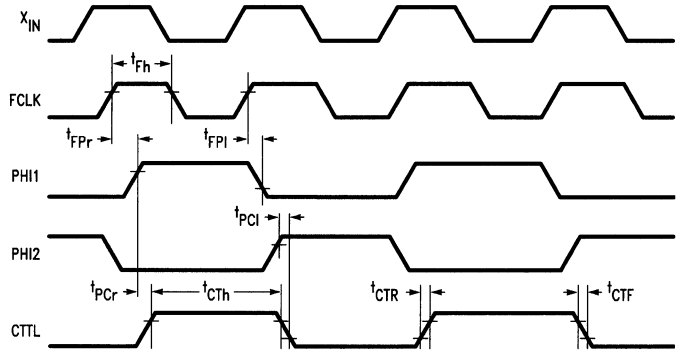
2.0 Device Specifications (Continued)

2.4.4 Timing Diagrams



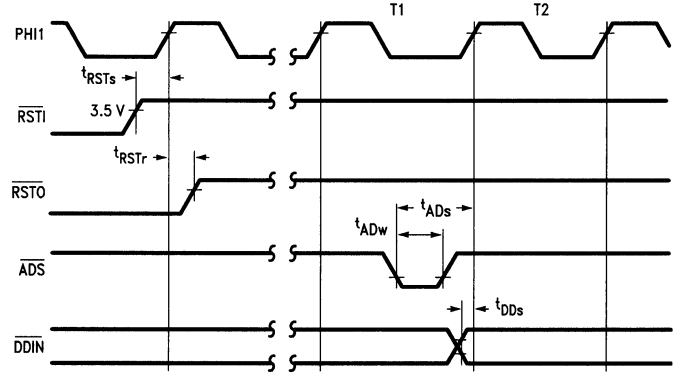
TL/EE/8524–21

FIGURE 2-2. Clock Signals (a)



TL/EE/8524–22

FIGURE 2-3. Clock Signals (b)

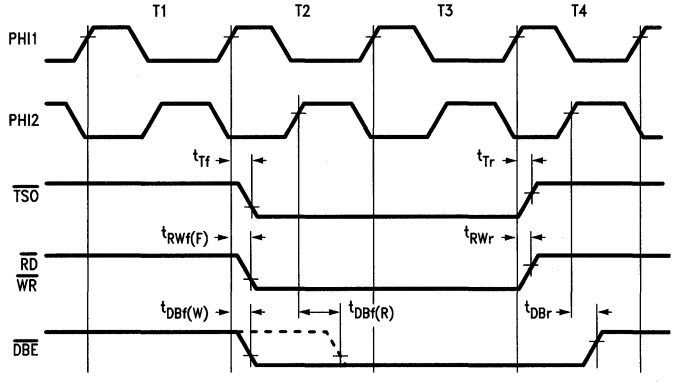


TL/EE/8524–23

FIGURE 2-4. Control Inputs

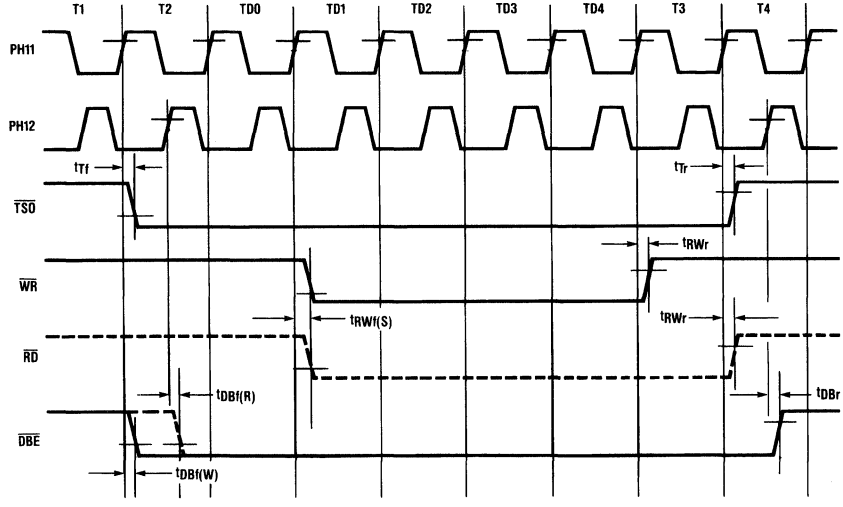
4

2.0 Device Specifications (Continued)



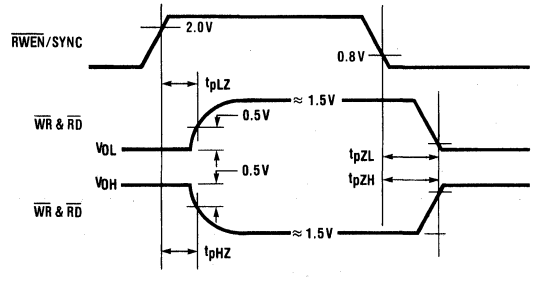
TL/EE/8524–24

FIGURE 2-5. Control Outputs (Fast Cycle)



TL/EE/8524–25

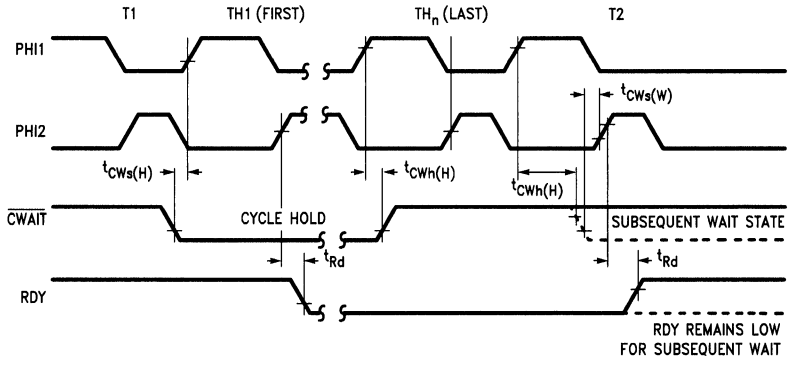
FIGURE 2-6. Control Outputs (Peripheral Cycle)



TL/EE/8524–26

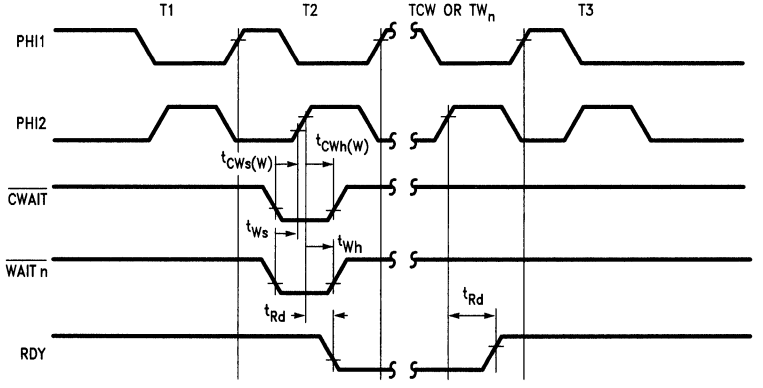
FIGURE 2-7. Control Outputs (TRI-STATE Timing)

2.0 Device Specifications (Continued)



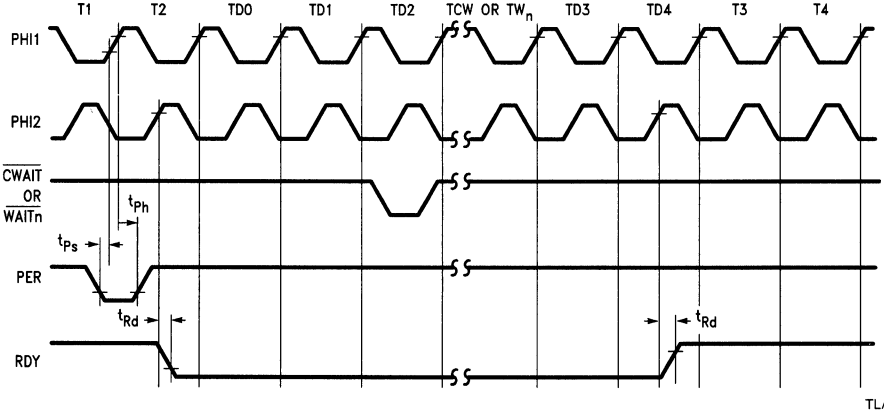
TL/EE/8524-27

FIGURE 2-8. Cycle Hold



TL/EE/8524-28

FIGURE 2-9. Wait State (Fast Cycle)

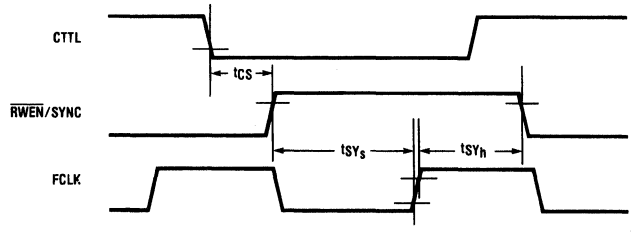


TL/EE/8524-29

FIGURE 2-10. Wait State (Peripheral Cycle)

4

2.0 Device Specifications (Continued)



TL/EE/8524–30

FIGURE 2-11. Synchronization Timing

National Semiconductor Corporation

NS32201-6/NS32201-8/NS32201-10 Timing Control Units

General Description

The NS32201 Timing Control Unit (TCU) is a 24-pin device fabricated on a Schottky bipolar process. It provides a two-phase clock, system control logic and cycle extension logic for the Series 32000® microprocessor family. The TCU input clock can be provided by either a crystal or an external clock signal whose frequency is twice the system clock frequency.

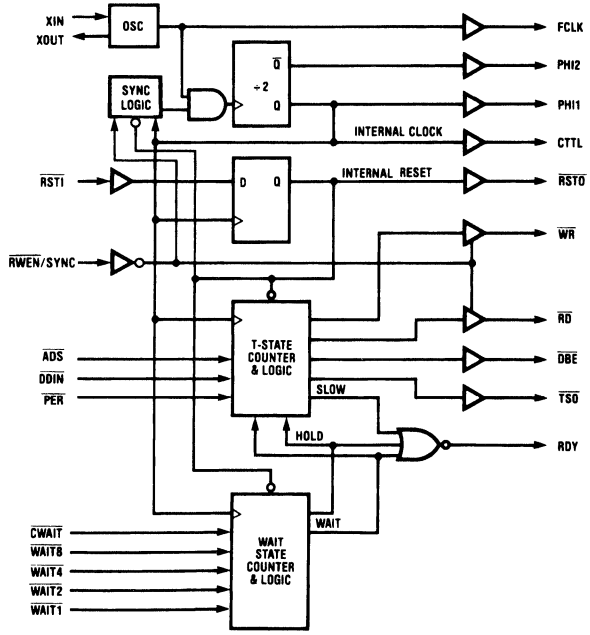
In addition to the two-phase clock for the CPU and MMU (PHI1 and PHI2), it also provides two system clocks for general use within the system (FCLK and CTTL). FCLK is a fast clock whose frequency is the same as the input clock, while CTTL is a TTL replica of PHI1 clock.

The system control logic and cycle extension logic make the TCU very attractive by providing extremely accurate bus control signals, and allowing extensive control over the bus cycle timing.

Features

- Oscillator at twice the CPU clock frequency
- 2 phase full V_{CC} swing high capacitance clock drivers (PHI1 and PHI2)
- 4-bit input ($\overline{WAITn}$) allowing precise specification of 0 to 15 wait states
- Cycle Hold for system arbitration and/or memory refresh
- System timing (FCLK, CTTL) and control ($\overline{RD}$, $\overline{WR}$, and $\overline{DBE}$) outputs
- General purpose Timing State Output ($\overline{TSO}$) that identifies internal states
- Peripheral cycle to accomodate slower MOS peripherals
- Provides "ready" (RDY) output for the Series 32000 CPUs
- Synchronous system reset generation from Schmitt trigger input
- Phase synchronization to a reference signal
- Single 5V power supply
- 24-pin dual-in-line package

Block Diagram



TL/EE/5590–2

Table of Contents

1.0 FUNCTIONAL DESCRIPTION

1.1 Power and Grounding
1.2 Crystal Oscillator Characteristics
1.3 Clocks
1.4 Resetting
1.5 Synchronizing Two or More TCUs
1.6 Bus Cycles
1.7 Bus Cycle Extension
1.7.1 Normal Wait States
1.7.2 Peripheral Cycle
1.7.3 Cycle Hold
1.8 Bus Cycle Extension Combinations
1.9 Overriding $\overline{\text{WAIT}}$ Wait States

2.0 DEVICE SPECIFICATIONS

2.1 Pin Descriptions
2.1.1 Supplies
2.1.2 Input Signals
2.1.3 Output Signals
2.2 Absolute Maximum Ratings
2.3 Electrical Characteristics
2.4 Switching Characteristics
2.4.1 Definitions
2.4.2 Output Load Circuits
2.4.3 Timing Tables
2.4.4 Timing Diagrams

List of Illustrations

Crystal Connection 1-1
PHI1 and PHI2 Clock Signals 1-2
Recommended Reset Connections (Non Memory-Managed System) 1-3a
Recommended Reset Connections (Memory-Managed System) 1-3b
Slave TCU does not use $\overline{\text{RWEN}}$ during Normal Operation 1-4a
Slave TCU Uses Both SYNC and $\overline{\text{RWEN}}$ 1-4b
Synchronizing Two TCUs 1-5
Synchronizing One TCU to an External Pulse 1-6
Basic TCU Cycle (Fast Cycle) 1-7
Wait State Insertion Using $\overline{\text{CWAIT}}$ (Fast Cycle) 1-8
Wait State Insertion Using $\overline{\text{WAITn}}$ (Fast Cycle) 1-9
Peripheral Cycle 1-10
Cycle Hold Timing Diagram 1-11
Fast Cycle with 12 Wait States 1-12
Peripheral Cycle with Six Wait States 1-13
Cycle Hold with Three Wait States 1-14
Cycle Hold of a Peripheral Cycle 1-15
Overriding WAITn Wait States 1-16
Connection Diagram 2-1
Clock Signals (a) 2-2
Clock Signals (b) 2-3
Control Inputs 2-4
Control Outputs (Fast Cycle) 2-5
Control Outputs (Peripheral Cycle) 2-6
Control Outputs (TRI-STATE Timing) 2-7
Cycle Hold 2-8
Wait States (Fast Cycle) 2-9
Wait States (Peripheral Cycle) 2-10
Synchronization Timing 2-11

1.0 Functional Description

1.1 POWER AND GROUNDING

The NS32201 requires a single +5V power supply, applied to pin 24 (V_{CC}). See D.C. Electrical Characteristics. The Logic Ground on pin 12 (GND), is the common pin for the TCU.

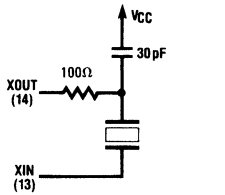
A 0.1 μf, ceramic decoupling capacitor must be connected across V_{CC} and GND, as close to the TCU as possible.

1.2 CRYSTAL OSCILLATOR CHARACTERISTICS

The NS32201 has a "Pierce"-type oscillator. Connections of the crystal and bias components to XIN and XOUT are shown in *Figure 1-1.* It is important that the crystal and the RC components be mounted in close proximity to the XIN, XOUT and V_{CC} pins to keep printed circuit trace lengths to an absolute minimum.

Typical Crystal Specifications:

Type ..... At-Cut
Tolerance ..... 0.005% at 25°C
Stability ..... 0.01% from 0° to 70°C
Resonance ..... Fundamental parallel
Capacitance ..... 20 pF
Maximum Series Resistance 50Ω



TL/EE/5590-3

CRYSTAL FREQUENCY (MHz)	R (OHM)
6–12	470
12–18	220
18–24	100
24–30	47

FIGURE 1-1. Crystal Connection Diagram

1.3 CLOCKS

The NS32201 TCU has four clock output pins. The PHI1 and PHI2 clocks are required by the Series 32000 CPUs. These clocks are non-overlapping as shown in *Figure 1-2.*

Each rising edge of PHI1 defines a transition in the timing state of the CPU.

As the TCU generates the various clock signals with very short transition timings, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible. It is also recommended that only the Series 32000 CPU and, if used, the MMU (Memory Management Unit) be connected to the PHI1 and PHI2 clocks. In addition to the CPU and MMU, 25 pF ceramic capacitors from these pins to ground are recommended as they provide a better VOH on the outputs. These capacitors should be mounted close to the TCU to minimize trace inductances.

CTTL is a TTL compatible clock signal which runs at the same frequency as PHI1 and is closely balanced with it. CTTL is intended for driving TTL loads.

FCLK is also a TTL compatible clock, running at the frequency of XIN input. This clock is also intended for driving TTL loads and has a frequency that is twice the CTTL clock frequency. The exact phase relationship between PHI1, PHI2, CTTL and FLCK can be found in Section 2.

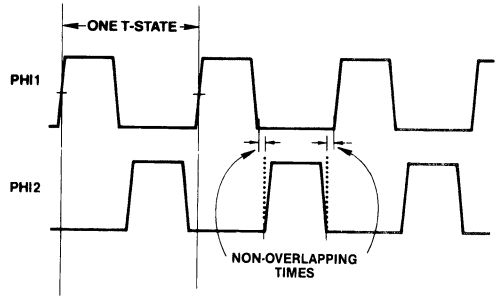
1.4 RESETTING

The NS32201 TCU provides circuitry to meet the reset requirements of the Series 32000 CPUs. If the Reset Input line, $\overline{RSTI}$ is pulled low, the TCU asserts $\overline{RSTO}$ which resets the Series 32000 CPU. This Reset Output may also be used as a system reset signal. *Figure 1-3a* illustrates the reset connections for a non Memory-Managed system. *Figure 1-3b* illustrates the reset connections for a Memory-Managed system.

1.5 SYNCHRONIZING TWO OR MORE TCUs

During reset, (when $\overline{RSTO}$ is low), one or more TCUs can be synchronized with a reference (Master) TCU. The $\overline{RWEN}$/SYNC input to the slave TCU(s) is used for synchronization. The Slave TCU samples the $\overline{RWEN}$/SYNC input on the rising edge of FCLK when $\overline{RSTO}$ is low and CTTL is high (see *Figure 1-5*). If $\overline{RWEN}$/SYNC is sampled high, the phase of CTTL of the Slave TCU is shifted by one XIN clock cycle.

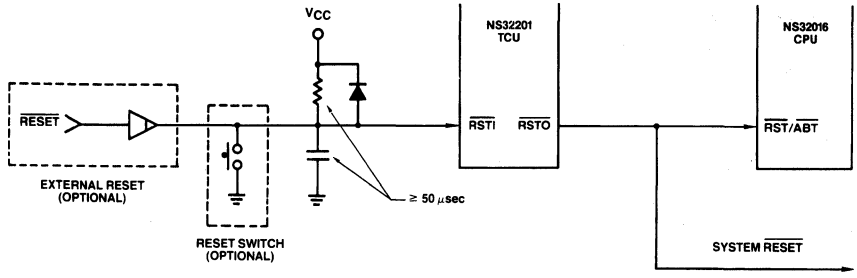
Two possible circuits for TCU synchronization are illustrated in *Figures 1-4a* and *1-4b.* It should be noted that when $\overline{RWEN}$/SYNC is high, the $\overline{RD}$ and $\overline{WR}$ signals will be TRI-STATE on the slave TCU.



TL/EE/5590-4

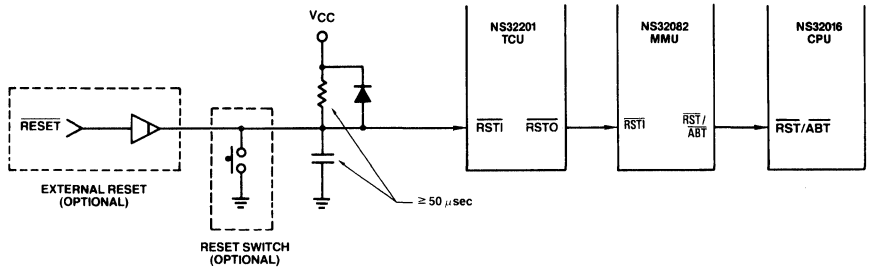
FIGURE 1.2. PHI1 and PHI2 Clock Signals

1.0 Functional Description (Continued)



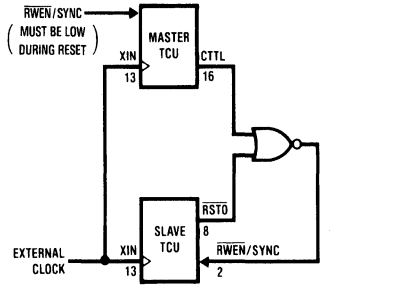
TL/EE/5590–5

FIGURE 1-3a. Recommended Reset Connections (Non Memory-Managed System)



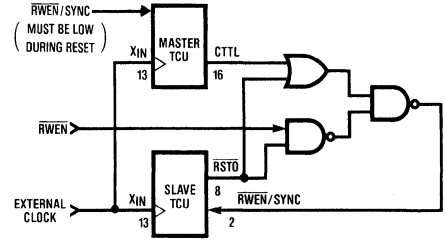
TL/EE/5590–6

FIGURE 1-3b. Recommended Reset Connections (Memory-Managed System)



TL/EE/5590–7

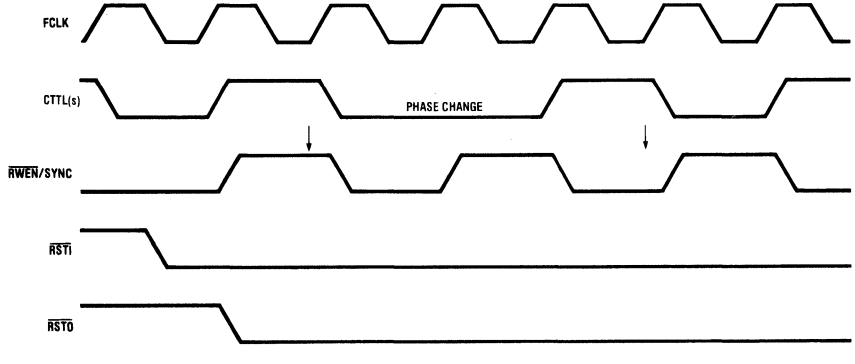
FIGURE 1-4a. Slave TCU does not use RWEN during Normal Operation



TL/EE/5590–8

FIGURE 1-4b. Slave TCU Uses Both SYNC and RWEN

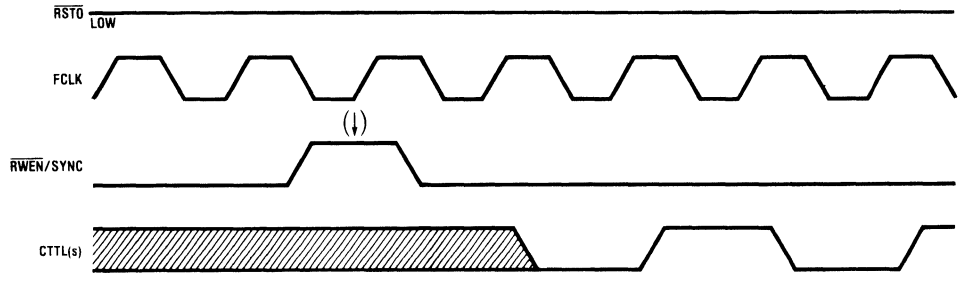
Note: When two or more TCUs are to be synchronized, the XIN of all the TCUs should be connected to an external clock source. For details on the external clock, see Switching Characteristics in Section 2.



TL/EE/5590–9

FIGURE 1-5. Synchronizing Two TCUs

1.0 Functional Description (Continued)



TL/EE/5590–10

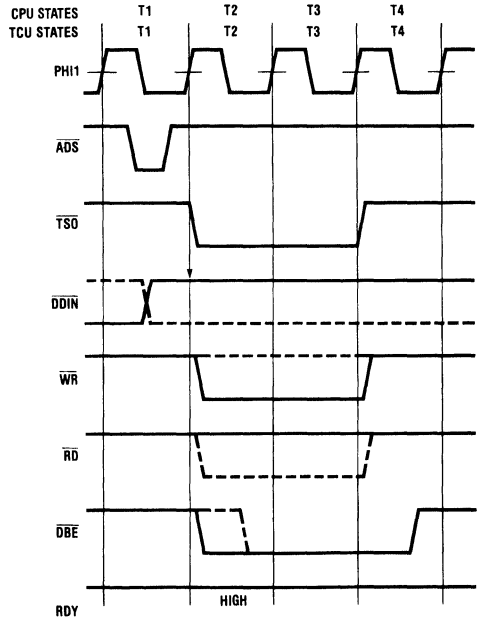
FIGURE 1-6. Synchronizing one TCU to an External Pulse

In addition to synchronizing two or more TCUs, the $\overline{\text{RWEN}}$/SYNC input can be used to "fix" the phase of one TCU to an external pulse. The pulse to be used must be high for only one rising edge of FCLK. Independent of CTTL's state at the FCLK rising edge, its state following the next FCLK rising edge should be low. *Figure 1-6* shows the timing of this sequence.

1.6 BUS CYCLES

In addition to providing all the necessary clock signals, the NS32201 TCU provides bus control signals to the system. The TCU senses the $\overline{\text{ADS}}$ signal from the CPU or MMU to start a bus cycle. The $\overline{\text{DDIN}}$ input signal is also sampled to determine whether a Read or Write cycle is to be generated. In addition to $\overline{\text{RD}}$ and $\overline{\text{WR}}$, other signals are provided: $\overline{\text{DBE}}$ and $\overline{\text{TSO}}$. $\overline{\text{DBE}}$ is used to enable data buffers. The leading edge of $\overline{\text{DBE}}$ is delayed a half clock period during Read cycles to avoid bus conflicts between data buffers and either the CPU or the MMU. This is shown in *Figure 1-7*.

The Timing State Output ($\overline{\text{TSO}}$) is a general purpose signal that may be used by external logic for synchronizing to a System cycle. $\overline{\text{TSO}}$ is activated at the beginning of state T2 and returns to the high level at the beginning of state T4 of the CPU cycle. $\overline{\text{TSO}}$ can be used to gate the $\overline{\text{CWAIT}}$ signal when continuous waits are required. Another application of $\overline{\text{TSO}}$ is the control of interface circuitry for dynamic RAMs.



TL/EE/5590–11

FIGURE 1-7. Basic TCU Cycle (Fast Cycle)

Notes:

1. The CPU and TCU view some timing states (T-states) differently. For clarity, references to T-states will sometimes be followed by (TCU) or (CPU). (CPU) also implies (MMU).
2. Arrows indicate when the TCU samples the input.
3. $\overline{\text{RWEN}}$ is assumed low ($\overline{\text{RD}}$ and WR enabled) unless specified differently.
4. For clarity, T-states for both the TCU and CPU are shown above the diagrams. (See Note 1.)

1.0 Functional Description (Continued)

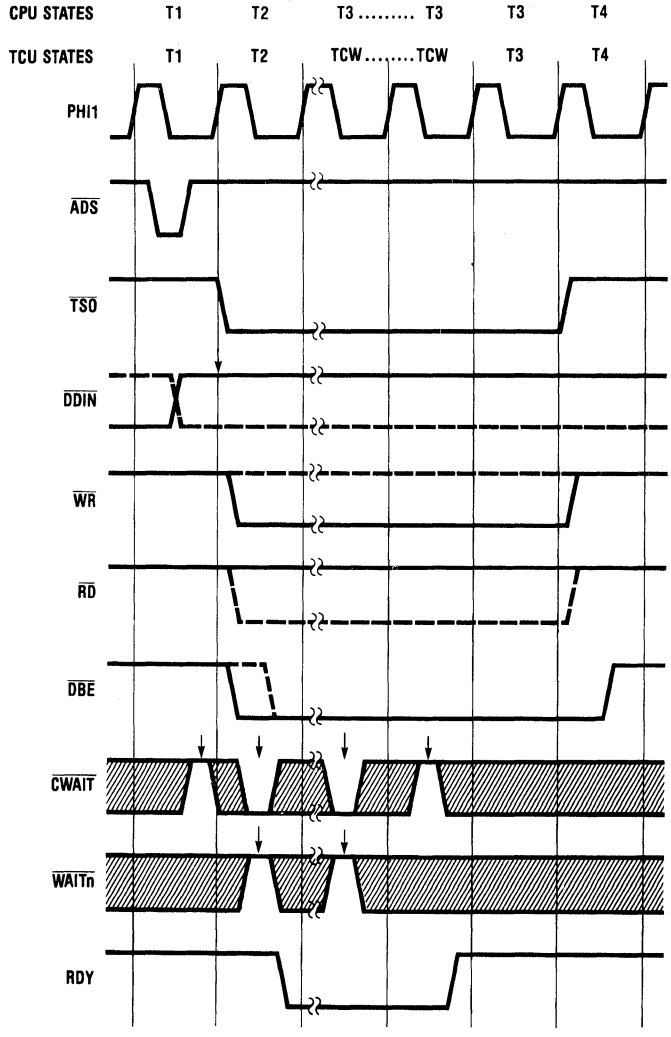
1.7 BUS CYCLE EXTENSION

The NS32201 TCU uses the Wait input signals to extend normal bus cycles. A normal bus cycle consists of four PHI1 clock cycles. Whenever one or more Wait inputs to the TCU are activated, a bus cycle is extended by at least one PHI1 clock cycle. The purpose is to allow the CPU to access slow memories or peripherals. The TCU responds to the Wait signals by pulling the RDY signal low as long as Wait States are to be inserted in the Bus cycle.

There are three basic cycle extension modes provided by the TCU, as described below.

1.7.1 Normal Wait States

This is a normal Wait State insertion mode. It is initiated by pulling $\overline{\text{CWAIT}}$ or any of the $\overline{\text{WAITn}}$ lines low in the middle of T2. *Figure 1-8* shows the timing diagram of a bus cycle when $\overline{\text{CWAIT}}$ is sampled high at the end of T1 and low in the middle of T2.



TL/EE/5590–12

FIGURE 1-8. Wait State Insertion Using $\overline{\text{CWAIT}}$ (Fast Cycle)

1.0 Functional Description (Continued)

The RDY signal goes low during T2 and remains low until $\overline{CWAIT}$ is sampled high by the TCU. RDY is pulled high by the TCU during the same PHI1 cycle in which the $\overline{CWAIT}$ line is sampled high.

If any of the $\overline{WAITn}$ signals are sampled low during T2 and $\overline{CWAIT}$ is high during the entire bus cycle, then the RDY line goes low for 1 to 15 clock cycles, depending on the binary weighted value of $\overline{WAITn}$. If, for example, $\overline{WAIT1}$ and $\overline{WAIT4}$ are sampled low, then five wait states will be inserted. This is shown in *Figure 1-9*.

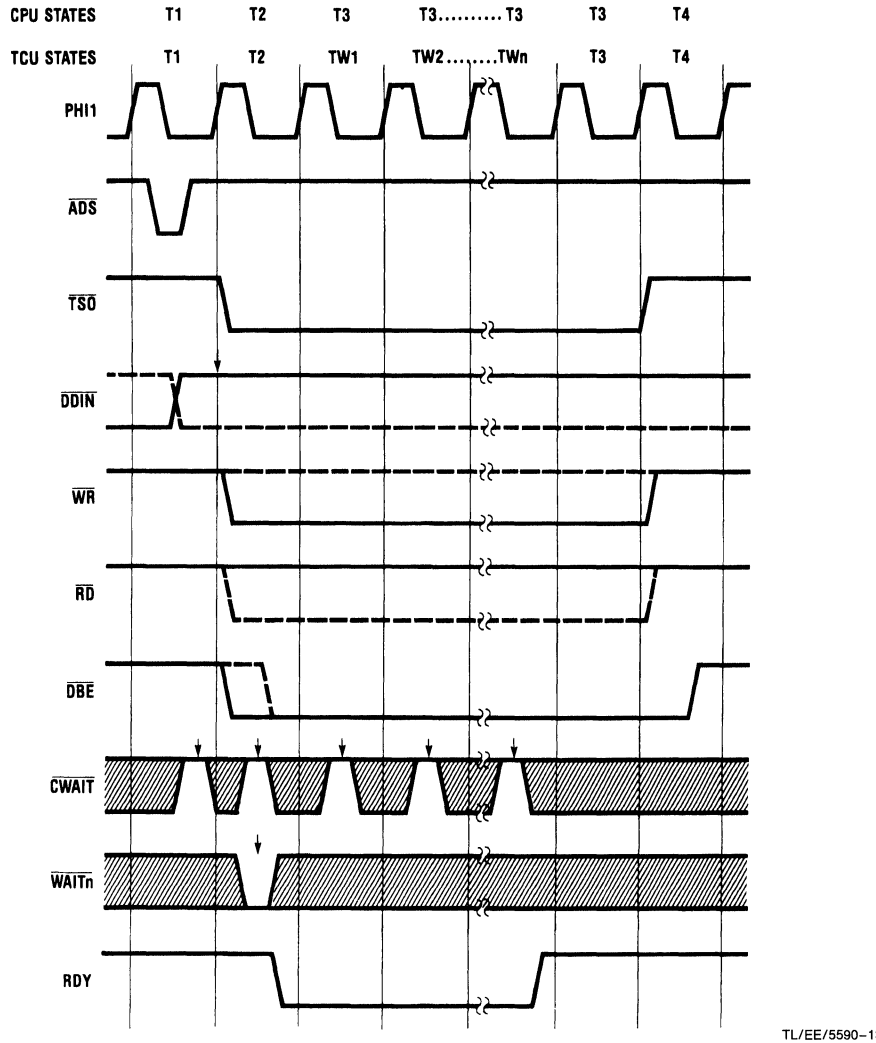


FIGURE 1-9. Wait State Insertion Using $\overline{WAITn}$ (Fast Cycle)

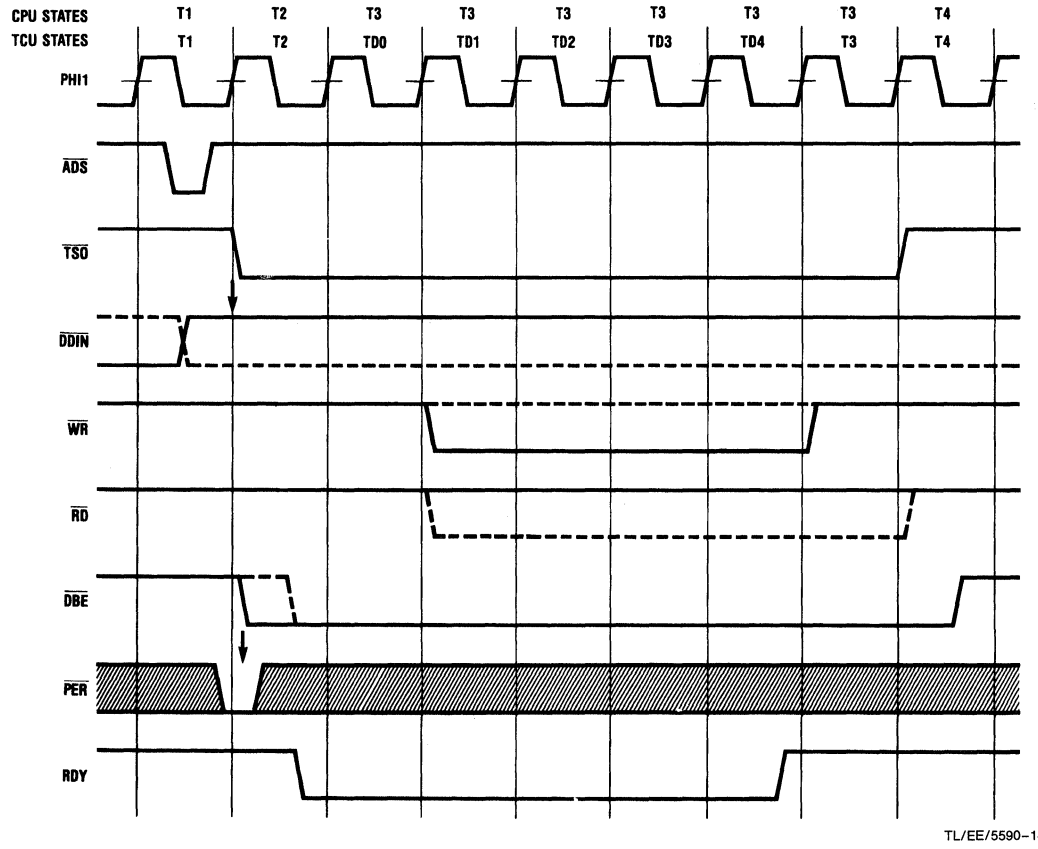
1.0 Functional Description (Continued)

1.7.2 Peripheral Cycle

This cycle is entered when the $\overline{PER}$ signal line is sampled low at the beginning of T2. The TCU adds five wait states identified as TD0-TD4 into a normal bus cycle. The $\overline{RD}$ and $\overline{WR}$ signals are also re-shaped so the setup and hold times for address and data will be increased.

This may be necessary when slower peripherals must be accessed.

Figure 1-10 shows the timing diagram of a peripheral cycle.



TL/EE/5590–14

FIGURE 1-10. Peripheral Cycle

1.0 Functional Description (Continued)

1.7.3 Cycle Hold

If the $\overline{\text{CWAIT}}$ input is sampled low at the end of state T1, the TCU will go into cycle hold mode and stay in this mode for as long as $\overline{\text{CWAIT}}$ is kept low. During this mode the control signals $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{TSO}}$ and $\overline{\text{DBE}}$ are kept inactive; RDY is pulled low, thus causing wait states to be inserted into the bus cycle. The cycle hold feature can be used in applications involving dynamic RAMs. A timing diagram showing the cycle hold feature is shown in *Figure 1-11*.

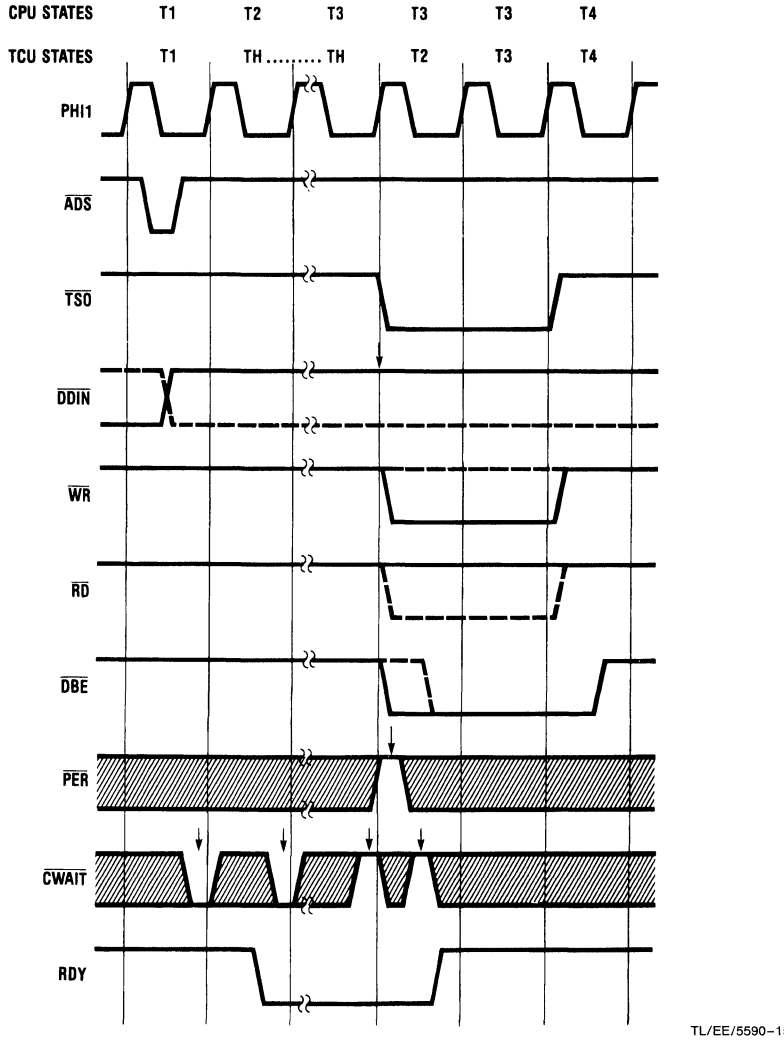


FIGURE 1-11. Cycle Hold Timing Diagram

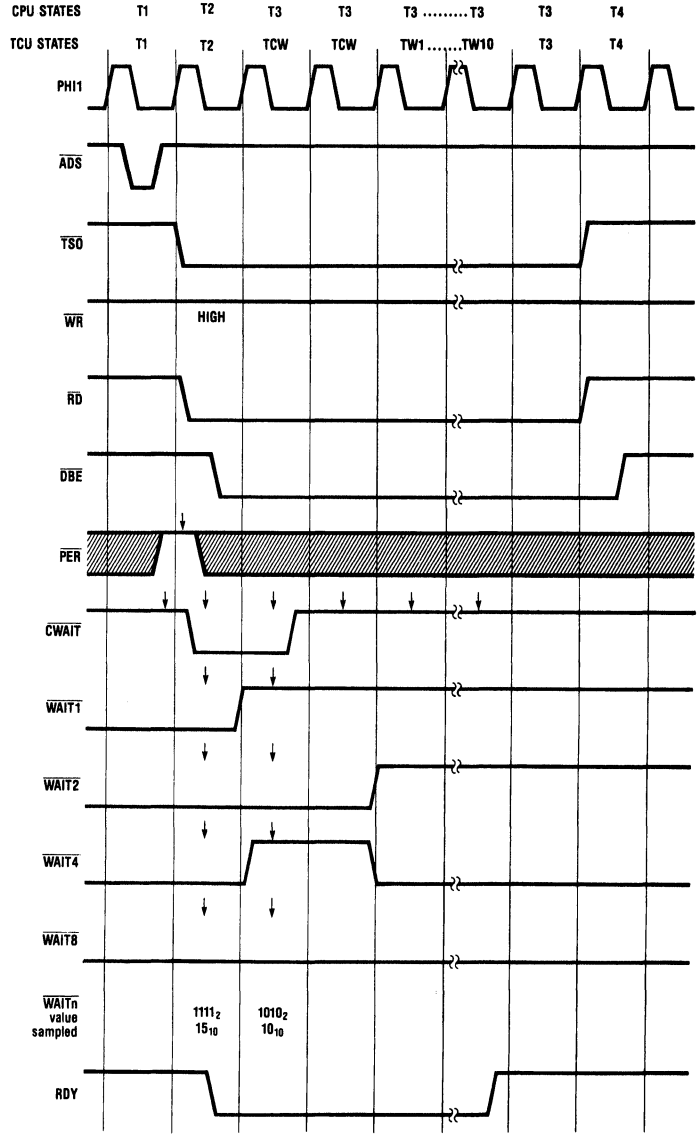
1.8 BUS CYCLE EXTENSION COMBINATIONS

Any combination of the TCU input signals used for extending a bus cycle can be activated at one time. The TCU will honor all of the requests according to a certain priority scheme. A cycle hold request is assigned top priority. It follows a peripheral cycle request, and then $\overline{\text{CWAIT}}$ and $\overline{\text{WAITn}}$ respectively.

If, for example, all the input signals $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$ and $\overline{\text{WAITn}}$ are asserted at the beginning of the cycle, the TCU will enter the cycle hold mode. As soon as $\overline{\text{CWAIT}}$ goes high, the input signal $\overline{\text{PER}}$ is sampled to determine whether a peripheral cycle is requested.

Next, the TCU samples $\overline{\text{CWAIT}}$ again and $\overline{\text{WAITn}}$ to check whether additional wait states have to be inserted into the bus cycle. This sampling point depends on whether $\overline{\text{PER}}$ was sampled high or low. If $\overline{\text{PER}}$ was sampled high, then the sampling point will be in the middle of the TCU state T2, (*Figure 1-14*), otherwise it will occur three clock cycles later (*Figure 1-15*). *Figures 1-12* to *1-15* show the timing diagrams for different combinations of cycle extensions.

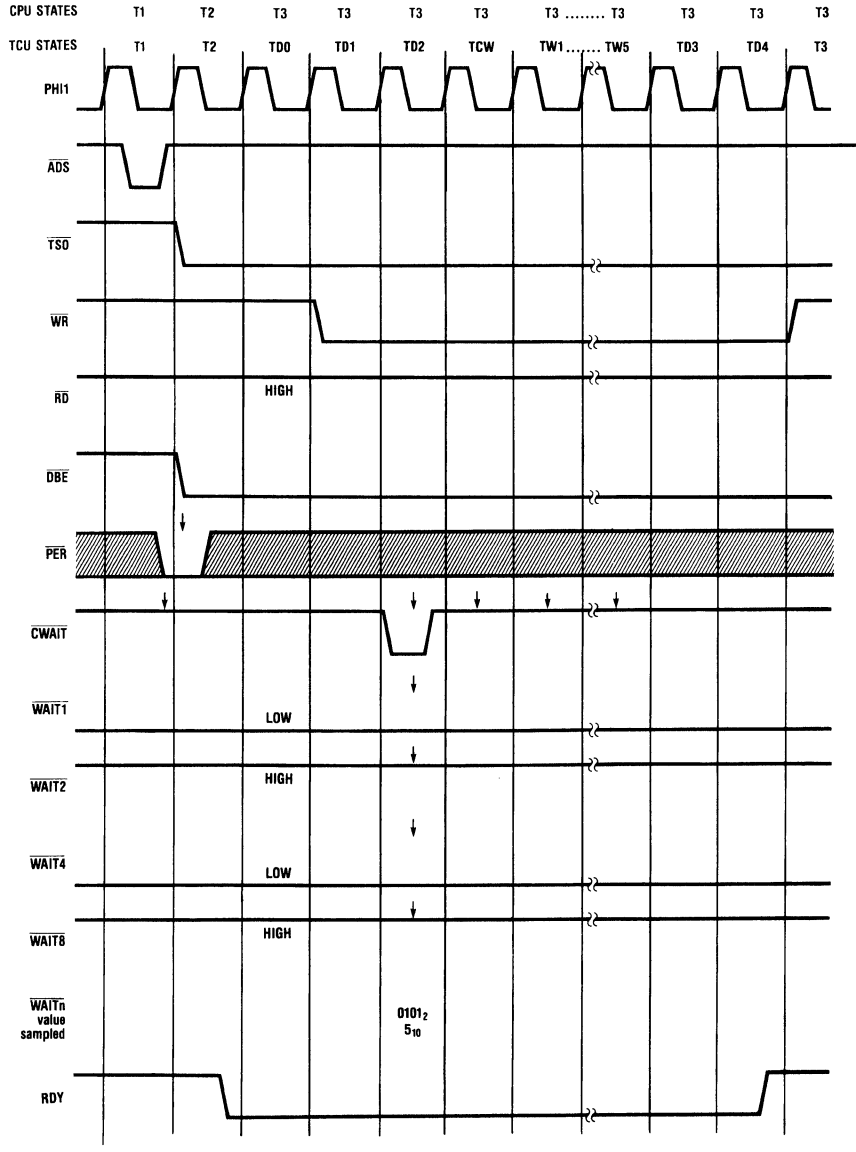
1.0 Functional Description (Continued)



TL/EE/5590–16

FIGURE 1-12. Fast Cycle With 12 Wait States (2 CWAIT and WAIT10) (Read Cycle)

1.0 Functional Description (Continued)

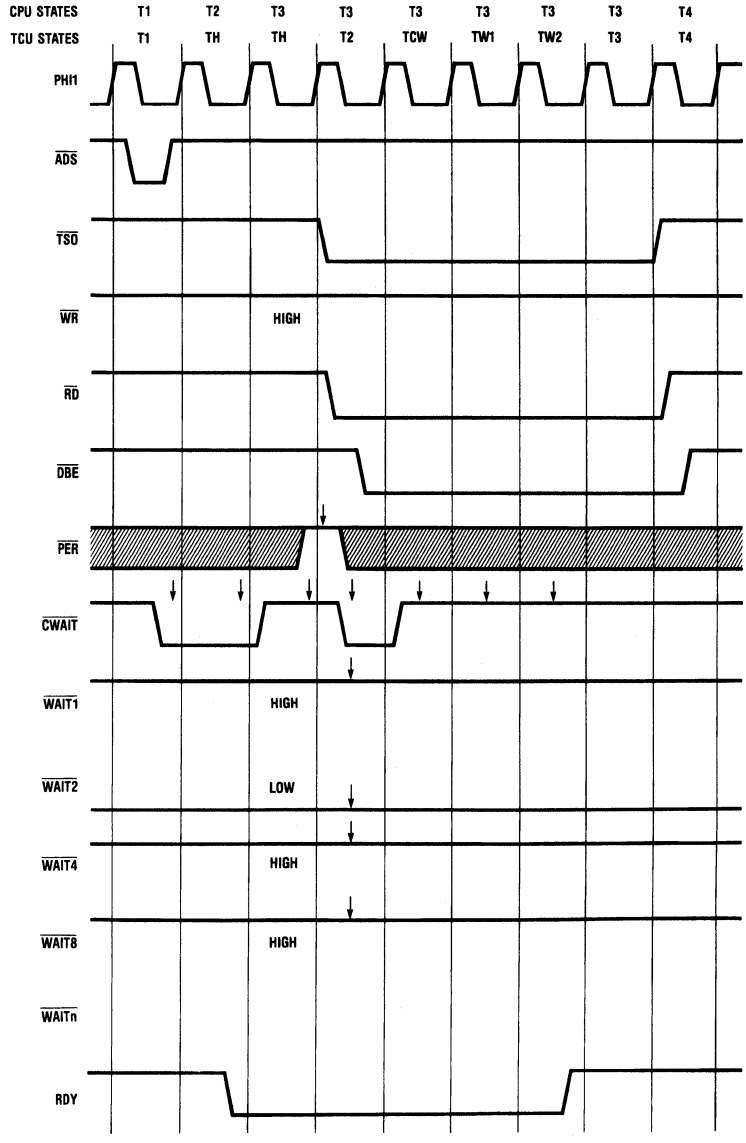


TL/EE/5590–17

FIGURE 1-13. Peripheral Cycle With Six Wait States (1 CWAIT and WAIT5) (Write Cycle)

4

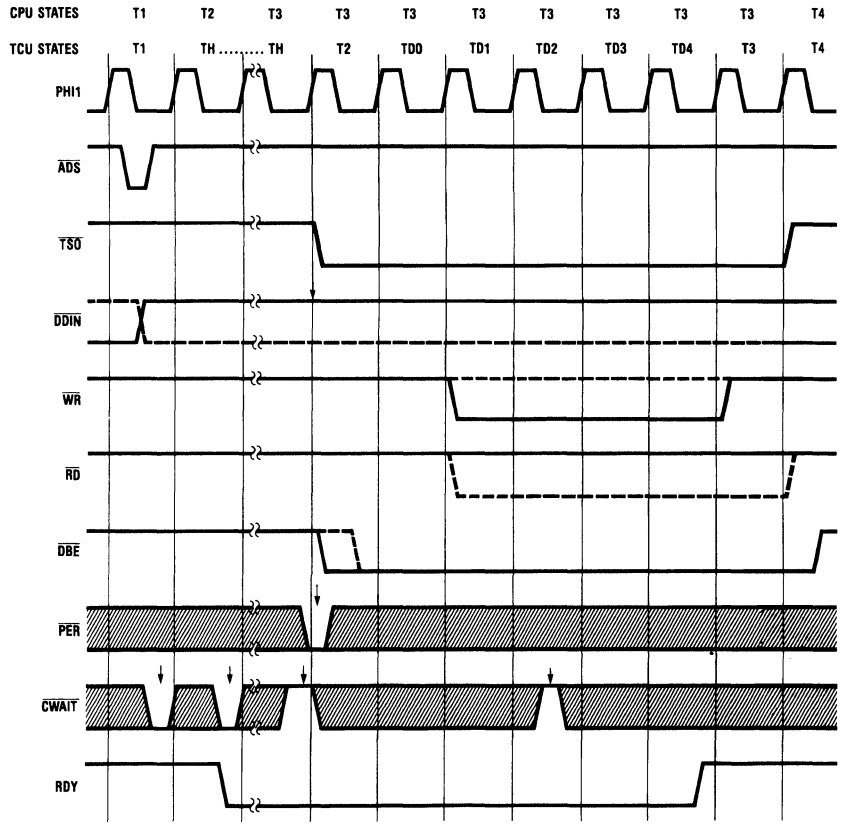
1.0 Functional Description (Continued)



TL/EE/5590–18

FIGURE 1-14. Cycle Hold With Three Wait States (1 $\overline{\text{CWAIT}}$ and $\overline{\text{WAIT2}}$) (Read Cycle)

1.0 Functional Description (Continued)



TL/EE/5590–19

FIGURE 1-15. Cycle Hold of a Peripheral Cycle

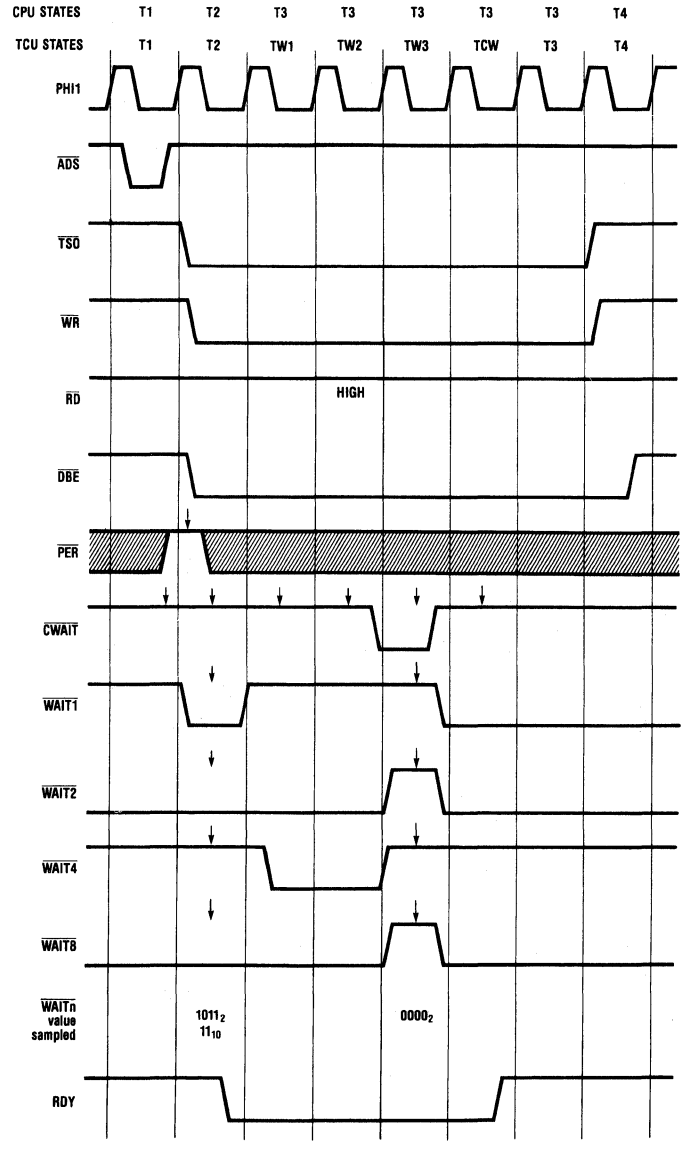
1.9 OVERRIDING WAITn Wait STATES

The TCU handles the WAITn Wait States by means of an internal counter that is reloaded with the binary value corresponding to the state of the WAITn inputs each time CWAIT is sampled low, and is decremented when CWAIT is high.

This allows to either extend a bus cycle of a predefined number of clock cycles, or prematurely terminate it. To terminate a bus cycle, for example, CWAIT must be asserted for at least one clock cycle, and the WAITn inputs must be forced to their inactive state.

At least one wait state is always inserted when using this procedure as a result of CWAIT being sampled low. *Figure 1-16* shows the timing diagram of a prematurely terminated bus cycle where eleven wait states were being inserted.

1.0 Functional Description (Continued)



TL/EE/5590–20

FIGURE 1-16. Overriding $\overline{\text{WAITn}}$ Wait States (Write Cycle)

2.0 Device Specifications

2.1 NS32201 PIN DESCRIPTIONS

The following is a description of all NS32201 pins. The descriptions reference portions of the Functional Description, Chapter 2.

2.1.1 SUPPLIES

Power (V_{CC}): +5V positive supply. Sec. 1.1.

Ground (GND): Power supply return. Sec. 1.1.

2.1.2 INPUT SIGNALS

Reset Input ($\overline{RSTI}$): Active low. Schmitt triggered, asynchronous signal used to generate a system reset. Sec. 1.4.

Address Strobe ($\overline{ADS}$): Active low. Identifies the first timing state (T1) of a bus cycle.

Data Direction Input ($\overline{DDIN}$): Active low. Indicates the direction of the data transfer during a bus cycle. Implies a Read when low and a Write when high.

Read/Write Enable and Synchronization ($\overline{RWEN}$/SYNC): TRI-STATE® the $\overline{RD}$ and the $\overline{WR}$ outputs when high and enables them when low. Also used to synchronize the phase of the TCU clock signals, when two or more TCUs are used. Sec. 1.5.

Crystal or External Clock Source (XIN): Input from a crystal or an external clock source. Sec. 1.3.

Continuous Wait ($\overline{CWAIT}$): Active low. Initiates a continuous wait if sampled low in the middle of T2 during a fast cycle, or in the middle of TD2 during a peripheral cycle. If $\overline{CWAIT}$ is low at the end of T1, it initiates a Cycle Hold. Sec. 1.7.1.

Four-Bit Wait State Inputs ($\overline{WAIT1}$, $\overline{WAIT2}$, $\overline{WAIT4}$ and $\overline{WAIT8}$): Active low. These inputs, (collectively called $\overline{WAIT}$n), allow from zero to fifteen wait states to be specified. They are binary weighted. Sec. 1.7.1.

Peripheral Cycle ($\overline{PER}$): Active low. If active, causes the TCU to insert five wait states into a normal bus cycle. It also causes the Read and Write signals to be re-shaped to meet the setup and hold timing requirement of slower MOS peripherals. Sec. 1.7.2.

2.1.3 OUTPUT SIGNALS

Reset Output ($\overline{RSTO}$): Active low. This signal becomes active when $\overline{RSTI}$ is low, initiating a system reset. $\overline{RSTO}$ goes high on the first rising edge of PHI1 after $\overline{RSTI}$ goes high. Sec. 1.4.

Read Strobe ($\overline{RD}$): (TRI-STATE) Active low. Identifies a Read cycle. It is decoded from $\overline{DDIN}$ and TRI-STATE by $\overline{RWEN}$/SYNC. Sec. 1.6.

Write Strobe ($\overline{WR}$): (TRI-STATE) Active low. Identifies a Write cycle. It is decoded from $\overline{DDIN}$ and TRI-STATE by $\overline{RWEN}$/SYNC. Sec. 1.6.

NOTE: $\overline{RD}$ and $\overline{WR}$ are mutually exclusive in any cycle. Hence they are never low at the same time.

Data Buffer Enable ($\overline{DBE}$): Active low. This signal is used to control the data bus buffers. It is low when the data buffers are to be enabled. Sec. 1.6.

Timing State Output ($\overline{TSO}$): Active low. The falling edge of $\overline{TSO}$ signals the beginning of state T2 of a bus cycle. The rising edge of $\overline{TSO}$ signals the beginning of state T4. Sec. 1.6.

Ready (RDY): Active high. This signal will go low and remain low as long as wait states are to be inserted in a bus cycle. It is normally connected to the RDY input of the CPU. Sec. 1.7.

Fast Clock (FCLK): This is a TTL level clock running at the same frequency as the crystal or the external source. Its frequency is twice that of the CPU clocks. Sec. 1.3.

CPU Clocks (PHI1 and PHI2): These outputs provide the Series 32000 CPU with two phase, non-overlapping clock signals. Their frequency is half that of the crystal or external source. Sec. 1.3.

TTL System Clock (CTTL): This is a TTL compatible version of the PHI1 clock. Hence, it operates at the CPU clock frequency. Sec. 1.3.

Crystal Output (XOUT): This line is used as the return path for the crystal (if used). It must be left open when an external clock source is used to drive XIN. Sec. 1.2.

2.0 Device Specifications (Continued)

2.2 ABSOLUTE MAXIMUM RATINGS (Note 1)

Supply Voltage	7V
Input Voltages	−1 to +5.5V
Output Voltages	−1 to +5.5V
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Continuous Power Dissipation at 25°C Free-Air (Note 1)	
Cavity Package	3030 mW
Molded Package	2840 mW

Note: *Absolute maximum ratings indicate limits beyond which permament damage may occur. Continuous operation at these limits is not intended, operation should be limited to those conditions specified under Electrial Characteristics.*

2.3 ELECTRICAL CHARACTERISTICS T_A = 0° to +70°C, V_{CC} = 5V ±5%, GND=0V

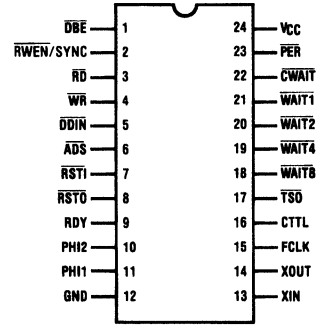
Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	All Inputs Except $\overline{RSTI}$ & XIN			0.8	V
V_{IH}	Input High Voltage	All Inputs Except $\overline{RSTI}$ & XIN	2			V
V_{T+}	$\overline{RSTI}$ Rising Threshold Voltage		0.50 V_{CC}	0.60 V_{CC}	0.70 V_{CC}	V
V_{HYS}	$\overline{RSTI}$ Hysteresis Voltage		0.10 V_{CC}	0.20 V_{CC}	0.25 V_{CC}	V
V_{IX}	XIN Input Threshold Voltage		0.40 V_{CC}	0.5 V_{CC}	0.60 V_{CC}	V
I_{IX}	XIN Input Load Current	0.5V ≤ V_{IN} ≤ 4.5V			±500	μA
I_{IL}	Input Low Current	V_{IN}=0.5V Except XIN			−500	μA
I_{IH}	Input High Current	V_{IN}=5.25V Except XIN			50	μA
V_{OL}	Output Low Voltage	PHI1 & PHI2 I = 1 mA	−0.5		0.3	V
		All Other Outputs Except XOUT I = 20 mA			0.5	
V_{OH}	Output High Voltage	PHI1 & PHI2 I = −1 mA	V_{CC}−0.3			V
		All Other Outputs Except XOUT I = −1 mA	2.4			
$I_{O(off)}$	Output Leakage Current On $\overline{RD}$ and $\overline{WR}$	0.4V ≤ V_{OUT} ≤ V_{CC}			±50	μA
V_{CLAMP}	Input Clamp Voltage	I_{IN} = −18 mA Except XIN		−0.7	−1.2	V
I_{CC}	Supply Current	fXIN = 20 MHz		180	260	mA

Note 1: For operation over 25°C free-air temperature, for cavity package, derate linearly to 2121 mW at 70°C at the rate of 20.2 mW/°C, and for molded package, derate linearly to 1818 mW at 70°C at the rate of 22.7 mW/°C.

Note 2: All typical values are for V_{CC} = 5V and T_A = 25°C.

Connection Diagram

Dual-In-Line Package



Top View

TL/EE/5590–1

FIGURE 2-1

Order Number NS32201D or NS32201N
See NS Package Number D24C or N24A

2.0 Device Specifications (Continued)

2.4 SWITCHING CHARACTERISTICS

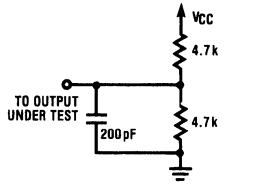
2.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 or PHI2, and to 0.8V or 2.0V on all TTL compatible signals, unless specifically stated otherwise.

ABBREVIATIONS

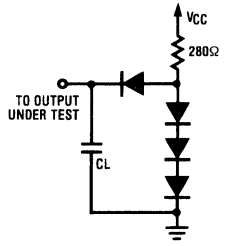
L.E.—Leading Edge
T.E.—Trailing Edge
R.E.—Rising Edge
F.E.—Falling Edge

2.4.2 Output Load Circuits (Notes 1, 2, 3)



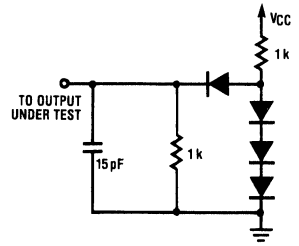
TL/EE/5590–21

LOAD 1



TL/EE/5590–22

LOAD 2



TL/EE/5590–23

LOAD 3

Note 1: Unless otherwise specified, the timing measurements are taken with the output pins in the following conditions.

Load 1 PHI1 and PHI2

Load 2 CL = 50 pF all TTL outputs

CL = 100 pF only CTTL

Load 3 $\overline{RD}$ and $\overline{WR}$ for TRI-STATE measurements only.

Note 2: Load Capacitance includes probe and jig capacitance.

Note 3: All diodes are 1N914 or equivalent.

2.4.3 Timing Tables

Name	Figure	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
				Min	Max	Min	Max	Min	Max	
CLOCK-SIGNALS (XIN, FCLK, PHI1 & PHI2) TIMING (Note 3)										
t_{Cp}	2.2	Clock Period	PHI1 R.E. to Next PHI1 R.E.	160		120		100		ns
t_{CLh}	2.2	Clock High Time	At V_{CC} −0.9V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp} −17 ns	0.5 t_{Cp} −7 ns	0.5 t_{Cp} −16 ns	0.5 t_{Cp} −7 ns	0.5 t_{Cp} −15 ns	0.5 t_{Cp} −7 ns	
t_{CLl}	2.2	Clock Low Time	At 0.8V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp}	0.5 t_{Cp} +12 ns	0.5 t_{Cp}	0.5 t_{Cp} +11 ns	0.5 t_{Cp}	0.5t_{CP} +10 ns	
$t_{CLw(1,2)}$	2.2	Clock Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp} −14 ns	0.5 t_{Cp} −4 ns	0.5 t_{Cp} −12 ns	0.5 t_{Cp} −4 ns	0.5 t_{Cp} −10 ns	0.5 t_{Cp} −4 ns	
t_{CLwas}		PHI1, PHI2 Asymmetry (t_{CLW1}) - $t_{CLW(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns
t_{CLR}	2.2	Clock Rise Time	0.8V to V_{CC} −0.9V on PHI1, PHI2 R.E.		9		8		7	ns
t_{CLF}	2.2	Clock Fall Time	V_{CC} −0.9V to 0.8V on PHI1, PHI2 F.E.		7		7		7	ns
$t_{nOVL(1,2)}$	2.2	Clock Nonoverlap Time	0.8V on PHI1, PHI2 F.E. to 0.8V on PHI2, PHI1 R.E.	0	5	0	5	0	5	ns
t_{nOVLas}		Non-Overlap Asymmetry ($t_{nOVL(1)}$ - $t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{XIR}	2.3	XIN Rise Time	1.5V to V_{CC} −1.5V on XIN R.E.		15		11		9	ns
t_{XIF}	2.3	XIN Fall Time	V_{CC} −1.5V to 1.5V on XIN F.E.		15		11		9	ns
t_{Xh}	2.2	XIN High Time (External Input)	2.5V on XIN R.E. to 2.5V on XIN F.E.	25		20		16		ns

2.0 Device Specifications (Continued)

2.4 SWITCHING CHARACTERISTICS (Continued)

2.4.3 Timing Tables

Name	Figure	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
				Min	Max	Min	Max	Min	Max	
t_{XI}	2.2	XIN Low Time (External Input)	2.5V on XIN F.E. to 2.5V on XIN R.E.	25		20		16		ns
t_{XFr}	2.2	XIN to FCLK R.E. Delay	2.5V on XIN R.E. to FCLK R.E.	15	29	15	28	15	27	ns
t_{XFf}	2.2	XIN to FCLK F.E. Delay	2.5V on XIN F.E. to FCLK F.E.	15	29	15	28	15	27	ns
t_{XCr}	2.2	XIN to CTTL R.E. Delay	2.5V on XIN R.E. to CTTL R.E.	24	40	24	39	24	35	ns
t_{XPr}	2.2	XIN to PHI1 R.E. Delay	2.5V on XIN R.E. to PHI1 R.E.	21	40	21	37	21	32	ns
t_{FCR}	2.3	FCLK Rise Time	0.8V to 2.0V on FCLK R.E.		12		9		7	ns
t_{FCF}	2.3	FCLK Fall Time	2.0V to 0.8V on FCLK F.E.		12		9		7	ns
t_{FCr}	2.2	FCLK to CTTL R.E. Delay	FCLK R.E. to CTTL R.E.	5	17	5	16	5	15	ns
t_{FCf}	2.2	FCLK to CTTL F.E. Delay	FCLK R.E. to CTTL F.E.	5	17	5	16	5	15	ns
t_{FPr}	2.3	FCLK to PHI1 R.E. Delay	FCLK R.E. to PHI1 R.E.	2	17	2	13	2	10	ns
t_{FPf}	2.3	FCLK to PHI1 F.E. Delay	FCLK R.E. to PHI1 F.E.	−4	8	−4	6	−4	4	ns
t_{Fw}	2.3	FCLK High Time with Crystal	At 2.0V on FCLK (Both Edges)	0.25 t_{Cp} −7 ns	0.25 t_{Cp} +7 ns	0.25 t_{Cp} −6 ns	0.25 t_{Cp} +6 ns	0.25 t_{Cp} −5 ns	0.25 t_{Cp} +5 ns	
t_{PCf}	2.3	PHI2 R.E.to CTTL F.E. Delay	PHI2 R.E. to CTTL F.E.	−8	12	−7	11	−6	10	ns
t_{CTw}	2.3	CTTL High Time	At 2.0V on CTTL (Both Edges)	0.5 t_{Cp} −8 ns	0.5 t_{Cp} +8 ns	0.5 t_{Cp} −8 ns	0.5 t_{Cp} +8 ns	0.5 t_{Cp} −7 ns	0.5 t_{Cp} +7 ns	
CTTL TIMING (CL = 50 pF)										
t_{PCr}	2.4	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	−2	7	−2	6	−2	5	ns
t_{CTR}	2.3	CTTL Rise Time	0.8V to 2.0V on CTTL R.E.		6		6		5	ns
t_{CTF}	2.3	CTTL Fall Time	2.0V to 0.8V on CTTL F.E.		5		5		4	ns

Note 1: PHI1 and PHI2 are interchangeable for the following parameters: t_{Cp}, t_{CLh}, t_{CLl}, t_{CLw}, t_{CLR}, t_{CLF}, t_{nOVL}, t_{XPr}, t_{FPr}, t_{FPf}.

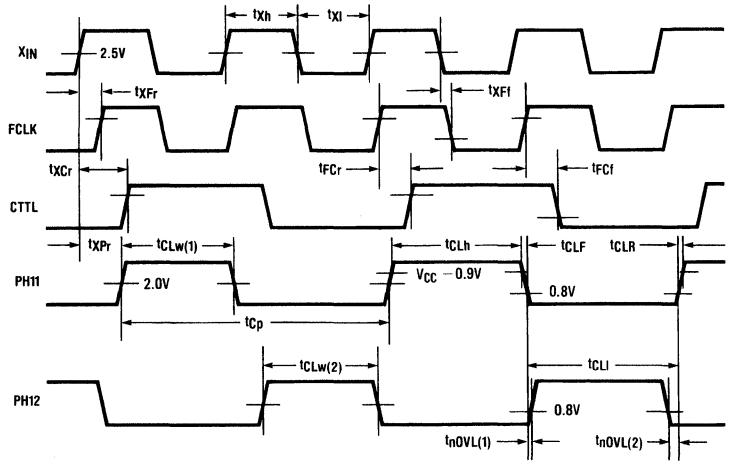
2.0 Device Specifications (Continued)

2.4.3 Timing Tables (Continued)

Name	Figure	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
				Min	Max	Min	Max	Min	Max	
CTTL TIMING (CL = 100 pF)										
t_{PCr}	2.3	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	−2	8	−2	7	−2	6	ns
t_{CTR}	2.3	CTTL Rise Time	0.8V to 2.0V on CTTL R.E.		8		8		7	ns
t_{CTF}	2.2	CTTL Fall Time	2.0V to 0.8V on CTTL F.E.		6		6		5	ns
CONTROL INPUTS ($\overline{RST1}$, $\overline{RSTO}$, $\overline{ADS}$, $\overline{DDIN}$) TIMING										
t_{RSTr}	2.4	$\overline{RSTO}$ R.E. Delay	After PHI1 R.E.		25		20		15	ns
t_{RSTs}	2.4	$\overline{RSTI}$ Setup Time	Before PHI1 R.E.	20		20		20		ns
t_{ADs}	2.4	$\overline{ADS}$ Setup Time	Before PHI1 R.E.	30		28		25		ns
t_{ADw}	2.4	$\overline{ADS}$ Pulse Width	$\overline{ADS}$ L.E. to $\overline{ADS}$ T.E.	25		25		25		ns
t_{DDs}	2.4	$\overline{DDIN}$ Setup Time	Before PHI1 R.E.	10		10		10		ns
t_{DDh}	2.4	$\overline{DDIN}$ Hold Time	After PHI1 R.E.	15		15		15		ns
CONTROL OUTPUTS ($\overline{TSD}$, $\overline{RD}$, $\overline{WR}$, $\overline{DBE}$ & $\overline{RWEN}$/SYNC) TIMING										
t_{Tf}	2.5	$\overline{TSO}$ L.E. Delay	After PHI1 R.E.		12		11		10	ns
t_{Tr}	2.5	$\overline{TSO}$ T.E. Delay	After PHI1 R.E.		20		18		15	ns
$t_{RWf(F)}$	2.5	$\overline{RD}/\overline{WR}$ L.E. Delay (Fast Cycle)	After PHI1 R.E.		50		40		30	ns
$t_{RWf(S)}$	2.6	$\overline{RD}/\overline{WR}$ L.E. Delay (Peripheral Cycle)	After PHI1 R.E.		30		23		15	ns
t_{RWr}	2.5/6	$\overline{RD}/\overline{WR}$ T.E. Delay	After PHI1 R.E.		25		23		20	ns
$t_{DBf(W)}$	2.5/6	$\overline{DBE}$ L.E. Delay (Write Cycle)	After PHI1 R.E.		35		30		24	ns
$t_{DBf(R)}$	2.5/6	$\overline{DBE}$ L.E. Delay (Read Cycle)	After PHI2 R.E.		30		23		15	ns
t_{DBr}	2.5/6	$\overline{DBE}$ T.E. Delay	After PHI2 R.E.		20		20		20	ns
t_{pLZ}	2.7	$\overline{RD}$,$\overline{WR}$ Low Level to TRI-STATE	After $\overline{RWEN}$/SYNC R.E.		20		20		20	ns
t_{pHZ}	2.7	$\overline{RD}$,$\overline{WR}$ High Level to TRI-STATE	After $\overline{RWEN}$/SYNC R.E.		20		20		20	ns
t_{pZL}	2.7	$\overline{RD}$,$\overline{WR}$ TRI-STATE to Low Level	After $\overline{RWEN}$/SYNC F.E.		25		23		20	ns
t_{pZH}	2.7	$\overline{RD}$,$\overline{WR}$ TRI-STATE to High Level	After $\overline{RWEN}$/SYNC F.E.		25		23		20	ns
WAIT STATES & CYCLE HOLD ($\overline{CWAIT}$, $\overline{WAITn}$, $\overline{PER}$ & $\overline{RDY}$) TIMING										
$t_{CWs(H)}$	2.8	$\overline{CWAIT}$ Setup Time (Cycle Hold)	Before PHI1 R.E.	35		30		25		ns
$t_{CWh(H)}$	2.8	$\overline{CWAIT}$ Hold Time (Cycle Hold)	After PHI1 R.E.	0		0		0		ns
$t_{CWs(W)}$	2.8/9	$\overline{CWAIT}$ Setup Time (Wait States)	Before PHI2 R.E.	13		12		10		ns
$t_{CWh(W)}$	2.9	$\overline{CWAIT}$ Hold Time (Wait States)	After PHI2 R.E.	20		14		8		ns
t_{Ws}	2.9	$\overline{WAIT}$n Setup Time	Before PHI2 R.E.	5		5		5		ns
t_{Wh}	2.9	$\overline{WAIT}$n Hold Time	After PHI2 R.E.	25		20		15		ns
t_{Ps}	2.10	$\overline{PER}$ Setup Time	Before PHI1 R.E.	0		0		0		ns
t_{Ph}	2.10	$\overline{PER}$ Hold Time	After PHI1 R.E.	30		25		20		ns
t_{Rd}	2.8/9/10	$\overline{RDY}$ Delay	After PHI2 R.E.		30		26		23	ns
SYNCHRONIZATION (SYNC) TIMING										
t_{Sys}	2.11	SYNC Setup Time	Before FCLK R.E.	20		19		18		ns
t_{Syh}	2.11	SYNC Hold Time	After FCLK R.E.	3		2		0		ns
t_{CS}	2.11	CTTL/SYNC Inversion Delay	CTTL (master) to $\overline{RWEN}$/SYNC (slave)		25		20		15	ns

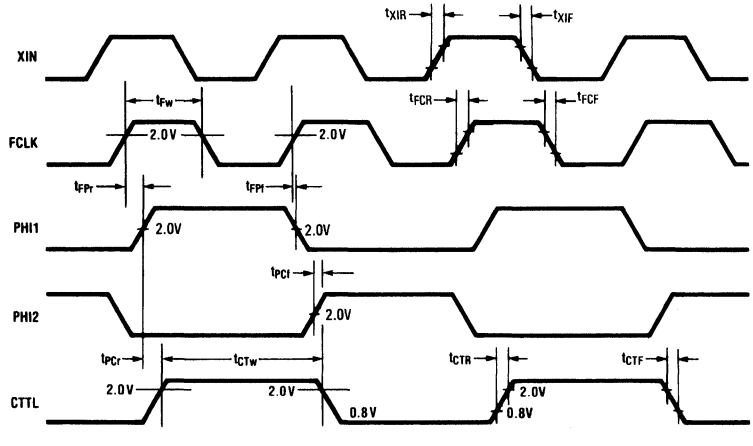
2.0 Device Specifications (Continued)

2.4.4 Timing Diagrams



TL/EE/5590–24

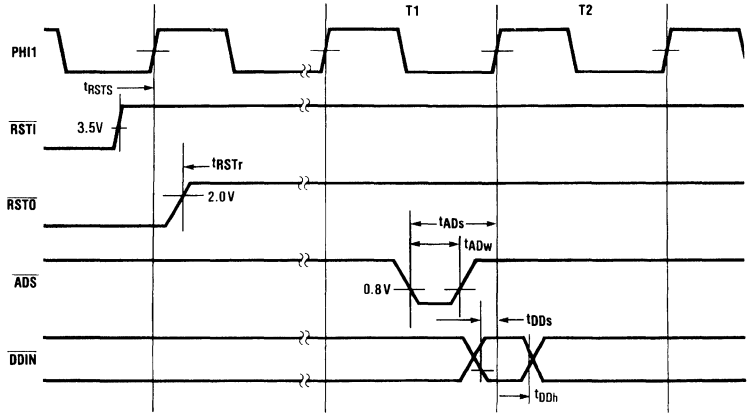
FIGURE 2-2. Clock Signals (a)



TL/EE/5590–25

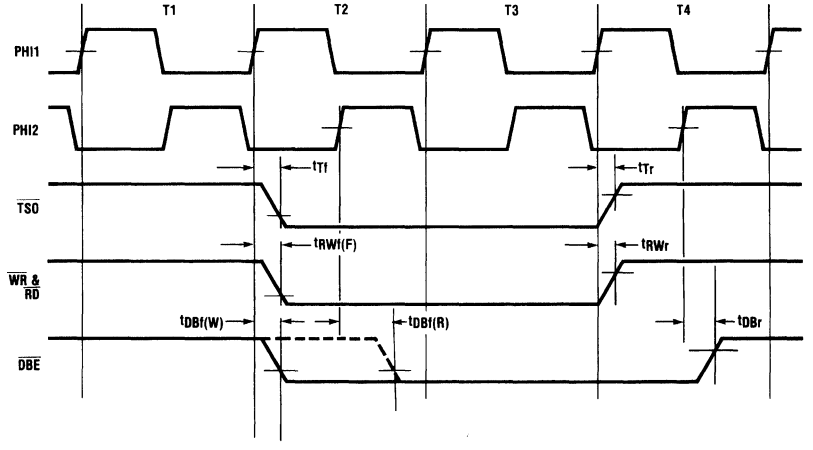
FIGURE 2-3. Clock Signals (b)

2.0 Device Specifications (Continued)



TL/EE/5590–26

FIGURE 2-4. Control Inputs

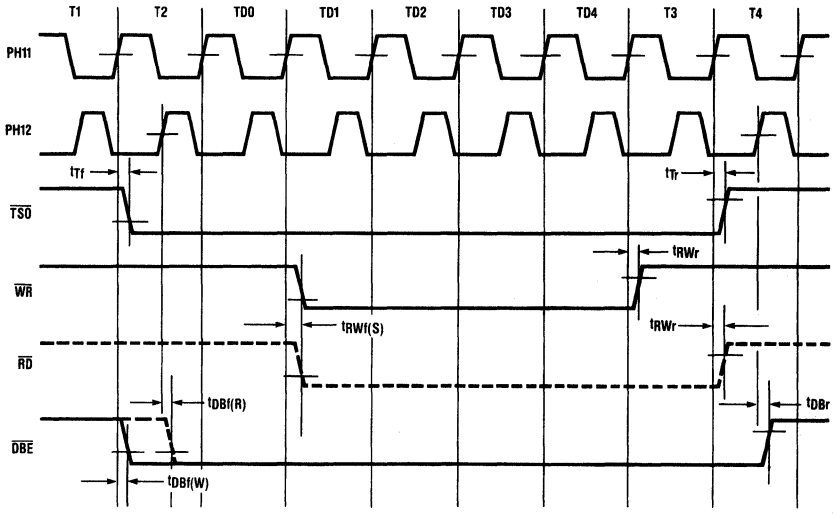


TL/EE/5590–27

FIGURE 2-5. Control Outputs (Fast Cycle)

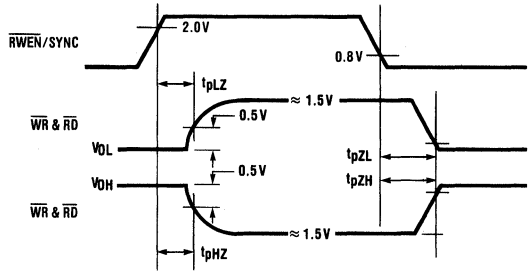
4

2.0 Device Specifications (Continued)



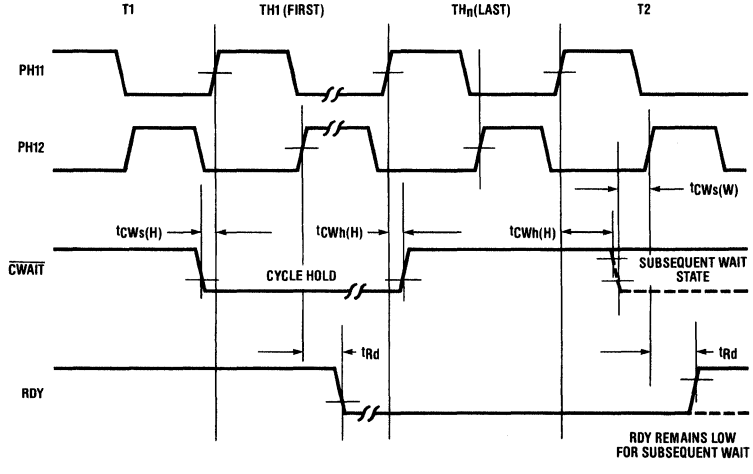
TL/EE/5590-28

FIGURE 2-6. Control Outputs (Peripheral Cycle)



TL/EE/5590-29

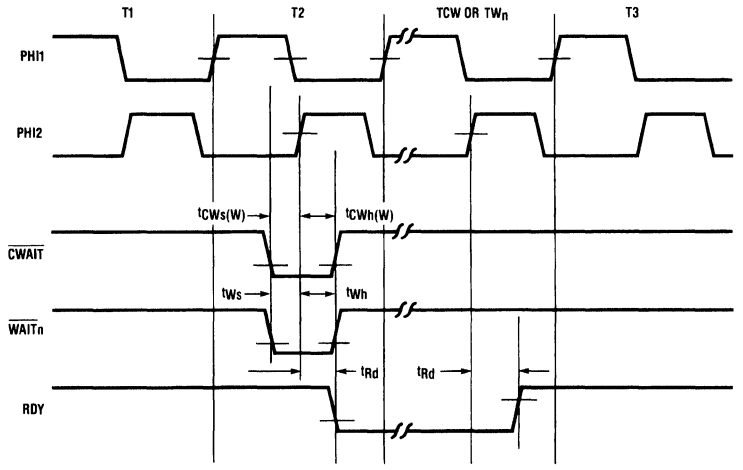
FIGURE 2-7. Control Outputs (TRI-STATE Timing)



TL/EE/5590-30

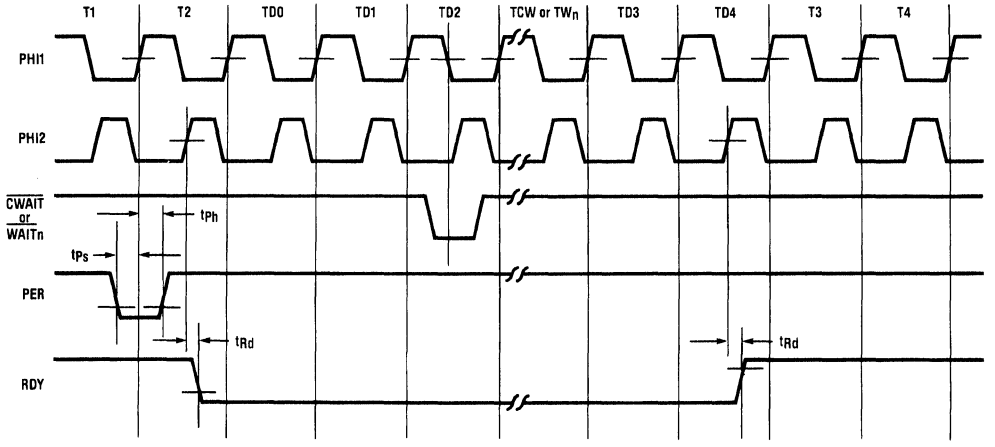
FIGURE 2-8. Cycle Hold

2.0 Device Specifications (Continued)



TL/EE/5590-31

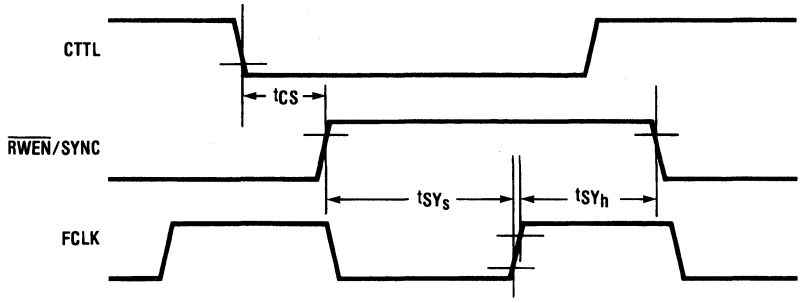
FIGURE 2-9. Wait States (Fast Cycle)



TL/EE/5590-32

FIGURE 2-10. Wait States (Peripheral Cycle)

2.0 Device Specifications (Continued)



TL/EE/5590–33

FIGURE 2-11. Synchronization Timing

National Semiconductor Corporation

ADVANCED INFORMATION

NS32301-10/NS32301-15 Timing Control Unit

1.0 General Description

The NS32301 Timing Control Unit (TCU) is a 28 pin device fabricated using an Advanced Low Power Schottky bipolar process. It provides a two phase clock, system control logic and cycle extension logic for the Series 32000® microprocessor family. The TCU input clock can be provided by either a crystal or an external clock signal whose frequency is twice the system clock frequency.

In addition to the two phase clock for the CPU and MMU (PHI1 and PHI2), the NS32301 TCU also provides a system clock for general system use (CTTL). CTTL is a TTL replica of the PHI1 clock.

The system control logic and cycle extension logic make the TCU very attractive by providing extremely accurate bus control signals, and allowing extensive control over the bus cycle timing.

The NS32301 provides all the essential functions found in the NS32201. It supports protocols in both 320XX and 323XX series microprocessors. Nominal operating frequency is increased to 15 MHz. Due to high switching currents and additional functions, the package and pin-out of the NS32301 are different from that of the NS32201.

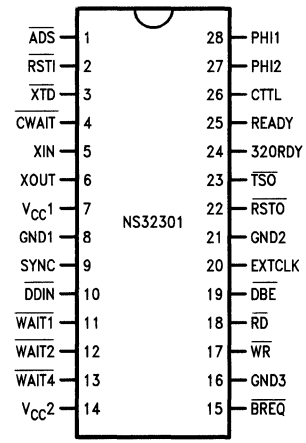
In order to accommodate new functions required for the support of the NS32332 CPU and mixed 320XX and 323XX protocols, the peripheral and fast clock functions are dropped from the NS32201 TCU. New features include burst access support and a dedicated synchronization pin.

2.0 Features

- Oscillator at twice the CPU clock frequency
- 2 phase full V_{CC} swing high capacitance clock drivers (PHI1 and PHI2)
- Provides all the essential functions in the NS32201-10 TCU
- Nominal operating frequency of 15 MHz with 10% guard band
- Supports protocols in both 320XX and 323XX series microprocessors
- NS32332 burst access support
- Improved noise level at all outputs
- $\overline{TSO}$ timing tightened to ease memory access
- Dedicated synchronization input
- 3-bit input ($\overline{WAITn}$) allowing precise specification of 0 to 7 wait states
- Cycle HOLD for system arbitration and/or memory refresh
- System timing (CTTL) and control ($\overline{RD}$, $\overline{WR}$, and $\overline{DBE}$) outputs
- General purpose Timing State Output ($\overline{TSO}$) that identifies internal states
- Provides "ready" outputs for the Series 32000 CPUs
- Synchronous system reset generation from Schmitt trigger input
- Single 5V power supply
- 28-pin dual-in-line package

Connection Diagram

Dual-In-Line Package



TL/EE/8777-1

Top View
(Pin-Out Subject to Change)

3.0 Pin Descriptions

3.1 SUPPLIES

VCC1. +5V

VCC2. +5V

GND 1. Ground

GND 2. Ground

GND 3. Ground

3.2 INPUT SIGNALS

Reset Input ($\overline{RSTI}$) Active low: Schmitt triggered, asynchronous signal used to generate the system reset.

Address Strobe ($\overline{ADS}$): Active low. Identifies the first cycle (T2/TMMU) of the three-cycle access timing.

Synchronization Input (SYNC): This input is used to accept an external clock synchronization signal from the master TCU during reset. Internal TCU logic samples this input and aligns the phase of its local clocks to that of the master TCU. During reset ($\overline{RSTO}$ active), the TCU samples this signal at the rising edge of EXTCLK when CTTL is high. If the sampled value is low, the phase of CTTL, PHI1, and PHI2 of the slave TCU will be shifted by one EXTCLK clock cycle. If only one TCU is used in a system, the SYNC input should be tied high through a resistor.

Burst Request Input ($\overline{BREQ}$): Active low. Indicates the activation of a burst cycle. This signal is sampled at the falling edge of PHI1 in T4 of each access. If $\overline{BREQ}$ is sampled low, the TCU will enter the T3 – T4 burst loop. Once inside the burst loop, $\overline{BREQ}$ will be sampled at the falling edge of each PHI1. If $\overline{BREQ}$ is is sampled high, the TCU will terminate the burst loop.

Data Direction Input ($\overline{DDIN}$): Indicates the direction of data transfer. Implies a read when low and a write when high.

Crystal or External Clock Input (XIN): Input from a crystal.

Continuous Wait Input ($\overline{CWAIT}$): Active low. Indicates a continuous wait if sampled low at the beginning of T3. This signal is also sampled in mid T2/TMMU and a CYCLE HOLD is initiated if $\overline{CWAIT}$ is found active.

Note: Activation of the $\overline{CWAIT}$ line in any cycle other than T2/TMMU and T3 (including wait states) will be ignored by the TCU.

Three-Bit Wait State Inputs ($\overline{WAIT1}$, $\overline{WAIT2}$, $\overline{WAIT4}$): Active low. These inputs allow from zero to seven wait states to be inserted into an access. They are binary weighted and sampled at the rising edge of PHI1 in the first T3 and resampled at the rising edge of PHI1 in each T3 where $\overline{CWAIT}$ is low.

Extend Input ($\overline{XTD}$): Active low. This signal is sampled at the rising edge of PHI1 in the first T3 of an access. A low on this line will extend the duration of $\overline{RD}$, $\overline{WR}$, $\overline{TSO}$, and $\overline{DBE}$ by one clock cycle.

External Clock Input (EXTCLK): When an external clock source is used to drive the TCU, it should be connected to the EXTCLK input. In applications where a crystal is used, this input should be tied high through a resistor.

3.3 OUTPUT SIGNALS

Reset Output ($\overline{RSTO}$): Active low. This signal becomes active when $\overline{RSTI}$ is low, initiating a reset. $\overline{RSTO}$ goes high on the first rising edge of PHI1 after $\overline{RST1}$ goes high.

Read Strobe ($\overline{RD}$): Active low. Identifies a read cycle. It is decoded from $\overline{DDIN}$. This signal will be extended by one clock cycle if $\overline{XTD}$ is active. During a burst access, $\overline{RD}$ will be activated between each T3 – T4 burst pair (see section on burst support).

Write Strobe ($\overline{WR}$): Active low. Identifies a write cycle. It is decoded from $\overline{DDIN}$. This signal will be extended by one clock cycle if $\overline{XTD}$ is active.

Data Buffer Enable ($\overline{DBE}$): Active low. This signal is used to control the data bus buffers. It is low when the data buffers are to be enabled. $\overline{DBE}$ will be extended by one clock cycle if $\overline{XTD}$ is active. $\overline{DBE}$ is activated at the beginning of T3 in each access. Deactivation of $\overline{DBE}$ is controlled by the $\overline{BREQ}$ signal. $\overline{BREQ}$ is sampled at the falling edge of PHI1 in T4. If $\overline{BREQ}$ is sampled inactive, $\overline{DBE}$ will return high after the rising edge of PHI2 in T4. If $\overline{BREQ}$ is sampled active, $\overline{DBE}$ will remain low until $\overline{BREQ}$ is deactivated (i.e., $\overline{DBE}$ will stay low throughout a burst access).

Timing State Output ($\overline{TSO}$): Active low. The falling edge of this signal marks the beginning of T3 while its rising edge marks the beginning of T4. This signal will be extended by one clock cycle if $\overline{XTD}$ is active. $\overline{TSO}$ is also active between T3 and T4 in burst accesses.

Ready (RDY): Active high. This signal stays low as long as wait states are inserted in the access. RDY goes high only during the last T3 of an access.

320 Ready (320RDY): Active high. This ready signal is used by the NS32032 CPU and the NS32082 MMU. If no wait states are inserted in an access, 320RDY stays high. If wait states are introduced by $\overline{WAITn}$ or $\overline{CWAIT}$, 320RDY will be pulled to a low level beginning at the middle of the first T3, or in the case of a cycle hold, in the middle of the first cycle hold time state. This signal will return to a high level at the middle of the last T3 state.

PHI1 and PHI2: These two clocks provide the CPU with a two-phase, non-overlapping clock signal. Their frequency is half that of the crystal or external clock source.

TTL System Clock (CTTL): This is a 50% duty cycle TTL clock running at system frequency. The rising edge of CTTL is closely synchronized to the leading edge of the PHI1 clock.

Crystal Output (XOUT): This line is used as the return path of the crystal.

4.0 Functional Description

The NS32301 TCU is a three (3) cycle state machine. The basic states are: T2/TMMU – T3 – T4. Wait states are inserted as additional T3s. The last T3 is the time state where RDY goes high. The T2/TMMU state is entered whenever the $\overline{ADS}$ input is pulsed low.

$\overline{CWAIT}$ is sampled at the falling edge of CTTL in the T2/TMMU state. The TCU enters the CYCLE HOLD mode and stays in this mode for as long as the sampled $\overline{CWAIT}$ value remains low. Cycle Hold states (TH) are inserted between T2/TMMU and the first T3.

If no cycle hold is detected, $\overline{CWAIT}$ is sampled at the rising edge of PHI1 in each T3 of a regular access. As long as $\overline{CWAIT}$ is sampled active, the TCU will continue to insert wait states by repeating T3.

The $\overline{XTD}$ line is sampled at the rising edge of PHI1 in the first T3 of an access. A low on this line will extend the duration of $\overline{TSO}$, $\overline{RD}$, $\overline{WR}$, and $\overline{DBE}$ signals by one clock cycle. When the Extend function is active, a Tx state will be inserted between the last T3, and T4.

4.0 Functional Description (Continued)

The $\overline{BREQ}$ line is sampled at the falling edge of PHI1 in each T4. If this line is sampled low, a burst cycle is detected. The TCU will complete T4 normally and then loop between states T3 and T4. $\overline{TSO}$ will be activated at the beginning of T3 and released at the beginning of T4 of each T3 – T4 pair. While looping between T3 and T4 states, the $\overline{BREQ}$ line will be sampled at the falling edge of PHI1 in each T3 and T4. If the sampled $\overline{BREQ}$ signal is high, the NS32301 will terminate its burst loop.

The first T3 is in the time state where all access related signals are activated:

- $\overline{TSO}$
- $\overline{RD}$ or $\overline{WR}$
- $\overline{DBE}$
- READY
- 320RDY

READY is a normally low signal. It goes high only during the T3 state. In accesses where wait states are inserted, READY will go high only during the last T3.

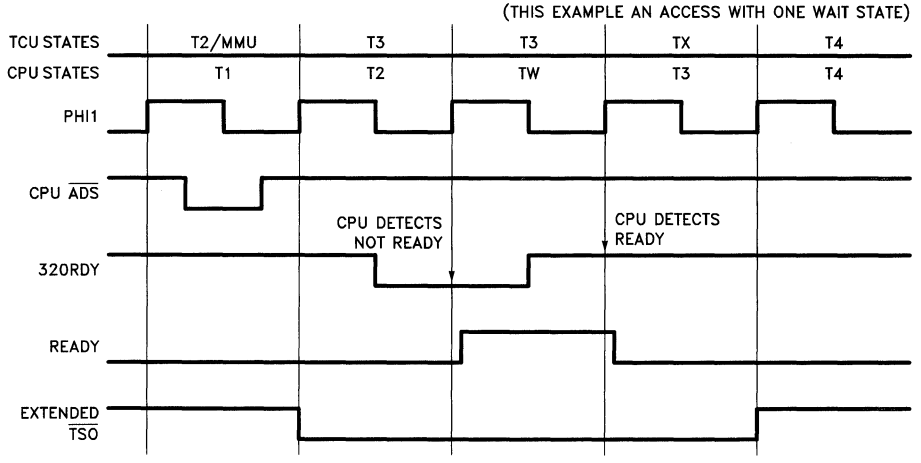
320RDY is the ready signal used by the NS32032 CPU and the NS32082 MMU. If the TCU enters cycle hold, 320RDY will be asserted (low) in the middle of the first cycle hold time state (TH). If no wait states are inserted in the access after cycle hold is released, 320RDY will return high in the middle of T3. If one or more wait states are inserted, 320RDY will remain low until the middle of the last T3 state. In accesses without cycle hold, if no wait states are inserted in the access, 320RDY will remain high. If one or more wait states are inserted, 320RDY will drop to a low level at the middle of the first T3 and return high at the middle of the last T3.

$\overline{DDIN}$ is sampled at the rising edge of PHI1 in the first T3. Its sampled value controls the subsequent generation of either a read or a write signal.

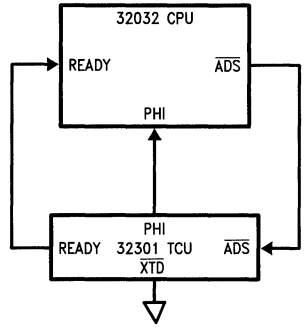
$\overline{WAITn}$ are 3 negative true wait signals. The binary weighted value of the $\overline{WAITn}$ lines determines the number of programmed wait states to be inserted into the access cycle. These lines are sampled at the rising edge of PHI1 in the first T3 of an access, and are sampled at the rising edge of PHI1 in each T3 where the $\overline{CWAIT}$ signal is low.

4.1 BURST SUPPORT

The NS32332 CPU is capable of reading in bursts of up to 16 bytes each. The basic burst access states are: T1 – T2 – T3 – T4 – T3 – T4 – T3 – T4 – T3 – T4. In order to accommodate the burst feature, the TCU must sample the Burst Request signal ($\overline{BREQ}$) in mid T4 of each access. If $\overline{BREQ}$ is detected low, the NS32301 will complete T4 normally and enter into a T3 – T4 burst loop. Once in the burst



TL/EE/8777-2



TL/EE/8777-3

FIGURE 1. NS32032 CPU

4.0 Functional Description (Continued)

loop, the TCU will sample the $\overline{BREQ}$ line at the falling edge of PHI1 in each T3 and T4. If the sampled $\overline{BREQ}$ is high, the TCU will terminate the burst loop, (i.e., T2/TMMU – T3 – T4 ($\overline{BREQ}$ low) – T3 – T4 ($\overline{BREQ}$ low) – T3 – T4 ($\overline{BREQ}$ high).

The $\overline{CWAIT}$ line will be sampled at the rising edge of PHI1 in each T3 of a burst loop. The $\overline{WAIT}$n lines are sampled at the rising edge of PHI1 in the first T3 of each burst nibble and resampled at the rising edge of PHI1 in each T3 of that burst nibble where $\overline{CWAIT}$ is low. The appropriate number of wait states will be inserted through the READY and 320RDY signals. $\overline{TSO}$ and $\overline{RD}$ (burst access occurs only during read operations) will be activated at the beginning of the first T3 state and deactivated at the beginning of the T4 state in each T3 – T4 pair. The $\overline{DBE}$ (data buffer enable) signal will remain active (low) throughout the burst access: it will go low at the beginning of the first T3 and return high after the middle of the last T4.

The relative burst cycle timing is shown in *Figure 15* and *16*.

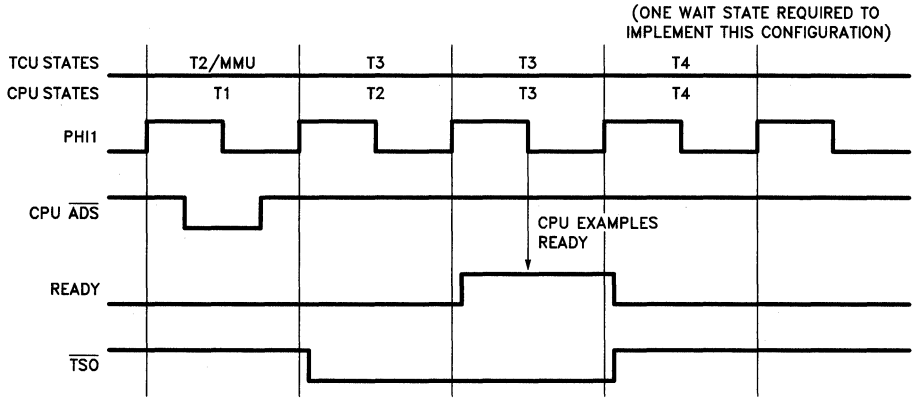
4.2 MIXED PROTOCOL

The NS32301 TCU supports both NS32032 and NS32332 type protocols. The NS32301 TCU also handles situations where a mixture of the two protocols are used together. One such case occurs when the NS32332 CPU uses an NS32082 MMU to perform address translation.

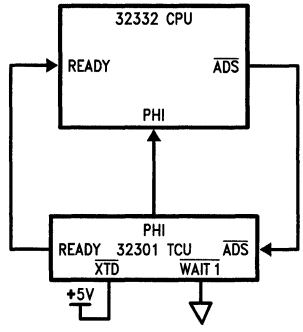
The possible combinations are:

- NS32332 CPU running by itself
- NS32032 CPU running by itself
- NS32032 CPU running with the NS32082 MMU
- NS32332 CPU running with the NS32082 MMU
- NS32332 CPU running with the NS32382 MMU

The timing and block diagrams in *Figures 1* through *8* demonstrate the use of the NS32301 TCU in each of the five combinations.



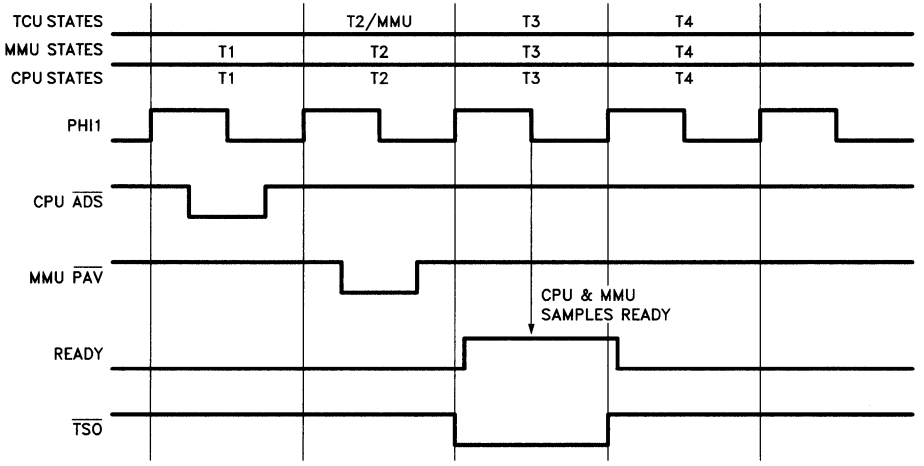
TL/EE/8777–4



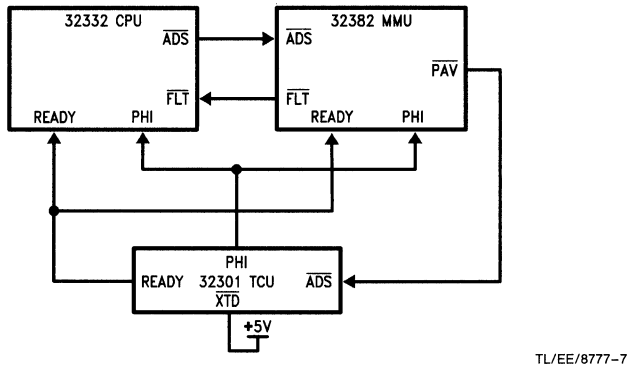
TL/EE/8777–5

FIGURE 2. NS32332 CPU

4.0 Functional Description (Continued)



TL/EE/8777-6

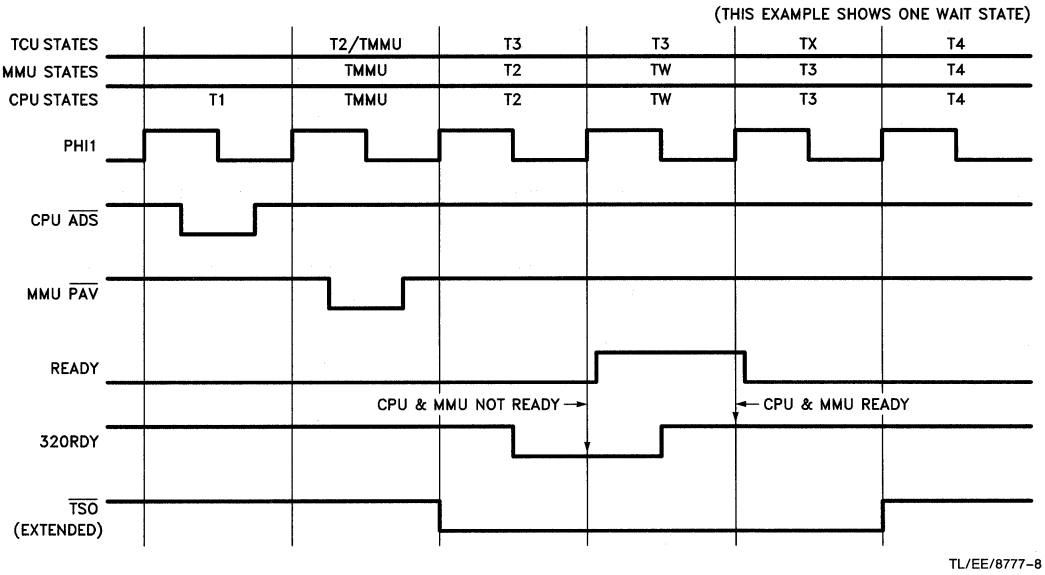


TL/EE/8777-7

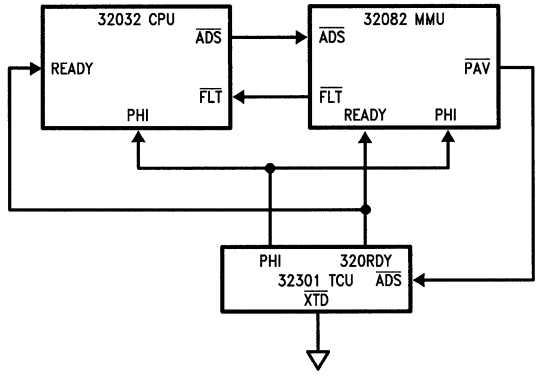
FIGURE 3. NS32332 CPU and NS32382 MMU

4

4.0 Functional Description (Continued)



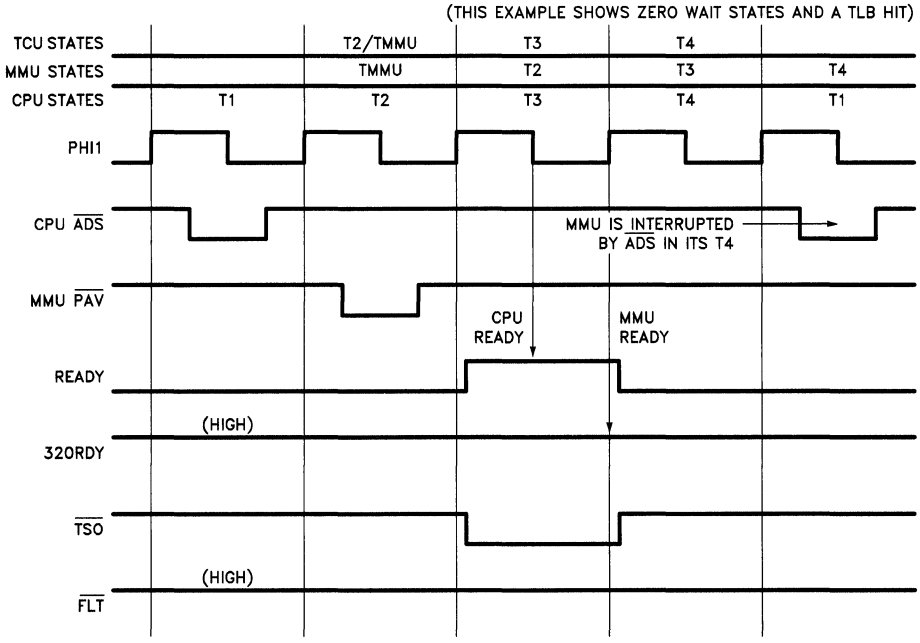
TL/EE/8777–8



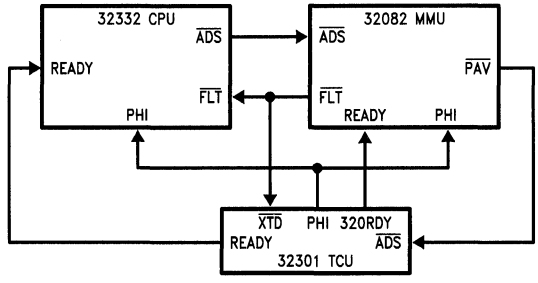
TL/EE/8777–9

FIGURE 4. NS32032 CPU and NS32082 MMU

4.0 Functional Description (Continued)



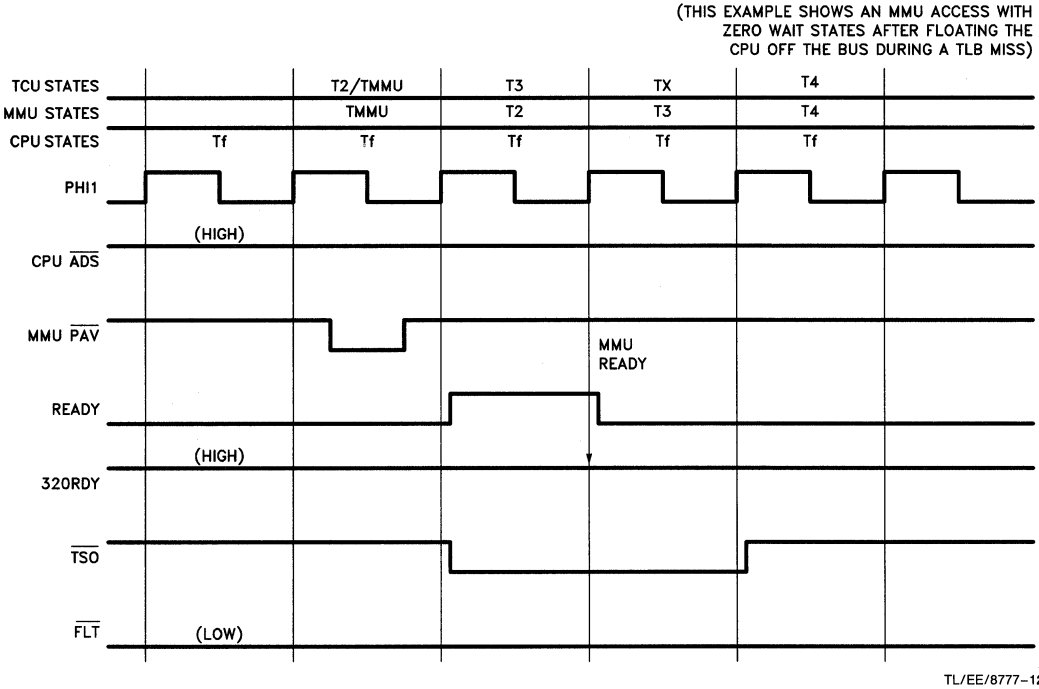
TL/EE/8777-10



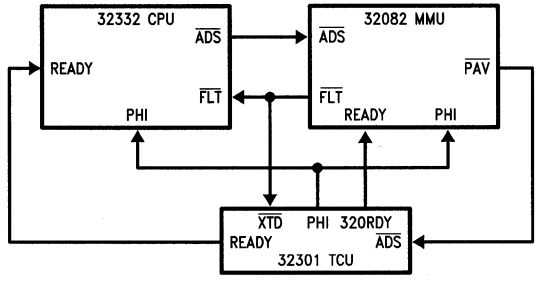
TL/EE/8777-11

FIGURE 5. NS32332 CPU and NS32082 MMU

4.0 Functional Description (Continued)



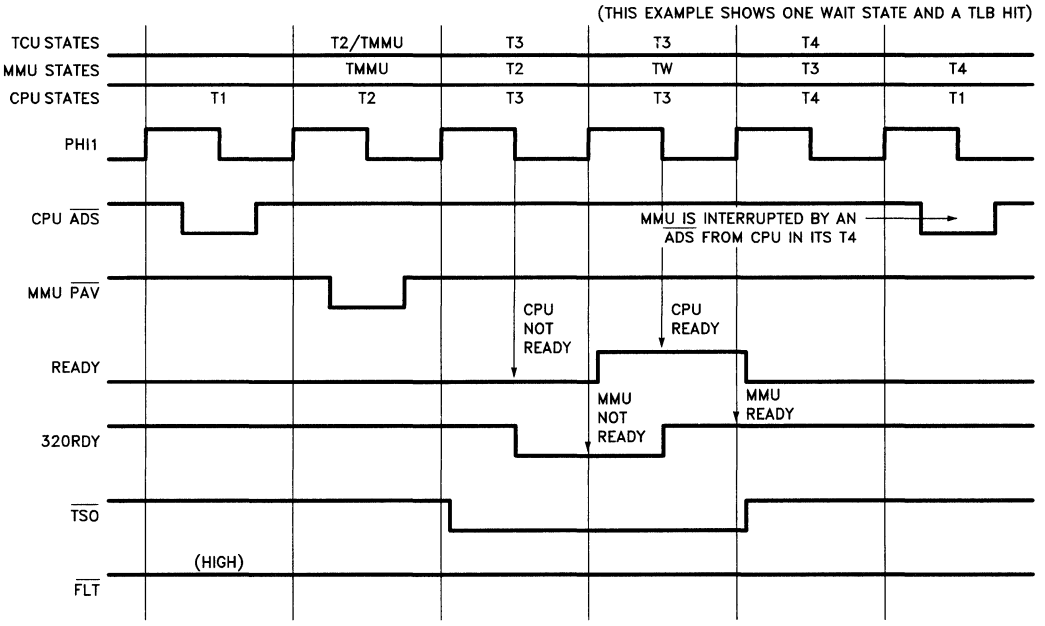
TL/EE/8777–12



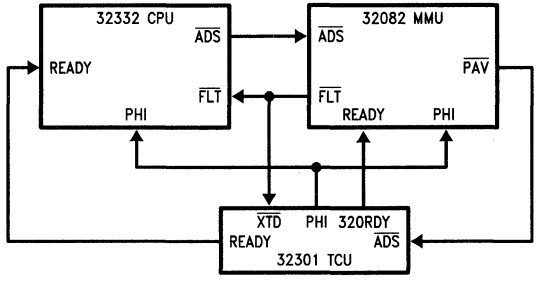
TL/EE/8777–13

FIGURE 6. NS32332 CPU and NS32082 MMU

4.0 Functional Description (Continued)



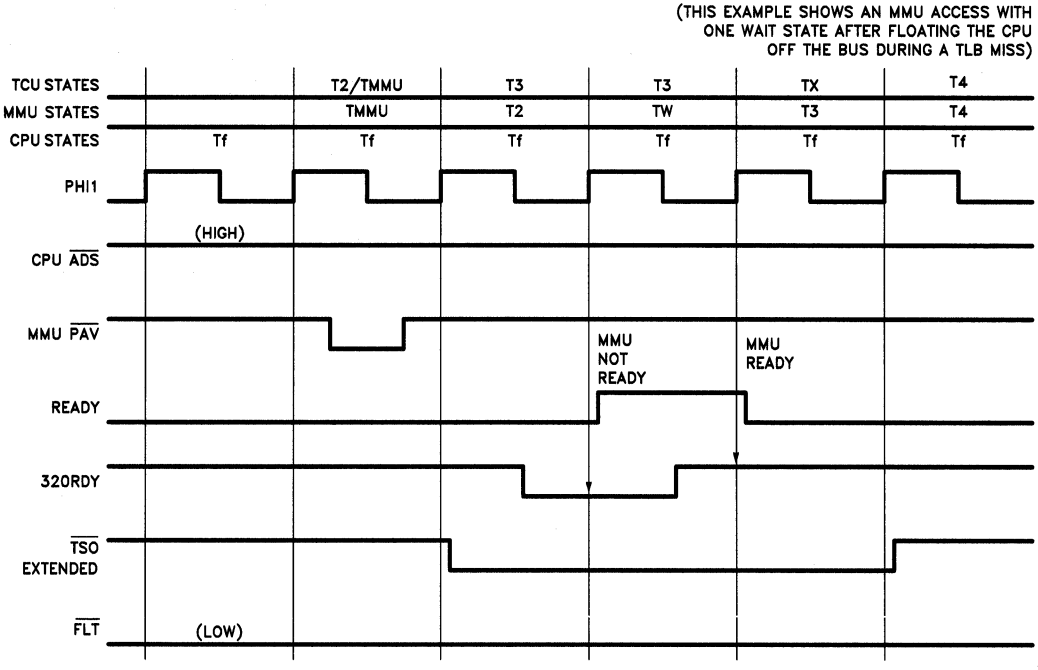
TL/EE/8777–14



TL/EE/8777–15

FIGURE 7. NS32332 CPU and NS32082 MMU

4.0 Functional Description (Continued)



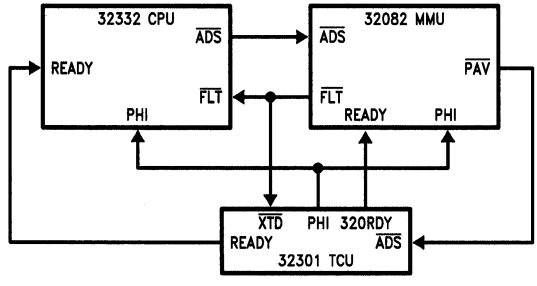


FIGURE 8. NS32332 CPU and NS32082 MMU

4.0 Functional Description (Continued)

4.3 CONTROL INPUT/OUTPUT TIMING

The relative timing for the control input signals and the respective control output signals is shown in *Figures 9* and *10*.

4.4 CLOCK SYNCHRONIZATION IN CYCLE EXTENSION

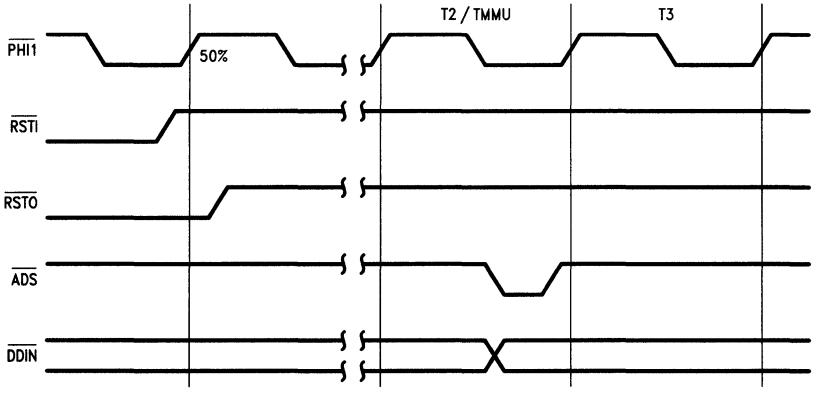
The relative wait state timing is shown in *Figure 11*. Cycle holds are shown in *Figure 12* clock synchronization timing is shown in *Figure 13*. Cycle extensions are shown in *Figure 14*.

4.5 BURST TIMING

Burst related timings are shown in *Figures 15* and *16*.

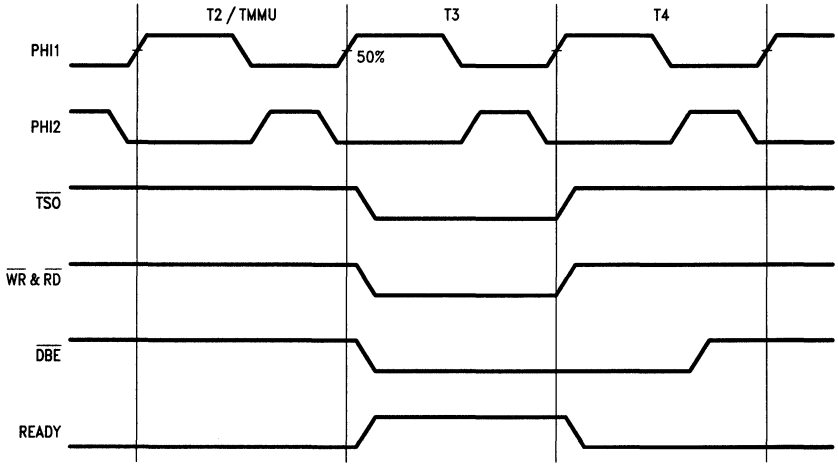
4.6 SYSTEM CLOCKS

System clock relations are shown in *Figure 17*.



TL/EE/8777-18

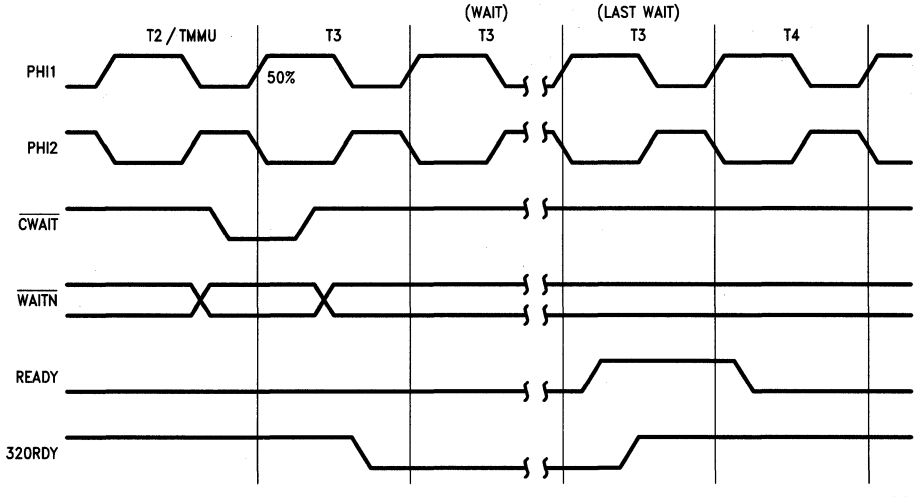
FIGURE 9. Control Inputs



TL/EE/8777-19

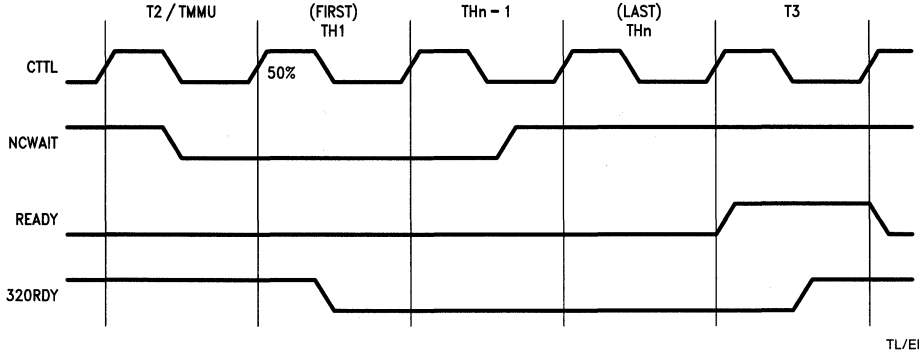
FIGURE 10. Control Outputs (Fast Cycle)

4.0 Functional Description (Continued)



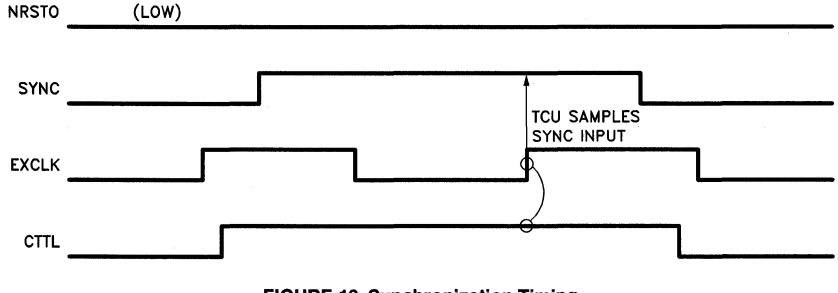
TL/EE/8777–20

FIGURE 11. Wait States with NCWAIT and $WAIT_n$ (Fast Cycle)



TL/EE/8777–21

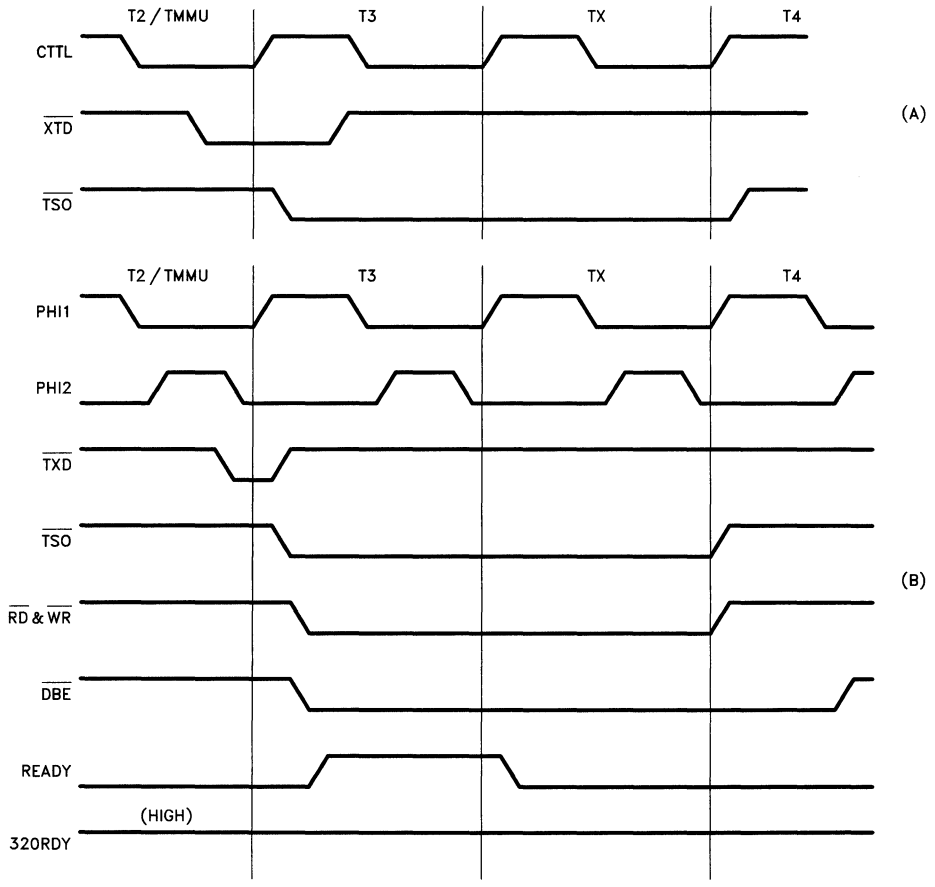
FIGURE 12. Cycle Hold (No Wait States)



TL/EE/8777–22

FIGURE 13. Synchronization Timing

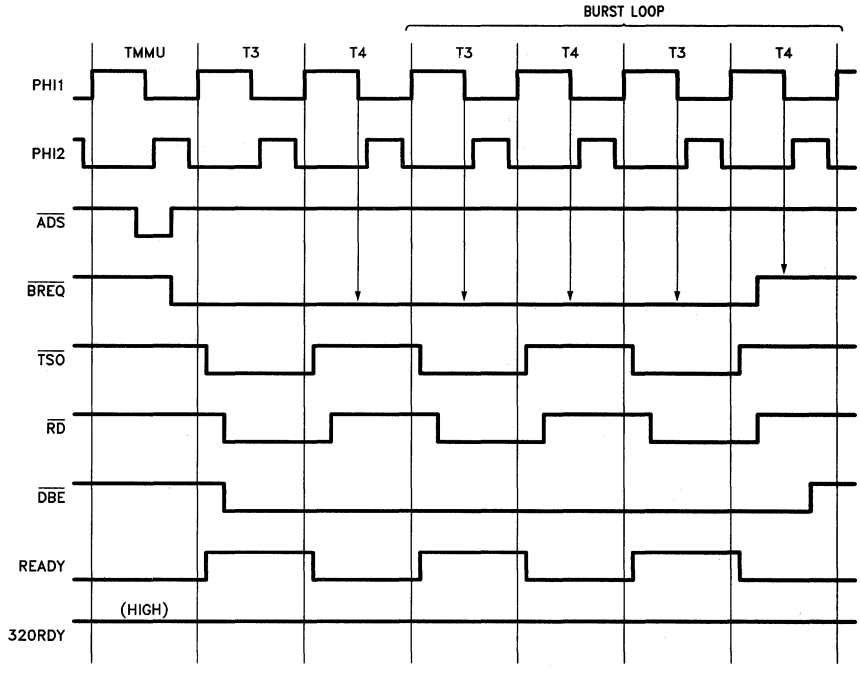
4.0 Functional Description (Continued)



TL/EE/8777–23

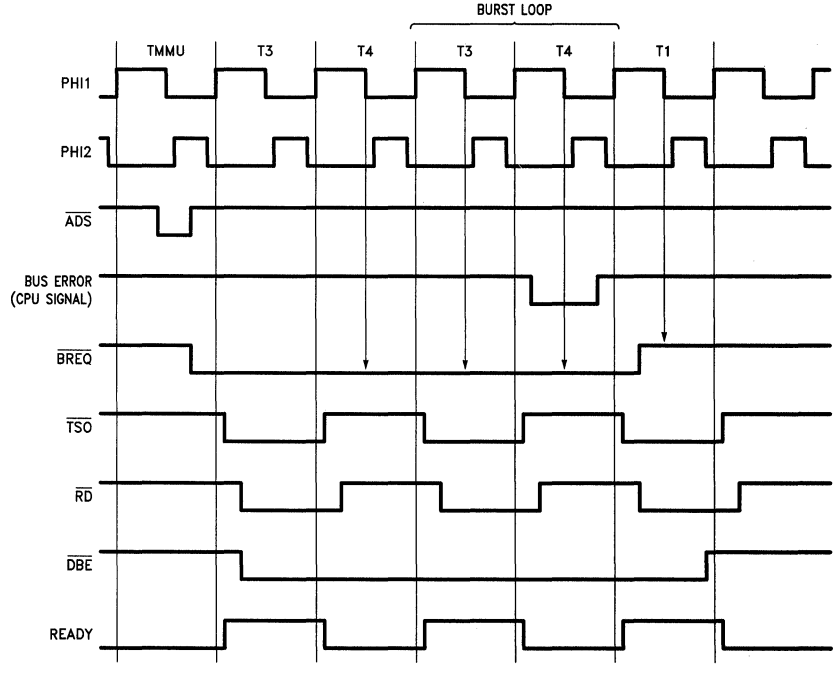
FIGURE 14. (A)(B) Extended Cycle Timing

4.0 Functional Description (Continued)



TL/EE/8777–24

FIGURE 15. Burst Support Timing (No Wait State)



TL/EE/8777–25

FIGURE 16. BREQ Release Before End of Burst Due to Bus Error (No Wait State)

4.0 Functional Description (Continued)

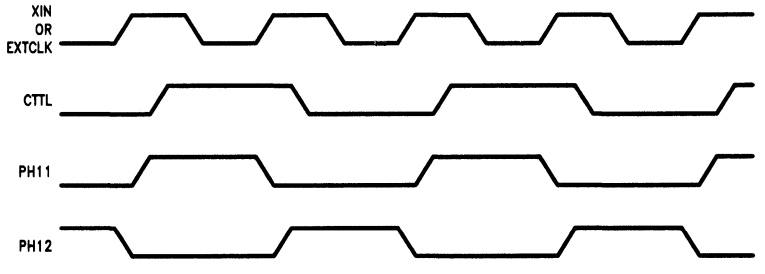


FIGURE 17. Clock Signals

National Semiconductor Corporation

NS32202-6/NS32202-8/NS32202-10 Interrupt Control Units

General Description

The NS32202 Interrupt Control Unit (ICU) is the interrupt controller for the Series 32000® microprocessor family. It is a support circuit that minimizes the software and real-time overhead required to handle multi-level, prioritized interrupts. A single NS32202 manages up to 16 interrupt sources, resolves interrupt priorities, and supplies a single-byte interrupt vector to the CPU.

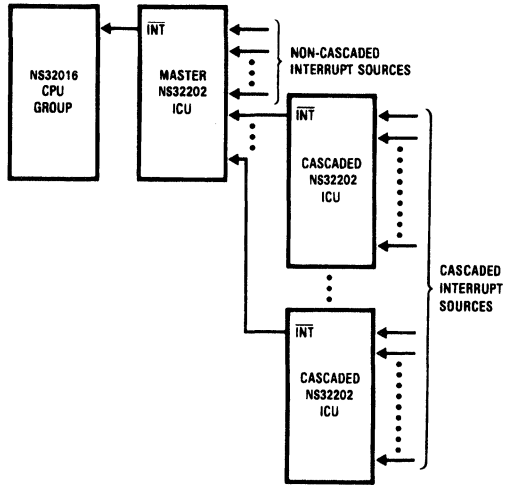
The NS32202 can operate in either of two data bus modes: 16-bit or 8-bit. In the 16-bit mode, eight hardware and eight software interrupt positions are available. In the 8-bit mode, 16 hardware interrupt positions are available, 8 of which can be used as software interrupts. In this mode, up to 16 additional ICUs may be cascaded to handle a maximum of 256 interrupts.

Two 16-bit counters, which may be concatenated under program control into a single 32-bit counter, are also available for real-time applications.

Features

- 16 maskable interrupt sources, cascadable to 256
- Programmable 8- or 16-bit data bus mode
- Edge or level triggering for each hardware interrupt with individually selectable polarities
- 8 software interrupts
- Fixed or rotating priority modes
- Two 16-bit, DC to 10 MHz counters, that may be concatenated into a single 32-bit counter
- Optional 8-bit I/O port available in 8-bit data bus mode
- High-speed XMOS™ technology
- Single, +5V supply
- 40-pin, dual in-line package

Basic System Configuration



TL/EE/5117–1

Table of Contents

1.0 PRODUCT INTRODUCTION

1.1 I/O Buffers
1.2 Read/Write Logic and Decoders
1.3 Timing and Control
1.4 Priority Control
1.5 Counters

2.0 FUNCTIONAL DESCRIPTION

2.1 Reset
2.2 Initialization
2.3 Vectored Interrupt Handling
2.3.1 Non-Cascaded Operation
2.3.2 Cascade Operation
2.4 Internal ICU Operating Sequence
2.5 Interrupt Priority Modes
2.5.1 Fixed Priority Mode
2.5.2 Auto-Rotate Mode
2.5.3 Special Mask Mode
2.5.4 Polling Mode

3.0 ARCHITECTURAL DESCRIPTION

3.1 HVCT - Hardware Vector Register (R0)
3.2 SVCT - Software Vector Register (R1)
3.3 ELTG - Edge/Level Triggering Registers (R2, R3)
3.4 TPL - Triggering Polarity Registers (R4, R5)
3.5 IPND - Interrupt Pending Registers (R6, R7)
3.6 ISRV - Interrupt In-Service Registers (R8, R9)
3.7 IMSK - Interrupt Mask Registers (R10, R11)
3.8 CSRC - Cascaded Source Registers (R12, R13)

3.0 ARCHITECTURAL DESCRIPTION (Continued)

3.9 FPRT - First Priority Registers (R14, R15)
3.10 MCTL - Mode Control Register (R16)
3.11 OSCASN - Output Clock Assignment (R17)
3.12 CIPTR - Counter Interrupt Pointer Register (R18)
3.13 PDAT - Port Dada Register (R19)
3.14 IPS - Interrupt/Port Select Register (R20)
3.15 PDIR - Port Direction Register (R21)
3.16 CCTL - Counter Control Register (R22)
3.17 CICTL - Counter Interrupt Control Register (R23)
3.18 LCSV/HCSV - L-Counter Starting Value/H-Counter Starting Value Registers (R24, R25, R26, and R27)
3.19 LCCV/HCCV - L-Counter Current Value/H-Counter Current Value Registers (R28, R29, R30, and R31)
3.20 Register Initialization

4.0 DEVICE SPECIFICATIONS

4.1 NS32202 Pin Descriptions
4.1.1 Power Supply
4.1.2 Input Signals
4.1.3 Output Signals
4.1.4 Input/Output Signals
4.2 Absolute Maximum Ratings
4.3 Electrical Characteristics
4.4 Switching Characteristics
4.4.1 Definitions
4.4.1.1 Timing Tables
4.4.1.2 Timing Diagrams

List of Illustrations

NS32202 ICU Block Diagram 1-1
Counter Output Signals in Pulsed Form and Square Waveform for Three Different Initial Values 1-2
Counter Configuration and Basic Operations 1-3
Interrupt Control Unit Connections in 16-Bit Bus Mode 2-1
Interrupt Control Unit Connections in 8-Bit Bus Mode 2-2
Cascaded Interrupt Control Unit Connections in 8-Bit Bus Mode 2-3
CPU Interrupt Acknowledge Sequence 2-4
Interrupt Dispatch and Cascade Tables 2-5
CPU Return from Interrupt Sequence 2-6
ICU Interrupt Acknowledge Sequence 2-7
ICU Return from Interrupt Sequence 2-8
ICU Internal Registers 3-1
HVCT Register Data Coding 3-2
Recommended ICU's Initialization Sequence 3-3
NS32202 ICU Connection Diagram 4-1
Timing Specification Standard 4-2
READ/INTA Cycle 4-3
Write Cycle 4-4
Interrupt Timing in Edge Triggering Mode 4-5
Interrupt Timing in Level Triggering Mode 4-6
External Interrupt-Sampling-Clock to be Provided at Pin COUT/SCIN When in Test Mode 4-7
Internal Interrupt-Sampling-Clock to be Provided at Pin COUT/SCIN 4-8
Relationship Between Clock Input at Pin CLK and Counter Output Signals at Pins COUT/SCIN or G0/R0–G3/R6, in Both Pulsed Form and Square Waveform 4-9

1.0 Product Introduction

The NS32202 ICU functions as an overall manager in an interrupt-oriented system environment. Its many features and options permit the design of sophisticated interrupt systems.

Figure 1-1 shows the internal organization of the NS32202. As shown, the NS32202 is divided into five functional blocks. These are described in the following paragraphs:

1.1 I/O BUFFERS AND LATCHES

The I/O Buffers and Latches block is the interface with the system data bus. It contains bidirectional buffers for the data I/O pins. It also contains registers and logic circuits that control the operation of pins G0/IR0, . . . ,G7/IR14 when the ICU is in the 8-bit bus mode.

1.2 READ/WRITE LOGIC AND DECODERS

The Read/Write Logic and Decoders manage all internal and external data transfers for the ICU. These include Data, Control, and Status Transfers. This circuit accepts inputs from the CPU address and control buses. In turn, it issues commands to access the internal registers of the ICU.

1.3 TIMING AND CONTROL

The Timing and Control Block contains status elements that select the ICU operating mode. It also contains state machines that generate all the necessary sequencing and control signals.

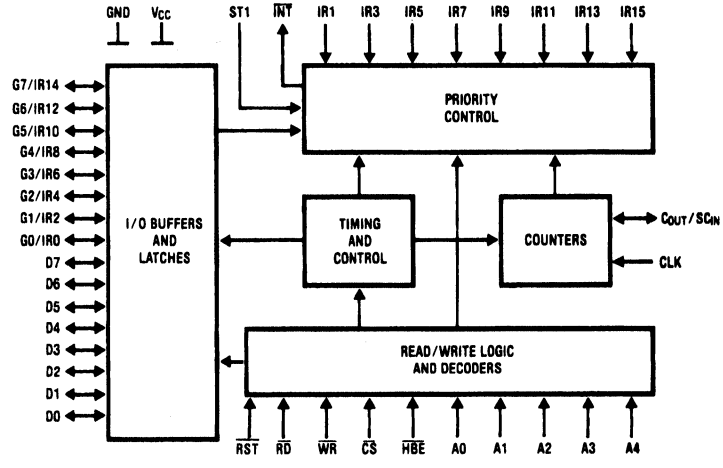
1.4 PRIORITY CONTROL

The Priority Control Block contains 16 units, one for each interrupt position. These units provide the following functions.

- Sensing the various forms of hardware interrupt signals e.g. level (high/low) or edge (rising/falling)
- Resolving priorities and generating an interrupt request to the CPU
- Handling cascaded arrangements
- Enabling software interrupts
- Providing for an automatic return from interrupt
- Enabling the assignment of any interrupt position to the internal counters
- Providing for rearrangement of priorities by assigning the first priority to any interrupt position
- Enabling automatic rotation of priorities

1.5 COUNTERS

This block contains two 16-bit counters, called the H-counter and the L-counter. These are down counters that count from an initial value to zero. Both counters have a 16-bit register (designated HCSV and LCSV) for loading their restarting values. They also have registers containing the current count values (HCCV and LCCV). Both sets of registers are fully described in Section 3.



TL/EE/5117-2

FIGURE 1-1. NS32202 ICU Block Diagram

1.0 Product Introduction (Continued)

The counters are under program control and can be used to generate interrupts. When the count reaches zero, either counter can generate an interrupt request to any of the 16 interrupt positions. The counter then reloads the start value from the appropriate registers and resumes counting. *Figure 1-2* shows typical counter output signals available from the NS32202.

The maximum input clock frequency is 2.5 MHz.

A divide-by-four prescaler is also provided. When the prescaler is used, the input clock frequency can be up to 10 MHz.

When intervals longer than provided by a 16-bit counter are needed, the L- and H-counters can be concatenated to form a 32-bit counter. In this case, both counters are controlled by the H-counter control bits. Refer to the discussion of the Counter Control Register in Section 3 for additional information. *Figure 1-3* summarizes counter read/write operations.

2.0 Functional Description

2.1 RESET

The ICU is reset when a logic low signal is present on the $\overline{RST}$ pin. At reset, most internal ICU registers are affected, and the ICU becomes inactive.

2.2 INITIALIZATION

After reset, the CPU must initialize the NS32202 to establish its configuration. Proper initialization requires knowledge of the ICU register's formats. Therefore, a flowchart of a recommended initialization sequence is shown in (*Figure 3-3*) after the discussion of the ICU registers.

The operation sequence shown in *Figure 3-3* ensures that all counter output pins remain inactive until the counters are completely initialized.

2.3 VECTORED INTERRUPT HANDLING

For details on the operation of the vectored interrupt mode for a particular Series 32000 CPU, refer to the data sheet for

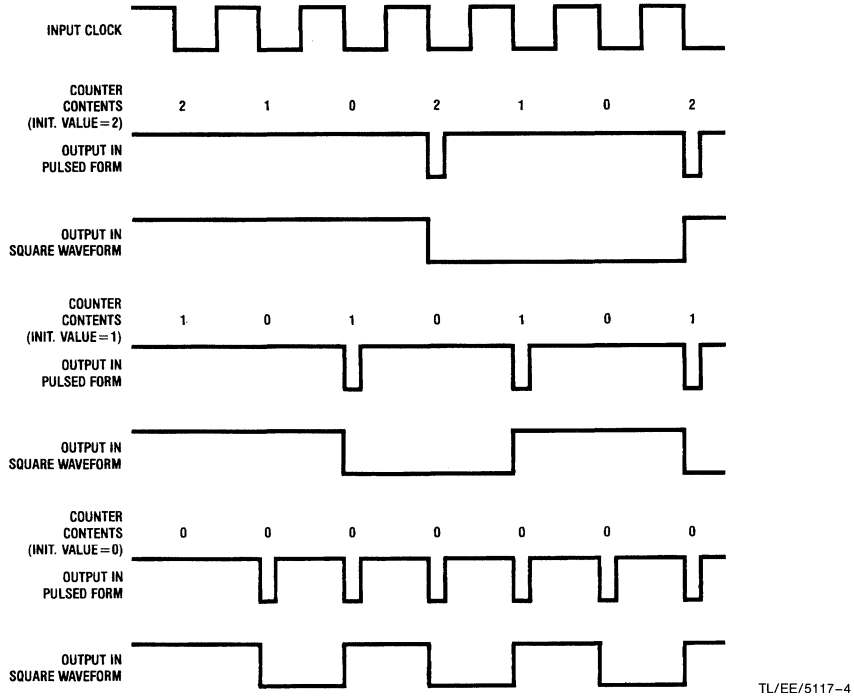
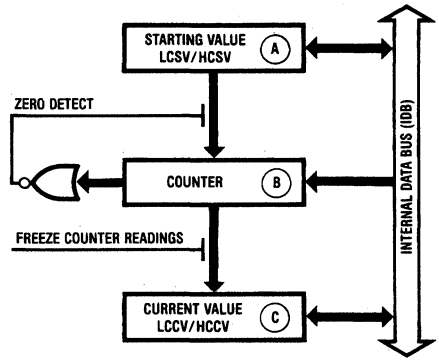


FIGURE 1-2. Counter Output Signals in Pulsed Form and Square Waveform for Three Different Initial Values

2.0 Functional Description (Continued)

that CPU. In this discussion, it is assumed that the NS32202 is working with a CPU in the vectored interrupt mode. Several ICU applications are discussed, including non-cascaded and cascaded operation. *Figures 2-1, 2-2,* and *2-3* show typical configurations of the ICU used with the NS32016 CPU.

A peripheral device issues an interrupt request by sending the proper signal to one of the NS32202 interrupt inputs. If the interrupt input is not masked, the ICU activates its Interrupt Output ($\overline{INT}$) pin and generates an interrupt vector byte. The interrupt vector byte identifies the interrupt source in its four least significant bits. When the CPU detects a low level on its Interrupt Input pin, it performs one or two interrupt acknowledge cycles depending on whether the interrupt request is from the master ICU or a cascaded ICU. *Figure 2-4* shows a flowchart of a typical CPU Interrupt Acknowledge sequence.



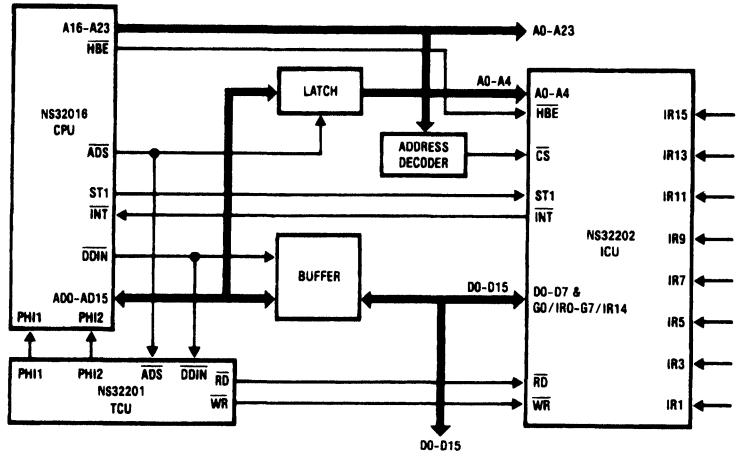
TL/EE/5117-5

BASIC OPERATIONS:

WRITING TO LCSV/HCSV	[A] ← (IDB)
READING LCSV/HCSV	[A] → (IDB)
WRITING TO LCCV/HCCV	[B] ← (IDB)
(only possible when counters are halted)	[C] ← (IDB)
READING LCCV/HCCV	[C] → (IDB)
(only possible when counter readings are frozen)	
COUNTER COUNTS AND READINGS ARE NOT FROZEN	[C] ← [B]
COUNTER RELOADS STARTING VALUE	[B] ← [A]
(occurs on the clock cycle following the one in which it reaches zero)	

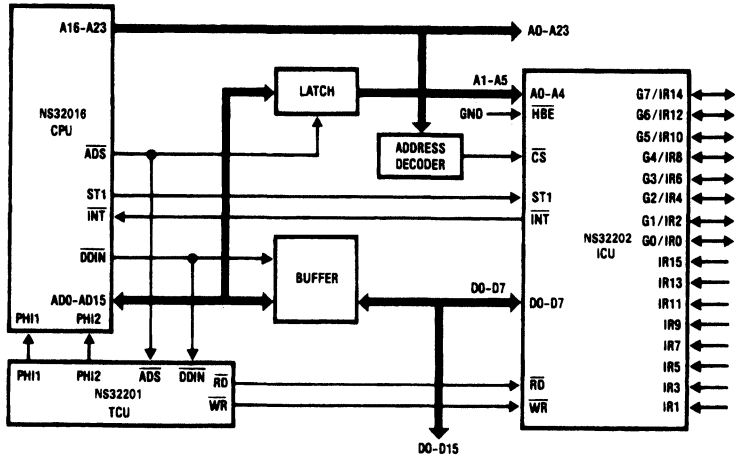
FIGURE 1-3. Counter Configuration and Basic Operations

2.0 Functional Description (Continued)



TL/EE/5117-6

FIGURE 2-1. Interrupt Control Unit Connections in 16-Bit Bus Mode



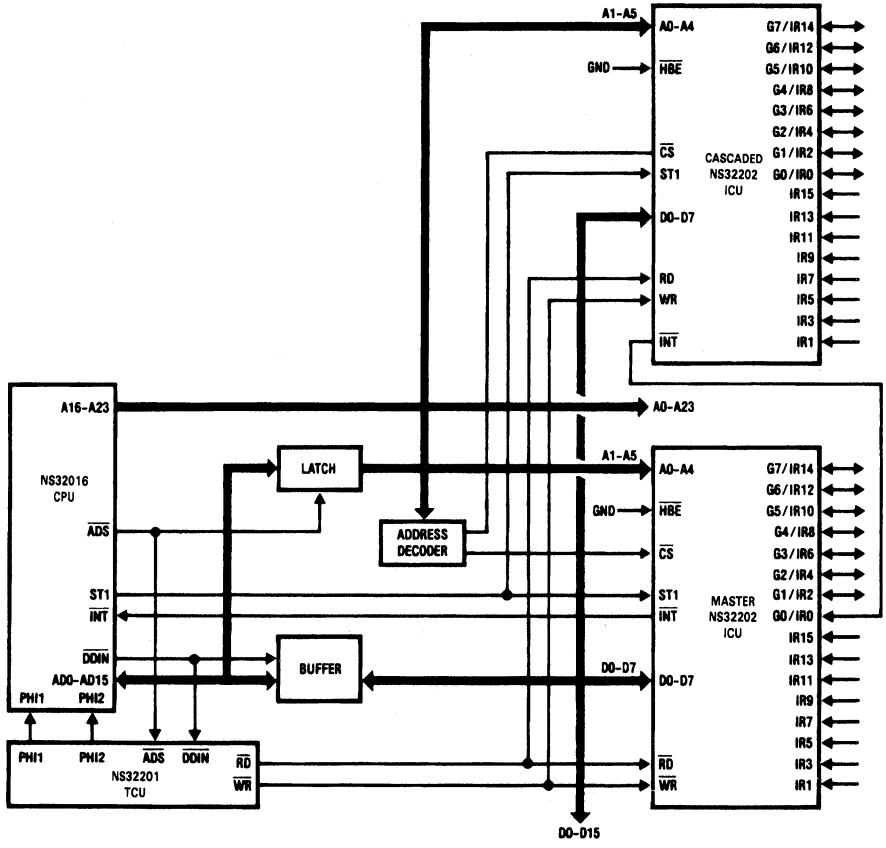
TL/EE/5117-7

NOTE: In the 8-Bit Bus Mode the Master ICU Registers appear at even addresses (A0 = 0) since the ICU communicates with the least significant byte of the CPU data bus.

FIGURE 2-2. Interrupt Control Unit Connections in 8-Bit Bus Mode

4

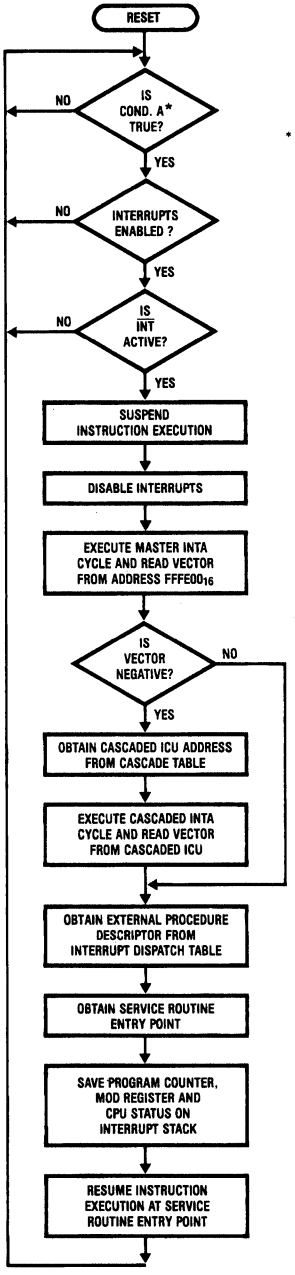
2.0 Functional Description (Continued)



TL/EE/5117–8

FIGURE 2–3. Cascaded Interrupt Control Unit Connections in 8-Bit Bus Mode

2.0 Functional Description (Continued)



* Cond. A is true if current instruction is terminated or an interruptible point in a string instruction is reached.

TL/EE/5117-9

FIGURE 2-4. CPU Interrupt Acknowledge Sequence

4

2.0 Functional Description (Continued)

In general, vectored interrupts are serviced by interrupt routines stored in system memory. The Dispatch Table stores up to 256 external procedure descriptors for the various service procedures. The CPU INTBASE register points to the top of the Dispatch Table. *Figure 2–5* shows the layout of the Dispatch Table. This figure also shows the layout of the Cascade Table, which is discussed with ICU cascaded operation.

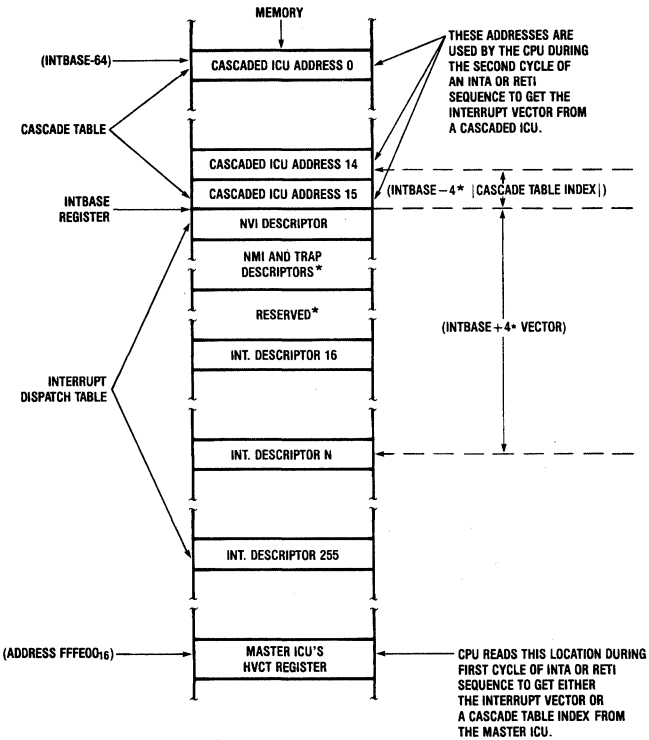
2.3.1 Non-Cascaded Operation. Whenever an interrupt request from a peripheral device is issued directly to the master ICU, a non-cascaded interrupt request to the CPU results. In a system using a single NS32202, up to 16 interrupt requests can be prioritized. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs a Master Interrupt-Acknowledge bus cycle, reading a vector byte from address $FFFE00_{16}$. This vector is then used as an index into the dispatch table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return-from-Interrupt (RET) instruction, which performs a Return-from-Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. *Figure 2–6* shows a typical CPU RETI sequence. In a system with only one ICU, the vectors provided must be in the range of 0 through 127; this can be ensured by writing 0XXXXXXX into the SVCT register. By providing a negative vector value, the master ICU flags the interrupt source as a cascaded ICU (see below).

2.3.2 Cascaded Operation. In cascaded operation, one or more of the interrupt inputs of the master ICU are connected to the Interrupt Output pin of one or more cascaded ICUs. Up to 16 cascaded ICUs may be used, giving a system total of 256 interrupts.

Note: The number of cascaded ICUs is practically limited to 15 because the Dispatch Table for the NS32016 CPU is constructed with entries 1 through 15 either used for NMI and Trap descriptors, or reserved for future use. Interrupt position 0 of the master ICU should not be cascaded, so it can be vectored through Dispatch Table entry 0, reserved for non-vectored interrupts. In this case, the non-vectored interrupt entry (entry 0) is also available for vectored interrupt operation, since the CPU is operating in the vectored interrupt mode.

The address of the master ICU should be $FFFE00_{16}$. (*) Cascaded ICUs can be located at any system address. A list of cascaded ICU addresses is maintained in the Cascade Table as a series of sixteen 32-bit entries.

(*)Note: The CPU status corresponding to both, master interrupt acknowledge and return from interrupt bus cycles, as well as address bit A8, could be used to generate the chip select ($\overline{CS}$) signal for accessing the master ICU during one of the above cycles. In this case the master ICU can reside at any system address. The only limitation is that the least significant 5 or 6 address bits (6 in the 8-bit bus mode) must be zero. The address bit A8 must be decoded to prevent an NMI bus cycle from reading the hardware vector register of the ICU. This could happen, since the NS32016 CPU performs a dummy read cycle from address $FFFF00_{16}$, with the same status as a master INTA cycle, when a non-maskable-interrupt is acknowledged.

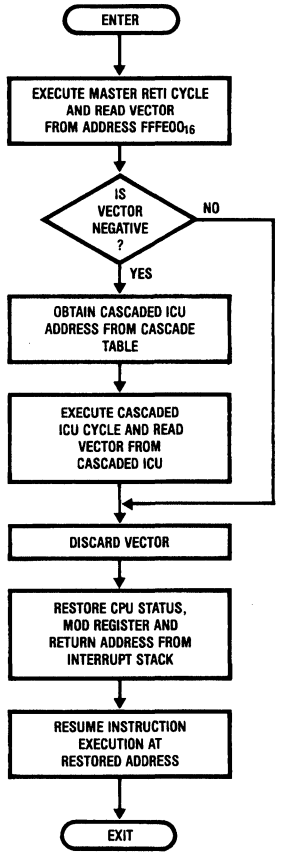


TL/EE/5117–10

* Table entries 1 to 15 should not be used by the ICU since they contain NMI and Trap Descriptors or are reserved for future use. (For more details refer to NS32016 data sheet.)

FIGURE 2–5. Interrupt Dispatch and Cascade Tables

2.0 Functional Description (Continued)



TL/EE/5117–11

FIGURE 2–6. CPU Return from Interrupt Sequence

The master ICU maintains a list (in the CSRC register pair) of its interrupt positions that are cascaded. It also provides a 4-bit (hidden) counter (in-service counter) for each interrupt position to keep track of the number of interrupts being serviced in the cascade ICUs. When a cascaded interrupt input is active, the master ICU activates its interrupt output and the CPU responds with a Master Interrupt Acknowledge Cycle. However, instead of generating a positive interrupt vector, the master ICU generates a negative Cascade Table index.

The CPU interprets the negative number returned from the master ICU as an index into the Cascade Table. The Cascade Table is located in a negative direction from the Dispatch Table, and it contains the virtual addresses of the hardware vector registers for any cascaded NS32202s in the system. Thus, the Cascade Table index supplied by the master ICU identifies the cascaded ICU that requested the interrupt.

Once the cascaded ICU is identified, the CPU performs a Cascaded Interrupt Acknowledge cycle. During this cycle, the CPU reads the final vector value directly from the cascaded ICU, and uses it to access the Dispatch Table. Each cascaded ICU, of course, has its own set of 16 unique interrupt vectors, one vector for each of its 16 interrupt positions.

The CPU interprets the vector value read during a Cascaded Interrupt Acknowledge cycle as an unsigned number. Thus, this vector can be in the range 0 through 255.

When a cascaded interrupt service routine completes its task, it must return control to the interrupted program with the same RETI instruction used in non-cascaded interrupt service routines. However, when the CPU performs a Master Return From Interrupt cycle, the CPU accesses the master ICU and reads the negative Cascade Table index identifying the cascaded ICU that originally received the interrupt request. Using the cascaded ICU address, the CPU now performs a Cascaded Return From Interrupt cycle, informing the cascaded ICU that the service routine is over. The byte provided by the cascaded ICU during this cycle is ignored.

2.4 INTERNAL ICU OPERATING SEQUENCE

The NS32202 ICU accepts two interrupt types, software and hardware.

Software interrupts are initiated when the CPU sets the proper bit in the Interrupt Pending (IPND) registers (R6, R7), located in the ICU. Bits are set and reset by writing the proper byte to either R6 or R7. Software interrupts can be masked, by setting the proper bit in the mask registers (R10, R11).

Hardware interrupts can be either internal or external to the ICU. Internal ICU hardware interrupts are initiated by the on-chip counter outputs. External hardware interrupts are initiated by devices external to the ICU, that are connected to any of the ICU interrupt input pins.

Hardware interrupts can be masked by setting the proper bit in the mask registers (R10, R11). If the Freeze bit (FRZ), located in the Mode Control Register (MCTL), is set, all incoming hardware interrupts are inhibited from setting their corresponding bits in the IPND registers. This prevents the ICU from recognizing any hardware interrupts.

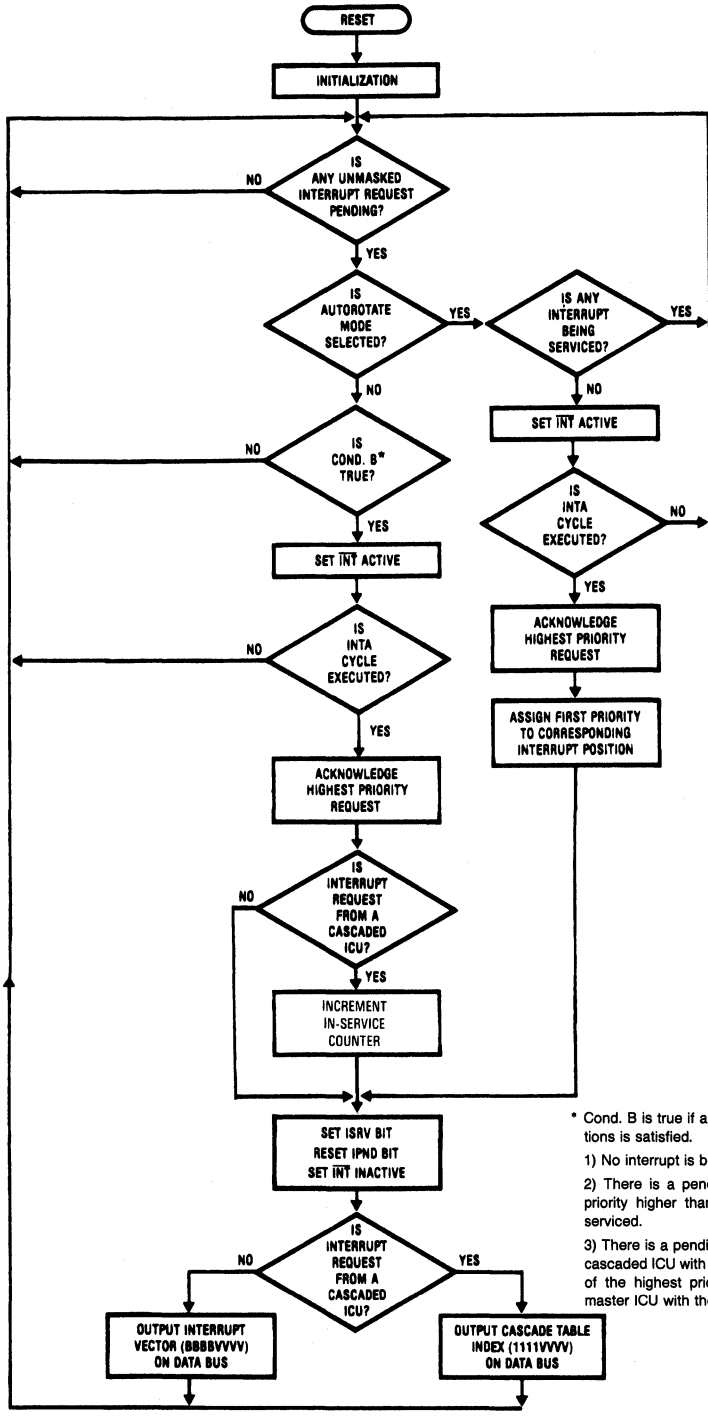
Once the ICU is initialized, it is enabled to accept interrupts. If an active interrupt is not masked, and has a higher priority than any interrupt currently being serviced, the ICU activates its Interrupt Output ($\overline{INT}$). *Figure 2–7* is a flowchart showing the ICU interrupt acknowledge sequence.

The CPU responds to the active $\overline{INT}$ line by performing an Interrupt Acknowledge bus cycle. During this cycle, the ICU clears the IPND bit corresponding to the active interrupt position and sets the corresponding bit in the Interrupt In-Service Registers (ISRV). The 4-bit in-service counter in the master ICU is also incremented by one if the fixed priority mode is selected and the interrupt is from a cascaded ICU. The ISRV bit remains set until the CPU performs a RETI bus cycle and the 4-bit in-service counter is decremented to zero. *Figure 2–8* is a flowchart showing ICU operation during a RETI bus cycle.

When the ISRV bit is set, the $\overline{INT}$ output is disabled. This output remains inactive until a higher priority interrupt position becomes active, or the ISRV bit is cleared.

An exception to the above occurs in the master ICU when the fixed priority mode is selected, and the interrupt input is connected to the $\overline{INT}$ output of a cascaded ICU. In this case the ISRV bit does not inhibit an interrupt of the same priority. This is to allow nesting of interrupts in a cascaded ICU.

2.0 Functional Description (Continued)



* Cond. B is true if any one of the following conditions is satisfied.

1) No interrupt is being serviced

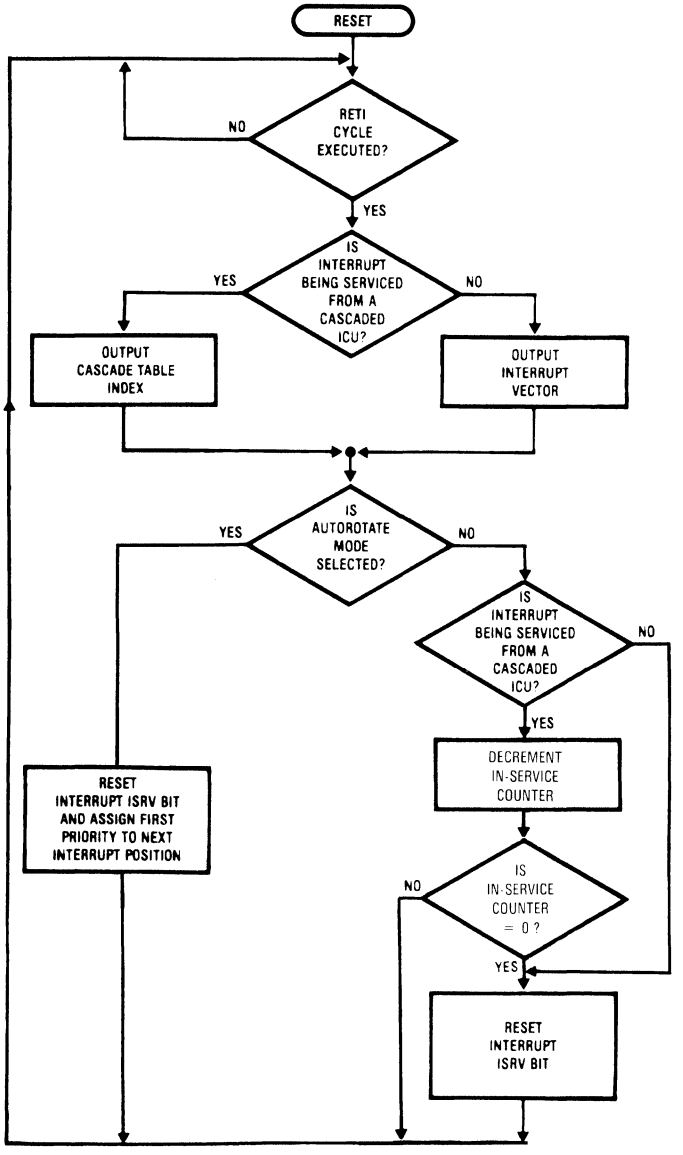
2) There is a pending unmasked interrupt with priority higher than that of the interrupt being serviced.

3) There is a pending unmasked interrupt from a cascaded ICU with priority higher or same as that of the highest priority interrupt position in the master ICU with the ISRV bit set.

TL/EE/5117–12

FIGURE 2–7. ICU Interrupt Acknowledge Sequence

2.0 Functional Description (Continued)



TL/EE/5117–13

FIGURE 2–8. ICU Return from Interrupt Sequence

4

2.0 Functional Description (Continued)

2.5 INTERRUPT PRIORITY MODES

The NS32202 ICU can operate in one of four interrupt priority modes: Fixed Priority; Auto-Rotate; Special Mask; and Polling. Each mode is described below.

2.5.1 Fixed Priority Mode

In the Fixed Priority Mode (also called Fully Nested Mode), each interrupt position is ranked in priority from 0 to 15, with 0 being the highest priority. In this mode, the processing of lower priority interrupts is nested with higher priority interrupts. Thus, while an interrupt is being serviced, any other interrupts of the same or lower priority are inhibited. The ICU does, however, recognize higher priority interrupt requests.

When the interrupt service routine executes its RETI instruction, the corresponding ISRV bit is cleared. This allows any lower priority interrupt request to be serviced by the CPU.

At reset, the default priority assignment gives interrupt IR0 priority 0 (highest priority), interrupt IR1 priority 1, and so forth. Interrupt IR15 is, of course, assigned priority 15, the lowest priority. The default priority assignment can be altered by writing an appropriate value into register FPRT (L) as explained in Section 3.9.

Note: When the ICU generates an interrupt request to the CPU for a higher priority interrupt while a lower priority interrupt is still being serviced by the CPU, the CPU responds to the interrupt request only if its internal interrupt enable flag is set. Normally, this flag is reset at the beginning of an interrupt acknowledge cycle and set during the RETI cycle. If the CPU is to respond to higher priority interrupts during any interrupt service routine, the service routine must set the internal CPU interrupt enable flag, as soon during the service routine as desired.

2.5.2 Auto-Rotate Mode

The Auto Rotate Mode is selected when the NTAR bit is set to 0, and is automatically entered after Reset. In this mode an interrupt source position is automatically assigned lowest priority after a request at that position has been serviced. Highest priority then passes to the next lower priority position. For example, when servicing of the interrupt request at position 3 is completed (ISRV bit 3 is cleared), interrupt position 3 is assigned lowest priority and position 4 assumes highest priority. The nesting of interrupts is inhibited, since the interrupt being serviced always has the highest priority.

This mode is used when the interrupting devices have to be assigned equal priority. A device requesting an interrupt, will have to wait, in the worst case, until each of the 15 other devices has been serviced at most once.

2.5.3 Special Mask Mode

The Special Mask Mode is used when it is necessary to dynamically alter the ICU priority structure while an interrupt is being serviced. For example, it may be desired in a particular interrupt service routine to enable lower priority interrupts during a part of the routine. To do so, the ICU must be programmed in fixed priority mode and the interrupt service routine must control its own in-service bit in the ISRV registers.

The bits of the ISRV registers are changed with either the Set Bit Interlocked or Clear Bit Interlocked instructions (SBITIW or CBITIW). The in-service bit is cleared to enable lower priority interrupts and set to disable them.

Note: For proper operation of the ICU, an interrupt service routine must set its ISRV bit before executing the RETI instruction. This prevents the RETI cycle from clearing the wrong ISRV bit.

2.5.4 Polling Mode

The Polling Mode gives complete control of interrupt priority to the system software. Either some or all of the interrupt positions can be assigned to the polling mode. To assign all interrupt positions to the polling mode, the CPU interrupt enable flag is reset. To assign only some of the interrupt positions to the polling mode, the desired interrupt positions are masked in the Interrupt Mask registers (IMSK). In either case, the polling operation consists of reading the Interrupt Pending (IPND) registers.

If necessary, the IPND read can be synchronized by setting the Freeze (FRZ) bit in the Mode Control register (MCTL). This prevents any change in the IPND registers during the read. The FRZ bit must be reset after the polling operation so the IPND contents can be updated. If an edge-triggered interrupt occurs while the IPND registers are frozen, the interrupt request is latched, and transferred to the IPND registers as soon as FRZ is reset.

The polling mode is useful when a single routine is used to service several interrupt levels.

3.0 Architectural Description

The NS32202 has thirty-two 8-bit registers that can be accessed either individually or in pairs. In 16-bit data bus mode, register pairs can be accessed with the CPU word or double-word reference instructions. *Figure 3-1* shows the ICU internal registers. This figure summarizes the name, function, and offset address for each register.

Because some registers hold similar data, they are grouped into functional pairs and assigned a single name. However, if a single register in a pair is referenced, either an L or an H is appended to the register name. The letters are placed in parentheses and stand for the low order 8 bits (L) and the high order 8 bits (H). For example, register R6, part of the Interrupt Pending (IPND) register pair, is referred to individually as IPND(L).

The following paragraphs give detailed descriptions of the registers shown in *Figure 3-1.*

3.1 HVCT — HARDWARE VECTOR REGISTER (R0)

The HVCT register is a single register that contains the interrupt vector byte supplied to the CPU during an Interrupt Acknowledge (INTA) or Return From Interrupt (RETI) cycle. The HVCT bit map is shown below:

7	6	5	4	3	2	1	0
B	B	B	B	V	V	V	V

3.0 Architectural Description (Continued)

REG. NUMBER AND ADDRESS IN HEX.		REG. NAME	REG. FUNCTION
	R0 (00_{16})	HVCT —	HARDWARE VECTOR
	R1 (01_{16})	SVCT —	SOFTWARE VECTOR
R3 (03_{16})	R2 (02_{16})	ELTG —	EDGE/LEVEL TRIGGERING
R5 (05_{16})	R4 (04_{16})	TPL —	TRIGGERING POLARITY
R7 (07_{16})	R6 (06_{16})	IPND —	INTERRUPTS PENDING
R9 (09_{16})	R8 (08_{16})	ISRV —	INTERRUPTS IN-SERVICE
R11 ($0B_{16}$)	R10 ($0A_{16}$)	IMSK —	INTERRUPT MASK
R13 ($0D_{16}$)	R12 ($0C_{16}$)	CSRC —	CASCADED SOURCE
R15 ($0F_{16}$)	R14 ($0E_{16}$)	FPRT —	FIRST PRIORITY
	R16 (10_{16})	MCTL —	MODE CONTROL
	R17 (11_{16})	OCASN —	OUTPUT CLOCK ASSIGNMENT
	R18 (12_{16})	CIPTR —	COUNTER INTERRUPT POINTER
	R19 (13_{16})	PDAT —	PORT DATA
	R20 (14_{16})	IPS —	INTERRUPT/PORT SELECT
	R21 (15_{16})	PDIR —	PORT DIRECTION
	R22 (16_{16})	CCTL —	COUNTER CONTROL
	R23 (17_{16})	CICTL —	COUNTER INTERRUPT CONTROL
R25 (19_{16})	R24 (18_{16})	LCSV —	L-COUNTER STARTING VALUE
R27 ($1B_{16}$)	R26 ($1A_{16}$)	HCSV —	H-COUNTER STARTING VALUE
R29 ($1D_{16}$)	R28 ($1C_{16}$)	LCCV —	L-COUNTER CURRENT VALUE
R31 ($1F_{16}$)	R30 ($1E_{16}$)	HCCV —	H-COUNTER CURRENT VALUE

FIGURE 3-1. ICU Internal Registers

3.0 Architectural Description (Continued)

The BBBB field is the bias which is programmed by writing $BBBB0000_2$ to the SVCT register (R1). The VVVV field identifies one of the 16 interrupt positions. The contents of the HVCT register provide various information to the CPU, as shown in *Figure 3-2*:

Note 1: The ICU always interprets a read of the HVCT register as either an INTA or RETI cycle. Since these cycles cause internal changes to the ICU, normal programs must never read the ICU HVCT register.

Note 2: If the HVCT register is read with ST1 = 0 (INTA cycle) and no unmasked interrupt is pending, the binary valve BBBB1111 is returned and any pending edge-triggered interrupt in position 15 is cleared.

If the auto-rotate priority mode is selected, the FPRT register is also cleared, thus preventing any interrupt from being acknowledged. In this case a re-intialization of the FPRT register is required for the ICU to acknowledge interrupts again.

If a read of the HVCT register is performed with ST1 = 1 (RETI cycle), the binary valve BBBB1111 is returned.

If the auto-rotate mode is selected, a priority rotation is also performed.

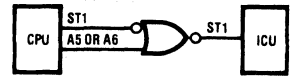
3.2 SVCT — SOFTWARE VECTOR REGISTER (R1)

The SVCT register is a copy of the HVCT register. It allows the programmer to read the contents of the HVCT register without initiating a INTA or RETI cycle in the ICU. It also allows a programmer to change the BBBB field of the HVCT register. The bit map of the SVCT register is the same as for the HVCT register.

During a write to SVCT, the four least significant bits are unaffected while the four most significant bits are written into both SVCT and HVCT (R1 and R0).

The SVCT register is updated dynamically by the ICU. The four least significant bits always contain the vector value that would be returned to the CPU if a INTA or RETI cycle were executed. Therefore, when reading the SVCT register, the state of the CPU ST1 pin is used to select either pending interrupt data or in service interrupt data. For example, if the SVCT register is read with ST1 = 0 (as for an INTA cycle), the VVVV field contains the encoded value of the highest priority pending interrupt. On the other hand, if the SVCT register is read with ST1 = 1, the VVVV field contains the encoded value of the highest priority in-service interrupt.

Note: If the CPU ST1 output is connected directly to the ICU ST1 input, the vector read from SVCT is always the RETI vector. If both the INTA and RETI vectors are desired, additional logic must be added to drive the ICU ST1 input. A typical circuit is shown below. In this circuit, the state of the ICU ST1 input is controlled by both the CPU ST1 output and the selected address bit.



TL/EE/5117-14

3.3 ELTG — EDGE/LEVEL TRIGGERING REGISTERS (R2, R3)

The ELTG registers determine the input trigger mode for each of the 16 interrupt inputs. Each input is assigned a bit in this register pair. An interrupt input is level-triggered if its bit in ELTG is set to 1. The input is edge-triggered if its bit is cleared. At reset, all bits in ELTG are set to 1.

Software interrupt positions are not affected by the state of their ELTG bits.

3.4 TPL — TRIGGERING POLARITY REGISTERS (R4, R5)

The TPL registers determine the polarity of either the active level or the active edge for each of the 16 interrupt inputs. As with the ELTG registers, each input is assigned a bit. Possible triggering modes for the various combinations of ELTG and TPL bits are shown below.

ELTG BIT	TPL BIT	TRIGGERING MODE
0	0	Falling Edge
0	1	Rising Edge
1	0	Low Level
1	1	High Level

Software interrupt positions are not affected by their TPL bits. At reset, all TPL bits are set to 0.

Note 1: If edged-triggered interrupts are to be handled, the TPL register should be programmed before the ELTG register.

This prevents spurious interrupt requests from being generated during the ICU initialization from edge-triggered interrupt positions.

Note 2: Hardware interrupt inputs connected to cascaded ICUs must have their TPL bits set to 0.

3.5 IPND — INTERRUPT PENDING REGISTERS (R6, R7)

The IPND registers track interrupt requests that are pending but not yet serviced. Each interrupt position is assigned a bit in IPND. When an interrupt is pending, the corresponding bit in IPND is set. The IPND data are used by the ICU to generate interrupts to the CPU. These data are also used in polling operations.

	INTA CYCLE (ST1 = 0)		RETI CYCLE (ST1 = 1)	
	Highest priority pending interrupt is from:		Highest priority in-service interrupt was from:	
BBBB	cascaded ICU	any other source	cascaded ICU	any other source
	1111	programmed bias*	1111	programmed bias*
VVVV	encoded value of the highest priority pending interrupt		encoded value of the highest priority in-service interrupt	

*The Programmed bias for the master ICU must range from 0000 to 0111_2 because the CPU interprets a one in the most significant bit position as a Cascade Table Index indicator for a cascaded ICU.

FIGURE 3-2. HVCT Register Data Coding

3.0 Architectural Description (Continued)

The IPND registers are also used for requesting software interrupts. This is done by writing specially formatted data bytes to either IPND(L) or IPND(H). The formats differ for registers R6 and R7. These formats are shown below:

IPND(L) (R6) — S0000PPP

IPND(H) (R7) — S0001PPP

Where: S = Set (S = 1) or Clear (S = 0)

PPP = is a binary number identifying one of eight bits

Note: The data read from either R6 or R7 are different from that written to the register because the ICU returns the register contents, rather than the formatted byte used to set the register bits.

The ICU automatically clears a set IPND bit when the pending interrupt request is serviced. All pending interrupts in a register can be cleared by writing the pattern 'X1XXXXXX' to it (X = don't care). To avoid conflicts with asynchronous hardware interrupt requests, the IPND registers should be frozen before pending interrupts are cleared. Refer to the Mode Control Register description for details on freezing the IPND registers.

At reset, all IPND bits are set to 0.

Note: The edge sensing mechanism used for hardware interrupts in the NS32202 ICU is a latching device that can be cleared only by acknowledging the interrupt or by changing the trigger mode to level sensing. Therefore, before clearing pending interrupts in the IPND registers, any edge-triggered interrupt inputs must first be switched to the level-triggered mode. This clears the edge-triggered interrupts; the remaining interrupts can then be cleared in the manner described above. This applies to clearing the interrupts only. Edge-triggered interrupts can be set without changing the trigger mode.

3.6 ISRV — INTERRUPT IN-SERVICE REGISTERS (R8, R9)

The ISRV registers track interrupt requests that are currently being serviced. Each interrupt position is assigned a bit in ISRV. When an interrupt request is serviced by the ICU, its corresponding bit is set in the ISRV registers. Before generating an interrupt to the CPU, the ICU checks the ISRV registers to ensure that no higher priority interrupt is currently being serviced.

Each time the CPU executes an RETI instruction, the ICU clears the ISRV bit corresponding to the highest priority interrupt in service. The ISRV registers can also be written into by the CPU. This is done to implement the special mask priority mode.

At reset, the ISRV registers are set to 0.

Note: If the ICU initialization does not follow a hardware reset, the ISRV register should be cleared during initialization by writing zeroes into it.

3.7 IMSK — INTERRUPT MASK REGISTERS (R10, R11)

Each NS32202 interrupt position can be individually masked. A masked interrupt source is not acknowledged by the ICU. The IMSK registers store a mask bit for each of the ICU interrupt positions. If an interrupt position's IMSK bit is set to 1, the position is masked.

The IMSK registers are controlled by the system software. At reset, all IMSK bits are set to 1, disabling all interrupts.

Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the IMSK register. However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem, the above operation should be performed with the CPU interrupt disabled.

3.8 CSRC — CASCADED SOURCE REGISTERS (R12, R13)

The CSRC registers track any cascaded interrupt positions. Each interrupt position is assigned a bit in the CSRC registers. If an interrupt position's CSRC bit is set, that position is connected to the $\overline{INT}$ output of another NS32202 ICU, i.e., it is a cascaded interrupt.

At reset, the CSRC registers are set to 0.

Note 1: If any cascaded ICU is used, the CSRC register should be cleared during initialization (if the initialization does not follow a hardware reset) by writing zeroes into it. This should be done before setting the bits corresponding to the cascaded interrupt positions. This operation ensures that the 4-bit in-service counters (associated with each interrupt position to keep track of cascaded interrupts) always get cleared when the ICU is re-initialized.

Note 2: Only the Master ICU should have any CSRC bits set. If CSRC bits are set in a cascaded ICU, incorrect operation results.

3.9 FPRT — FIRST PRIORITY REGISTERS (R14, R15)

The FPRT registers track the ICU interrupt position that currently holds first priority. Only one bit of the FPRT registers is set at one time. The set bit indicates the interrupt position with first (highest) priority.

The FPRT registers are automatically updated when the ICU is in the auto-rotate mode. The first priority interrupt can be determined by reading the FPRT registers. This operation returns a 16-bit word with only one bit set. An interrupt position can be assigned first priority by writing a formatted data byte to the FPRT(L) register. The format is shown below:

7	6	5	4	3	2	1	0
X	X	X	X	F	F	F	F

Where: XXXX = Don't Care

FFFF = A binary number from 0 to 15 indicating the interrupt position assigned first priority.

Note: The byte above is written only to the FPRT(L) register. Any data written to FPRT(H) is ignored.

At reset the FFFF field is set to 0, thus giving interrupt position 0 first priority.

3.10 MCTL — MODE CONTROL REGISTER (R16)

The contents of the MCTL set the operating mode of the NS32202 ICU. The MCTL bit map is shown below.

7	6	5	4	3	2	1	0
CFRZ	COUTD	COUTM	CLKM	FRZ	unused	NTAR	T16N8

3.0 Architectural Description (Continued)

CFRZ Determines whether or not the NS32202 counter readings are frozen. When frozen, the counters continue counting but the LCCV and HCCV registers are not updated. Reading of the true value of LCCV and HCCV is possible only while they are frozen.

CFRZ = 0 => LCCV and HCCV Not Frozen

CFRZ = 1 => LCCV and HCCV Frozen

COUTD Determines whether the COUT/SCIN pin is an input or an output. COUT/SCIN should be used as an input only for testing purposes. In this case an external sampling clock must be provided otherwise hardware interrupts will not be recognized.

COUTD = 0 => COUT/SCIN is Output

COUTD = 1 => COUT/SCIN is Input

COUTM When the COUT/SCIN pin is programmed as an output (COUTD=0), this bit determines whether the output signal is in pulsed form or in square wave form.

COUTM = 0 => Square Wave Form

COUTM = 1 => Pulsed Form

CLKM Used only in the 8-bit Bus Mode. This bit controls the clock wave form on any of the pins GO/IRO, . . . ,G3/IR6 programmed as counter output.

CLKM = 0 => Square Wave Form

CLKM = 1 => Pulsed Form

FRZ Freeze Bit. In order to allow a synchronous reading of the interrupt pending registers (IPND), their status may be frozen, causing the ICU to ignore incoming requests. This is of special importance if a polling method is used.

FRZ = 0 => IPND Not Frozen

FRZ = 1 => IPND Frozen

NTAR Determines whether the ICU is in the AUTO-ROTATE or FIXED Priority Mode. In AUTO-ROTATE mode, the interrupt source at the highest priority position, after being serviced, is assigned automatically lowest priority. In this mode, the interrupt in service always has highest priority and nesting of interrupts is therefore inhibited.

NTAR = 0 => Auto-Rotate Mode

NTAR = 1 => Fixed Mode

T16N8 Controls the data bus mode of operation.

T16N8 = 0 => 8-Bit Bus Mode

T16N8 = 1 => 16-Bit Bus Mode

At reset, all MCTL bits except COUTD, are reset to 0. COUTD is set to 1.

3.11 OCASN — OUTPUT CLOCK ASSIGNMENT REGISTER (R17)

Used only in the 8-bit Bus Mode. The four least significant bits of this register control the output clock assignments on pins G0/IR0, . . . ,G3/IR6. If any of these bits is set to 1, the clock generated by either the H-Counter or the H+L-Counter will be output to the corresponding pin. The four most significant bits of OCASN are not used. At Reset the four least significant bits are set to 0.

Note: The interrupt sensing mechanism on pins G0/IR0, . . . ,G3/IR6 is not disabled when any of these pins is programmed as clock output. Thus, to avoid spurious interrupts, the corresponding bits in register IPS should also be set to zero.

3.12 CIPTR — COUNTER INTERRUPT POINTER REGISTER (R18)

The CIPTR register tracks the assignment of counter outputs to interrupt positions. A bit map of this register is shown below.

7	6	5	4	3	2	1	0
H	H	H	H	L	L	L	L

Where: HHHH = A 4-bit binary number identifying the interrupt position assigned to the H-Counter (or the H+L-counter if the counters are concatenated).

LLLL = A 4-bit binary number identifying the interrupt position assigned to the L-counter.

Note: Assignment of a counter output to an interrupt position also requires control bits to be set in the CICTL register. If a counter output is assigned to an interrupt position, external hardware interrupts at that position are ignored.

At reset, all bits in the CIPTR are set to 1. (This means both counters are assigned to interrupt position 15.)

3.13 PDAT — PORT DATA REGISTER (R19)

Used only in the 8-bit Bus Mode. This register is used to input or output data through any of the pins G0/IR0, . . . ,G7/IR14 programmed as I/O ports by the IPS register. Any pin programmed as an output delivers the data written into PDAT. The input pins ignore it. Reading PDAT provides the logical value of all I/O pins, INPUT and OUTPUT.

3.14 IPS — INTERRUPT/PORT SELECT REGISTER (R20)

Used only in the 8-bit Bus Mode. This register controls the function of the pins G0/IR0, . . . ,G7/IR14. Each of these pins is individually programmed as an I/O port, if the corresponding bit of IPS is 0; as an interrupt source, if the corresponding bit is 1. The assignment of the H-Counter output to G0/IR0, . . . ,G3/IR6 by means of reg. OCASN overrides the assignment to these pins as I/O ports or interrupt inputs.

At Reset, all the IPS bits are set to 1.

Note: Whenever a bit in the IPS register is set to zero, to program the corresponding pin as an I/O port, any pending interrupt on the corresponding interrupt position will be cleared.

3.15 PDIR — PORT DIRECTION REGISTER (R21)

Used only in the 8-bit Bus Mode. This register determines the direction of any of the pins G0/IR0, . . . ,G7/IR14 programmed as I/O ports by the IPS register. A logic 1 indicates an input, while a logic 0 indicates an output.

At Reset, all the PDIR bits are set to 1.

3.16 CCTL — COUNTER CONTROL REGISTER (R22)

The CCTL register controls the operating modes of the counters. A bit map of CCTL is shown below.

7	6	5	4	3	2	1	0
CCON	CFNPS	COUT1	COUT0	CRUNH	CRUNL	CDCRH	CDCRL

CCON Determines whether the counters are independent or concatenated to form a single 32-bit counter (H+L-Counter). If a 32-bit counter is selected, the bits corresponding to the H-

3.0 Architectural Description (Continued)

Counter will control the H+L-Counter, while the bits corresponding to the L-Counter are not used.

CCON = 0 => Two 16-bit Counters

CCON = 1 => One 32-bit Counter

CFNPS — Determines whether the external clock is prescaled or not.

CFNPS = 0 => Clock Prescaled (divided by 4)

CFNPS = 1 => Clock Not Prescaled.

COUT1 & COUT0 — These bits are effective only when the COUT/SCIN pin is programmed as an OUTPUT (COUTD bit in reg. MCTL is 0). Their logic levels are decoded to provide different outputs for COUT/SCIN, as detailed in the table below:

COUT1	COUT0	COUT/SCIN Output Signal
0	0	Internal Sampling Oscillator
0	1	Zero Detect Of L-Counter
1	0	Zero Detect Of H-Counter
1	1	Zero Detect Of H+L-Counter*

*If the H- and L-Counters are not concatenated and COUT1/COUT0 are both 1, the COUT/SCIN pin is active when either counter reaches zero.

CRUNH — Determines the state of either the H-Counter or the H+L-Counter, depending upon the status of CCON.

CRUNH = 0 => H-Counter or H+L-Counter Halted

CRUNH = 1 => H-Counter or H+L-Counter Running

CRUNL — Effective only when CCON = 0. This bit determines whether the L-Counter is running or halted.

CRUNL = 0 => L-Counter Halted

CRUNL = 1 => L-counter Running

CDCRH — Effective only when CRUNH = 0 (Counter Halted). This bit is the single cycle decrement signal for either the H-Counter or the H+L-Counter.

CDCRH = 0 => No Effect

CDCRH = 1 => Decrement H-Counter or H+L-Counter

CDCRL — Effective only when CRUNL = 0 and CCON = 0. This bit is the single cycle decrement signal for the L-Counter.

CDCRL = 0 => No Effect

CDCRL = 1 => Decrement L-Counter

Note: The bits CDCRL and CDCRH are set when a logic 1 is written into them, but, they are automatically cleared after the end of the write operation. This is needed to accomplish the decrement operation. Therefore, these bits always contain 0 when read.

Reset does not affect the CCTL bits.

3.17 CICTL — COUNTER INTERRUPT CONTROL REGISTER (R23)

The CICTL register controls the counter interrupts and records counter interrupt status. Interrupts can be generated from either of the 16-bit counters. When the counters are concatenated, the interrupt control is through the H-Counter control bits. In this case the CIEL bit should be set to zero to avoid spurious interrupts from the L-Counter. A bit map of the CICTL register is shown following.

7	6	5	4	3	2	1	0
CERH	CIRH	CIEH	WENH	CERL	CIRL	CIEL	WENL

CERH — H-Counter Error Flag. This bit is set (1) when a second interrupt request from the H-Counter (or H+L-Counter) occurs before the first request is acknowledged.

CIRH — H-Counter Interrupt Request. It is set (1) when an interrupt is pending from the H-Counter (or H+L-Counter). It is automatically reset when the interrupt is acknowledged.

CIEH — H-Counter Interrupt Enable. When it is set, the H-Counter (or H+L-Counter) interrupt is enabled.

WENH — H-Counter Control Write Enable. When WEHN is set (1), bits CERH, CIRH, and CIEH can be written.

CERL — L-Counter Error Flag. This bit is set (1) when a second interrupt request from the L-Counter occurs before the first request is acknowledged.

CIRL — L-Counter Interrupt Request. It is set (1) when an interrupt is pending from the L-Counter. It is automatically reset when the interrupt is acknowledged.

CIEL — L-Counter Interrupt Enable. When it is set (1), the L-Counter interrupt is enabled.

WENL — L-Counter Control Write Enable. When WENL is set (1), bits CERL, CIRL, and CIEL can be written.

Note: Setting the write enable bits (WENH or WENL) and writing any of the other CICTL bits are concurrent operations. That is, the ICU will ignore any attempt to alter CICTL bits if the proper write enable bit is not set in the data byte.

At reset, all CICTL bits are set to 0. However, if the counters are running, the bits CIRL, CERL, CIRH and CERH may be set again after the reset signal is removed.

3.18 LCSV/HCSV — L-COUNTER STARTING VALUE/ H-COUNTER STARTING VALUE REGISTERS (R24, R25, R26, AND R27)

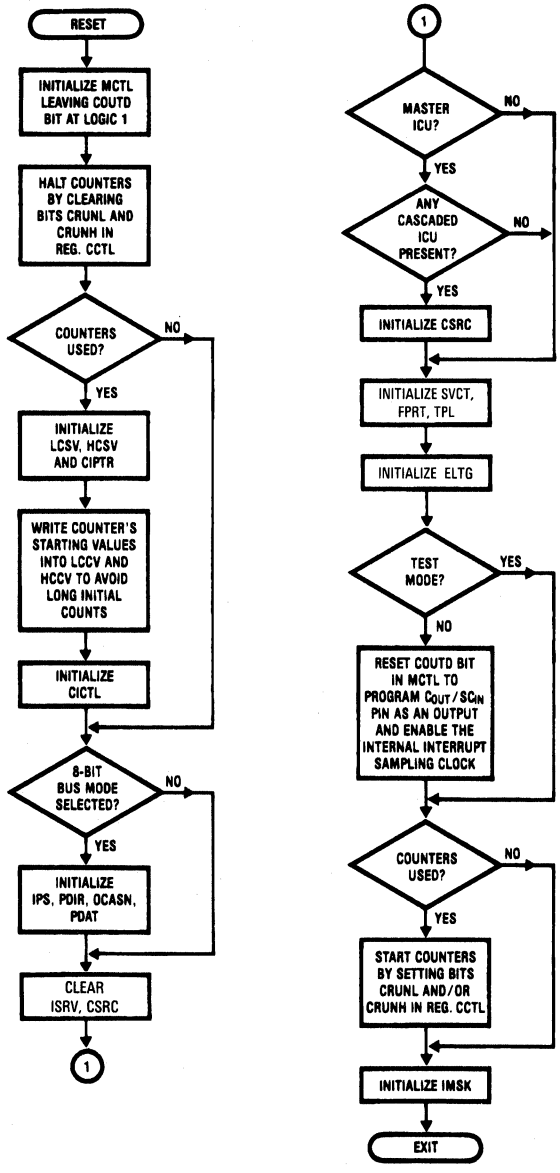
The LCSV and HCSV registers store the start values for the L-Counter and H-Counter, respectively. Each time a counter reaches zero, the start value is automatically reloaded from either LCSV or HCSV, one clock cycle after zero count is reached. Loading LCSV or HCSV from the CPU must be synchronized to avoid writing the registers while the reloading of the counters is occurring. One method is to halt the counters while the registers are loaded.

When the 16-bit counters are concatenated, the LCSV and HCSV registers hold the 32-bit start count, with the least significant byte in R24 and the most significant byte in R27.

3.19 LCCV/HCCV — L-COUNTER CURRENT VALUE/ H-COUNTER CURRENT VALUE REGISTERS (R28, R29, R30, AND R31)

The LCCV and HCCV registers hold the current value of the counters. If the CFRZ bit in the MCTL register is reset (0), these registers are updated on each clock cycle with the current value of the counters. LCCV and HCCV can be read only when the counter readings are frozen (CFRZ bit in the

3.0 Architectural Description (Continued)



TL/EE/5117–15

FIGURE 3–3. Recommended ICU's Initialization Sequence

3.0 Architectural Description (Continued)

MCTL register is 1). They can be written only when the counters are halted (CRUNL and/or CRUNH bits in the CCTL register are 0). This last feature allows new initial count values to be loaded immediately into the counters, and can be used during initialization to avoid long initial counts.

When the 16-bit counters are concatenated, the LCCV and HCCV registers hold the 32-bit current value, with the least significant byte in R28 and the most significant byte in R31.

3.20 REGISTER INITIALIZATION

Figure 3-3 shows a recommended initialization procedure for the ICU that sets up all the ICU registers for proper operation.

4.0 Device Specifications

4.1 NS32202 PIN DESCRIPTIONS

4.1.1 Power Supply

Power (V_{CC}): +5V DC Supply

Ground (GND): Power Supply Return

4.1.2 Input Signals

Reset ($\overline{RST}$): Active low. This signal initializes the ICU. (The ICU initializes to the 8-bit bus mode.)

Chip Select ($\overline{CS}$): Active low. This signal enables the ICU to respond to address, data, and control signals from the CPU.

Addresses (A0 through A4): Address lines used to select the ICU internal registers for read/write operations.

High Byte Enable ($\overline{HBE}$): Active low. Enables data transfers on the most-significant byte of the Data Bus. If the ICU is in the 8-bit Bus Mode, this signal is not used and should be connected to either GND or V_{CC}.

Read ($\overline{RD}$): Active low. Enables data to be read from the ICU's internal registers.

Write ($\overline{WR}$): Active low. Enables data to be written into the ICU's internal registers.

Status (ST1): Status signal from the CPU. When the Hardware Vector Register is read, this signal differentiates an INTA cycle from an RETI cycle. If ST1 = 0 the ICU initiates an INTA cycle. If ST1 = 1 an RETI cycle will result.

Interrupt Requests (IR1, IR3..., IR15): These eight inputs are used for hardware interrupts. Each may be individually triggered in one of four modes: Rising Edge, Falling Edge, Low Level, or High Level.

Counter Clock (CLK): External clock signal to drive the ICU internal counters.

4.1.3 Output Signals

Interrupt Output ($\overline{INT}$): Active low. This signal indicates that an interrupt is pending.

4.1.4 Input/Output Signals

Data Bus 0-7 (D0 through D7): Eight low-order data bus lines used in both 8-bit and 16-bit bus modes.

General Purpose I/O Lines (G0/IR0, G1/IR2,...,G7/IR14): These pins are the high-order data bits when the ICU is in the 16-bit bus mode. When the ICU is in the 8-bit bus mode, each of these can be individually assigned one of the following functions:

- Additional Hardware Interrupt Input (IR0 through IR14)
- General Purpose Data Input
- General Purpose Data Output
- Clock Output from H-Counter (Pins G0/IR0 through G3/IR6 only)

It should be noted that, for maximum flexibility in assigning interrupt priorities, the interrupt positions corresponding to pins G0/IR0,...,G7/IR14 and IR1,...,IR15 are interleaved.

Counter or Oscillator Output/Sampling Clock Input (COUT/SCIN): As an output, this pin provides either a clock signal generated by the ICU internal oscillator, or a zero detect signal from one or both of the ICU counters. As an input, it is used for an external clock, to override the internal oscillator used for interrupt sampling. This is done only for testing purposes.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7.0V
Power Dissipation	1.5 Watt

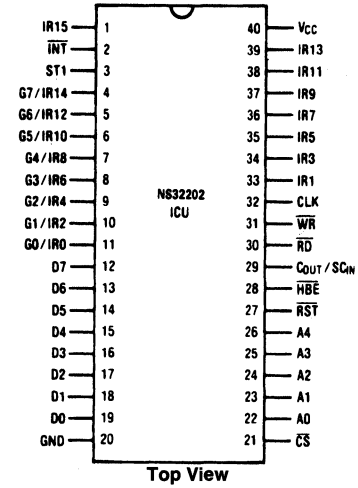
Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS

T_A = 0° to 70°C, V_{CC} = +5V ± 5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IL}	Input Low Voltage				0.8	V
V_{IH}	Input High Voltage		2.0			V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA			0.45	V
V_{OH}	Output High Voltage	I_{OH} = −400 μA	2.4			V
$I_{O\ (OFF)}$	Output Leakage Current (Output Pins in TRI-STATE Condition)	0.4 ≤ V_{OUT} ≤ V_{CC}	−20		20	μA
I_I	Input Load Current	V_{in} = 0 to V_{CC}	−20		20	μA
I_{CC}	Power Supply Current	I_{out} = 0, T = 0°C			300	mA

Connection Diagram



TL/EE/5117–3

Top View

Order Number NS32202D-6, NS32202D-10

See NS Package Number D40C

FIGURE 4–1

4.0 Device Specifications (Continued)

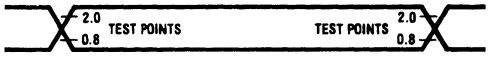
4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 0.8V or 2.0V on the input and output signals as illustrated in *Figure 1*, unless specifically stated otherwise.

Abbreviations:

L.E.—leading edge R.E.—rising edge
T.E.—trailing edge F.E.—falling edge



TL/EE/5117-16

FIGURE 4-2. Timing Specification Standard

4.4.1.1 Timing Tables

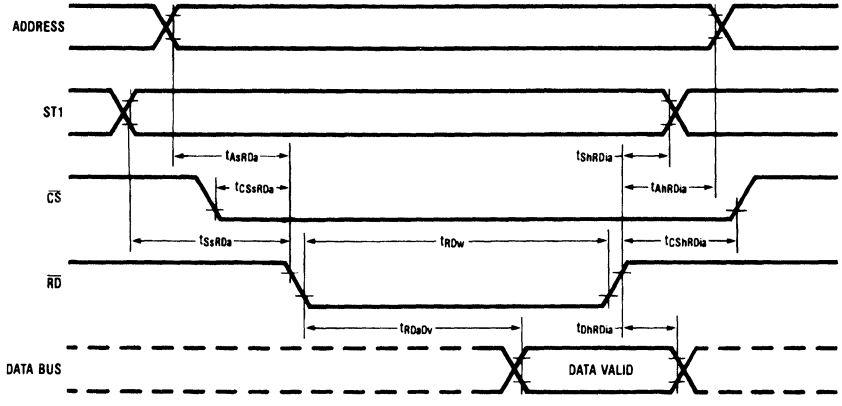
Symbol	Figure	Description	Reference/Conditions	NS32202-6		NS32202-8		NS32202-10		Units
				Min	Max	Min	Max	Min	Max	
READ CYCLE										
t_{AhRDia}	4-3	Address Hold Time	After $\overline{RD}$ T.E.	80		80		80		ns
t_{AsRDa}	4-3	Address Setup Time	Before $\overline{RD}$ L.E.	50		50		40		ns
$t_{CShRDia}$	4-3	$\overline{CS}$ Hold Time	After $\overline{RD}$ T.E.	80		80		80		ns
t_{CSsRDa}	4-3	$\overline{CS}$ Setup Time	Before $\overline{RD}$ L.E.	50		50		30		ns
t_{DhRDia}	4-3	Data Hold Time	After $\overline{RD}$ T.E.	0	50	0	50	0	50	ns
t_{RDaDv}	4-3	Data Valid	After $\overline{RD}$ L.E.		200		175		150	ns
t_{RDw}	4-3	$\overline{RD}$ Pulse Width	At 0.8V (Both Edges)	220		190		160		ns
t_{SsRDa}	4-3	ST1 Setup Time	Before $\overline{RD}$ L.E.	50		50		50		ns
t_{ShRDia}	4-3	ST1 Hold Time	After $\overline{RD}$ T.E.	−30		−30		−30		ns
WRITE CYCLE										
t_{AhWRia}	4-4	Address Hold Time	After $\overline{WR}$ T.E.	80		80		80		ns
t_{AsWRa}	4-4	Address Setup Time	Before $\overline{WR}$ L.E.	50		50		50		ns
$t_{CShWRia}$	4-4	$\overline{CS}$ Hold Time	After $\overline{WR}$ T.E.	80		80		80		ns
t_{CSsWRa}	4-4	$\overline{CS}$ Setup Time	Before $\overline{WR}$ L.E.	50		50		50		ns
t_{DhWRia}	4-4	Data Hold Time	After $\overline{WR}$ T.E.	60		60		50		ns
t_{DsWRia}	4-4	Data Setup Time	Before $\overline{WR}$ T.E.	150		125		100		ns
t_{WRiaPf}	4-4	Port Output Floating	After $\overline{WR}$ T.E. (To PDIR)		200		200		200	ns
t_{WRiaPv}	4-4	Port Output Valid	After $\overline{WR}$ T.E.		200		200		200	ns
t_{WRw}	4-4	$\overline{WR}$ Pulse Width	At 0.8V (Both Edges)	220		190		160		ns

4.0 Device Specifications (Continued)

4.4.1.1 Timing Tables (Continued)

Symbol	Figure	Description	Reference/Conditions	NS32202-6		NS32202-8		NS32202-10		Units
				Min	Max	Min	Max	Min	Max	
OTHER TIMINGS										
t_{COUTl}	4-8	Internal Sampling Clock Low Time	At 0.8V (Both Edges)	50		50		50		ns
t_{COUTp}	4-8	Internal Sampling Clock Period		400		400		400		ns
t_{SCINh}	4-7	External Sampling Clock High Time	At 2.0V (Both Edges)	100		100		100		ns
t_{SCINl}	4-7	External Sampling Clock Low Time	At 0.8V (Both Edges)	100		100		100		ns
t_{SCINp}	4-7	External Sampling Clock Period		800		800		800		ns
t_{Ch}	4-9	External Clock High Time (Without Prescaler)	At 2.0V (Both Edges)	160		130		100		ns
t_{Chp}	4-9	External Clock High Time (With Prescaler)	At 2.0V (Both Edges)	80		65		50		ns
t_{Cl}	4-9	External Clock Low Time (Without Prescaler)	At 0.8V (Both Edges)	160		130		100		ns
t_{Clp}	4-9	External Clock Low Time (With Prescaler)	At 0.8V (Both Edges)	80		65		50		ns
t_{Cy}	4-9	External Clock Period (Without Prescaler)		650		500		400		ns
t_{Cyp}	4-9	External Clock Period (With Prescaler)		160		130		100		ns
t_{GCOUTl}	4-9	Counter Output Transition Delay	After CLK F.E.		300		300		300	ns
t_{COUTw}	4-9	Counter Output Pulse Width in Pulsed Form	At 0.8V (Both Edges)	50		50		50		ns
t_{ACKIR}	4-5	Interrupt Request Delay	After Previous Interrupt Acknowledge	500		500		500		ns
t_{IRId}	4-5	$\overline{INT}$ Output Delay	After Interrupt Request Active		800		800		800	ns
t_{IRw}	4-5	Interrupt Request Pulse Width in Edge Trigger		50		50		50		ns
t_{RSTw}		$\overline{RST}$ Pulse Width	At 0.8V (Both Edges)	400		400		400		ns

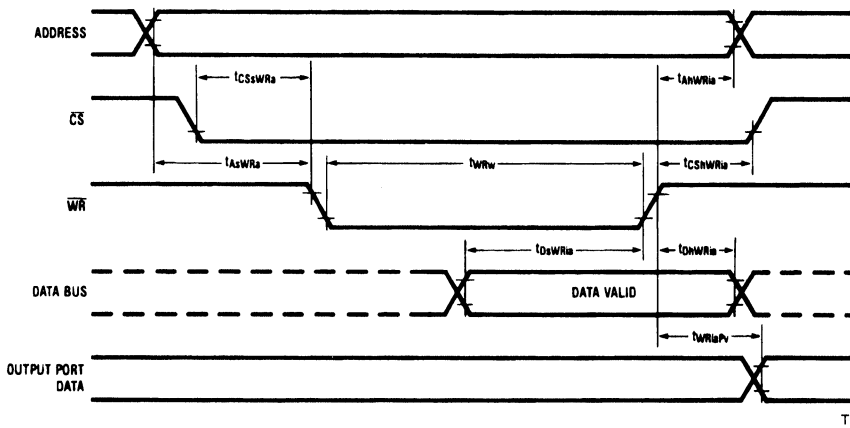
4.4.1.2 Timing Diagrams



TL/EE/5117–17

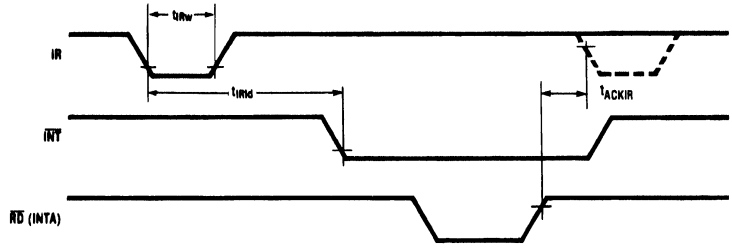
FIGURE 4–3. READ/INTA Cycle

4.0 Device Specifications (Continued)



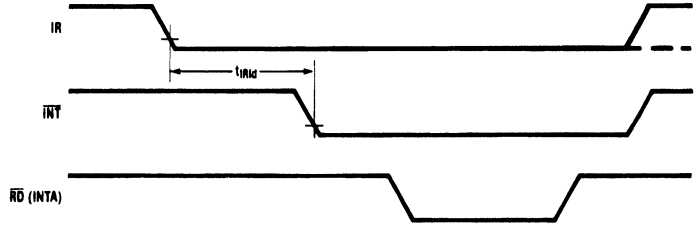
TL/EE/5117-18

FIGURE 4-4. Write Cycle



TL/EE/5117-19

FIGURE 4-5. Interrupt Timing in Edge Triggering Mode

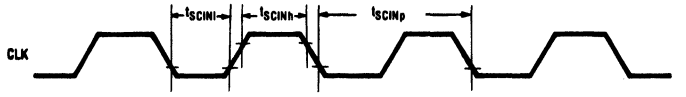


TL/EE/5117-20

FIGURE 4-6. Interrupt Timing in Level Triggering Mode

4

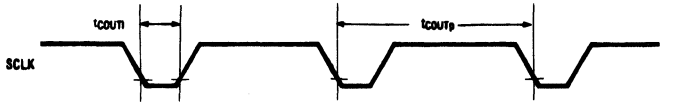
4.0 Device Specifications (Continued)



TL/EE/5117–21

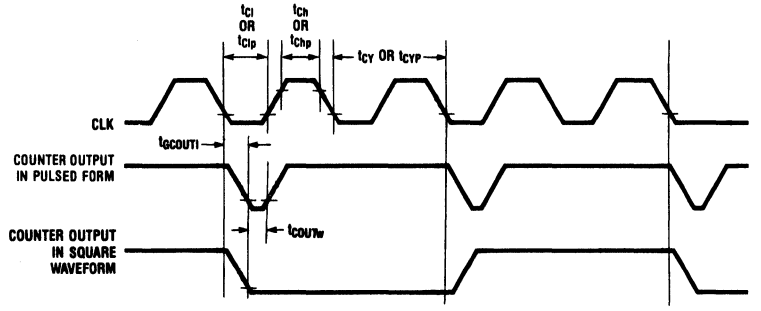
Note: Interrupts are sampled on the rising edge of CLK.

FIGURE 4–7. External Interrupt-Sampling-Clock to be Provided at Pin COUT/SCIN When in Test Mode



TL/EE/5117–22

FIGURE 4–8. Internal Interrupt-Sampling-Clock Provided at Pin COUT/SCIN



TL/EE/5117–23

FIGURE 4–9. Relationship Between Clock Input at Pin CLK and Counter Output Signals at Pins COUT/SCIN or G0/R0,...,G3/R6, in Both Pulsed Form and Square Waveform

National Semiconductor Corporation

NS16450/INS8250A/NS16C450/INS82C50A Asynchronous Communications Element

General Description

The NS16450 is an improved specification version of the INS8250A Asynchronous Communications Element (ACE). The improved specifications ensure compatibility with the NS32016 and other state-of-the-art CPUs. Functionally, the NS16450 is equivalent to the INS8250A. The ACE is fabricated using National Semiconductor's advanced scaled N-channel silicon-gate MOS process, XMOS.

The NS16C450 and INS82C50A are functionally equivalent to their XMOS counterparts, except that they are CMOS parts. (The CMOS parts will be available after June 1985.) It functions as a serial data input/output interface in a microcomputer system. The functional configuration of the ACE is programmed by the system software via a TRI-STATE® 8-bit bidirectional data bus; this includes the on-board baud rate generator.

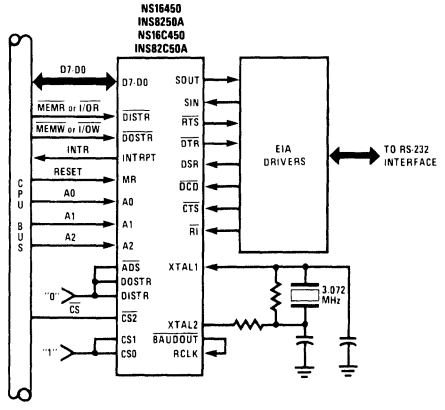
The ACE performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the ACE at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the ACE, as well as any error conditions (parity, overrun, framing, or break interrupt).

The ACE includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to $(2^{16}-1)$, and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. Also included in the ACE is a complete MODEM-control capability, and a processor-interrupt system that may be software tailored to the user's requirements to minimize the computing required to handle the communications link.

Features

- Easily interfaces to most popular microprocessors.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from serial data stream.
- Full double buffering eliminates need for precise synchronization.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator allows division of any input clock by 1 to $(2^{16}-1)$ and generates the internal 16 × clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no-parity bit generation and detection
  - 1-, 1½-, or 2-stop bit generation
  - Baud generation (DC to 56k baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE TTL drive capabilities for bidirectional data bus and control bus.
- Line break generation and detection.
- Internal diagnostic capabilities:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation.
- Full prioritized interrupt system controls.

Connection Diagram



TL/C/8401-1

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}	−0.5V to +7.0V
Power Dissipation	700 mW

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	NS16450 NS16C450 (Note 1)		INS8250A INS82C50A (Note 1)		Units
			Min	Max	Min	Max	
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2.0	V_{CC}	2.0	V_{CC}	V
V_{IL}	Input Low Voltage		−0.5	0.8	−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}	2.0	V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all *		0.4		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1.0 mA *	2.4		2.4		V
I_{CC}(AV)	Avg. Power Supply Current (V_{CC})	V_{CC} = 5.25V, T_A = 25°C No Loads on output SIN, DSR, RLSD, CTS, RI = 2.0V All other inputs = 0.8V		120		95	mA
I_{CC}(AV)	Avg. Power Supply Current (V_{CC}) CMOS Parts Only	V_{CC} = 5.25V, T_A = 25°C No Loads on output SIN, DSR, RLSD, CTS, RI = 2.0V All other inputs = 0.8V Baud Rate Generator is 4 MHz Baud Rate is 56k		10		10	mA
I_{IL}	Input Leakge	V_{CC} = 5.25V, V_{SS} = 0V All other pins floating. V_{IN} = 0V, 5.25V		±10		±10	µA
I_{CL}	Clock Leakage			±10		±10	µA
I_{OZ}	TRI-STATE Leakage	V_{CC} = 5.25V, V_{SS} = 0V V_{OUT} = 0V, 5.25V 1) Chip deselected 2) WRITE mode, chip selected		±20		±20	µA
V_{ILMR}	MR Schmitt V_{IL}			0.8		0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2.0		2.0		V

* Does not apply to XTAL2

Capacitance T_A = 25°C, V_{CC} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XTAL_2}	Clock Input Capacitance	f_c = 1 MHz Unmeasured pins returned to V_{SS}		15	20	pF
C_{XTAL_1}	Clock Output Capacitance			20	30	pF
C_{IN}	Input Capacitance			6	10	pF
C_{OUT}	Output Capacitance			10	20	pF

Note 1: All specifications for CMOS parts are preliminary. Inputs on the CMOS parts are TTL compatible; outputs on the CMOS parts drive to the rails.

AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Conditions	NS16450 NS16C450 (Note 1)		INS8250A INS82C50A (Note 1)		Units
			Min	Max	Min	Max	
t_{AW}	Address Strobe Width		60		90		ns
t_{AS}	Address Setup Time		60		90		ns
t_{AH}	Address Hold Time		0		0		ns
t_{CS}	Chip Select Setup Time		60		90		ns
t_{CH}	Chip Select Hold Time		0		0		ns
t_{DIW}	$\overline{DISTR}$/DISTR Strobe Width		125		175		ns
t_{RC}	Ready Cycle Delay		175		500		ns
RC	Ready Cycle = $t_{AR*} + t_{DIW} + t_{RC}$		360		755		ns
t_{DD}	$\overline{DISTR}$/DISTR to Driver Disable Delay	@100 pF loading***		60		75	ns
t_{DDD}	Delay from $\overline{DISTR}$/DISTR to Data	@100 pF loading		125		175	ns
t_{HZ}	$\overline{DISTR}$/DISTR to Floating Data Delay	@100 pF loading***	0	100	100		ns
t_{DOW}	$\overline{DOSTR}$/DOSTR Strobe Width		100		175		ns
t_{WC}	Write Cycle Delay		200		500		ns
WC	Write Cycle = $t_{AW} + t_{DOW} + t_{WC}$		360		755		ns
t_{DS}	Data Setup Time		40		90		ns
t_{DH}	Data Hold Time		40		60		ns
t_{CSC}*	Chip Select Output Delay from Select	@100 pF loading		100		125	ns
t_{RA}*	Address Hold Time from $\overline{DISTR}$/DISTR		20		20		ns
t_{RCS}*	Chip Select Hold Time from $\overline{DISTR}$/DISTR		20		20		ns
t_{AR}*	$\overline{DISTR}$/DISTR Delay from Address		60		80		ns
t_{CSR}*	$\overline{DISTR}$/DISTR Delay from Chip Select		50		80		ns
t_{WA}*	Address Hold Time from $\overline{DOSTR}$/DOSTR		20		20		ns
t_{WCS}*	Chip Select Hold Time from $\overline{DOSTR}$/DOSTR		20		20		ns
t_{AW}*	$\overline{DOSTR}$/DOSTR Delay from Address		60		80		ns
t_{CSW}*	$\overline{DOSTR}$/DOSTR Delay from Select		50		80		ns
t_{MRW}	Master Reset Pulse Width		5		10		μs
t_{XH}	Duration of Clock High Pulse	External Clock (3.1 MHz Max.)	140		140		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (3.1 MHz Max.)	140		140		ns
Baud Generator							
N	Baud Divisor		1	$2^{16}-1$	1	$2^{16}-1$	
t_{BLD}	Baud Output Negative Edge Delay	100 pF Load		125		250	ns
t_{BHD}	Baud Output Positive Edge Delay	100 pF Load		125		250	ns
t_{LW}	Baud Output Down Time	f_X = 2 MHz, ÷2, 100 pF Load	425		425		ns
t_{HW}	Baud Output Up Time	f_X = 3 MHz, ÷3, 100 pF Load	330		330		ns
Receiver							
t_{SCD}	Delay from RCLK to Sample Time			2		2	μs
t_{SINT}	Delay from Stop to Set Interrupt		1	1	1	1	RCLK** Cycles
t_{RINT}	Delay from $\overline{DISTR}$/DISTR (RD RBR/RDLSR) to Reset Interrupt	100 pF Load		1		1	μs

*Applicable only when $\overline{ADS}$ is tied low.

**RCLK is equal to t_{XH} and t_{XL}.

***Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

Note 1: All specifications for CMOS parts are preliminary.

4

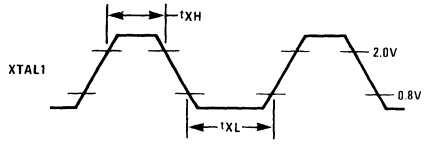
AC Electrical Characteristics (Continued)

Symbol	Parameter	Conditions	NS16450 NS16C450 (Note 1)		INS8250A INS82C50A (Note 1)		Units
			Min	Max	Min	Max	
Transmitter							
t_{HR}	Delay from $\overline{DOSTR}$/DOSTR (WR THR) to Reset Interrupt	100 pF Load		175		1000	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		8	24	8	24	RCLK Cycles
t_{SI}	Delay from Initial Write to Interrupt		16	32	16	32	RCLK Cycles
t_{STI}	Delay from Stop to Interrupt (THRE)		8	8	8	8	RCLK Cycles
t_{IR}	Delay from $\overline{DISTR}$/DISTR (RD IIR) to Reset Interrupt (THRE)	100 pF Load		250		1000	ns
Modem Control							
t_{MDO}	Delay from $\overline{DOSTR}$/DOSTR (WR MCR) to Output	100 pF Load		200		1000	ns
t_{SIM}	Delay to Set Interrupt from MODEM Input	100 pF Load				1000	ns
t_{RIM}	Delay to Reset Interrupt from $\overline{DISTR}$/DISTR (RD MSR)	100 pF Load		250		1000	ns

Note 1: All specifications for CMOS parts are preliminary.

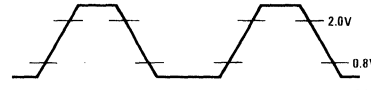
Timing Waveforms (All timings are referenced to valid 0 and valid 1)

External Clock Input (3.1 MHz Max.)



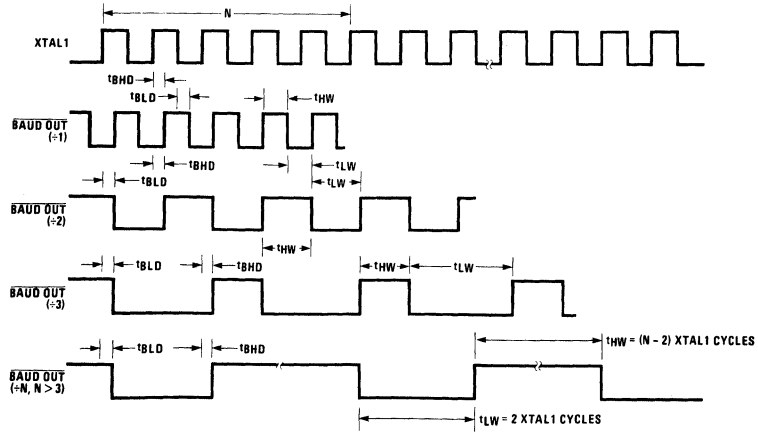
TL/C/8401–2

AC Test Points



TL/C/8401–3

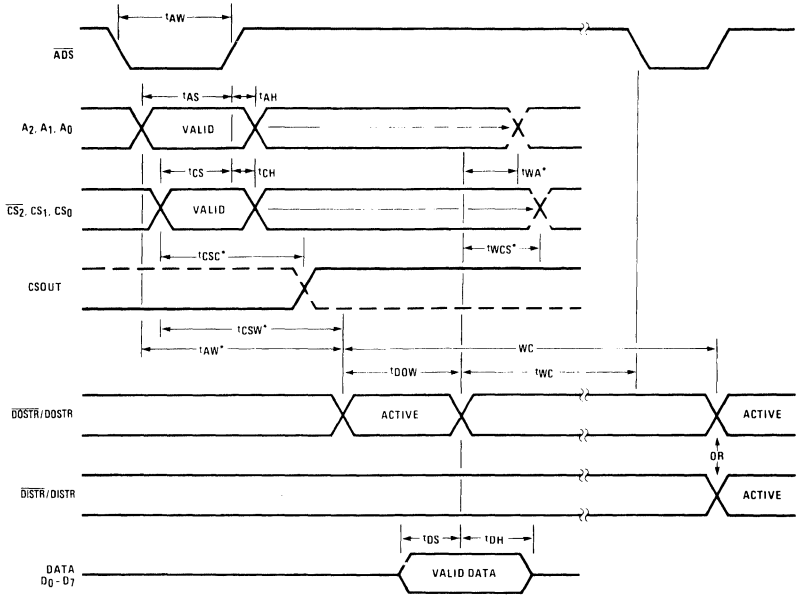
$\overline{BAUDOUT}$ Timing



TL/C/8401–4

Timing Waveforms (Continued)

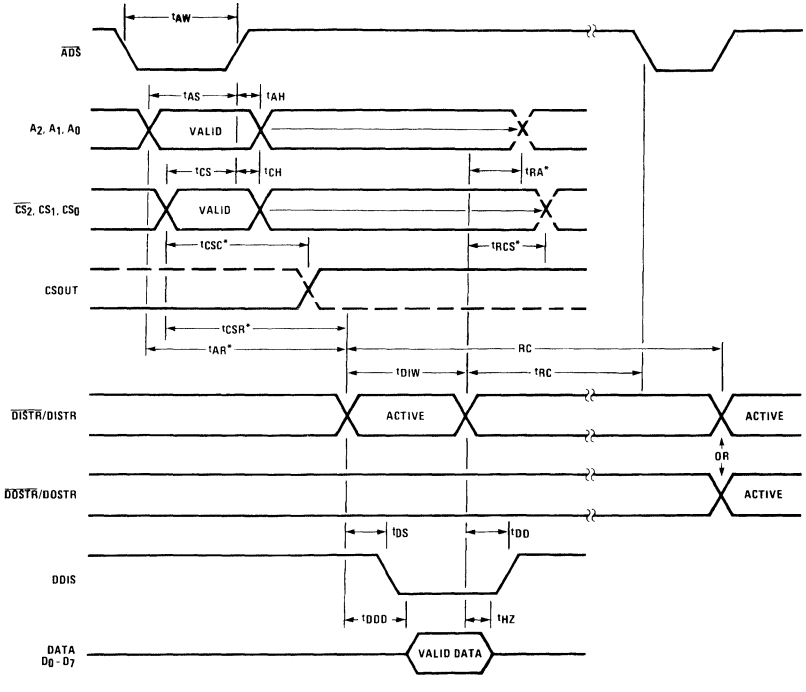
Write Cycle



TL/C/8401-5

*Applicable Only When $\overline{ADS}$ is Tied Low.

Read Cycle

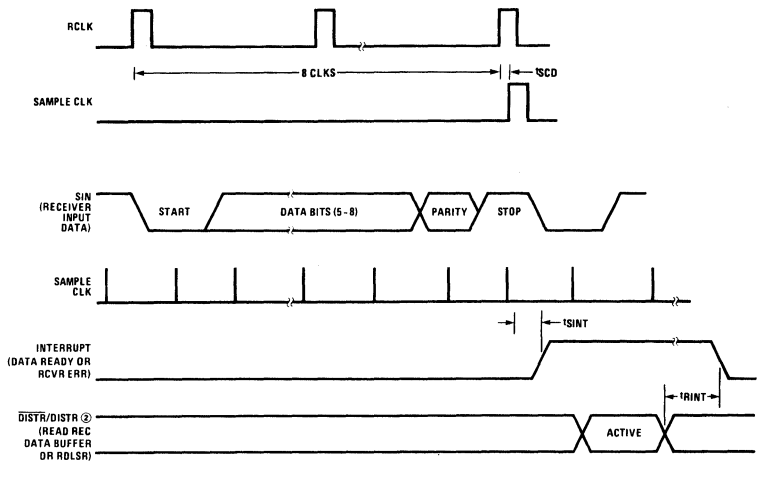


TL/C/8401-6

*Applicable Only When $\overline{ADS}$ is Tied Low.

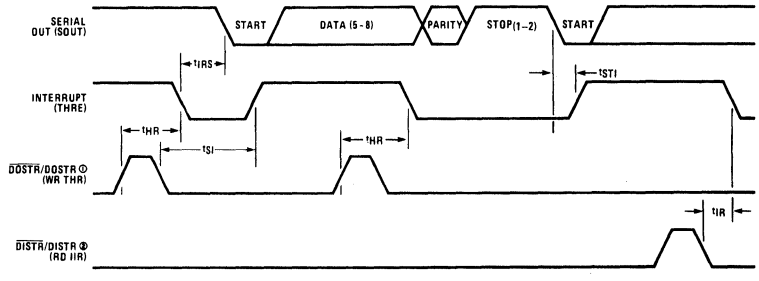
Timing Waveforms (Continued)

Receiver Timing



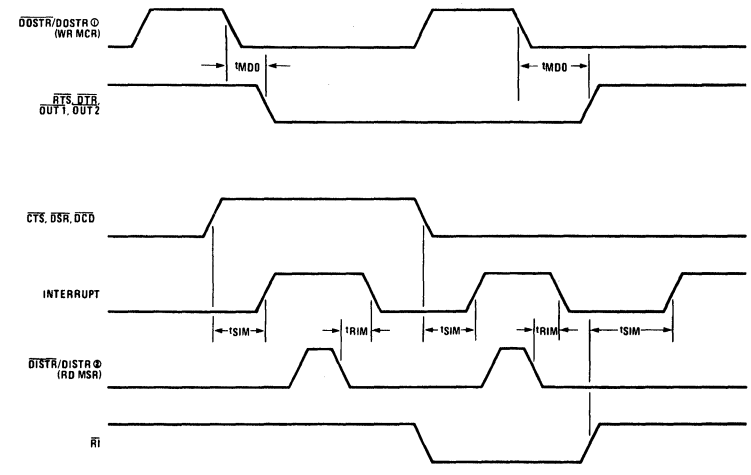
TL/C/8401-7

Transmitter Timing



TL/C/8401-8

MODEM Controls Timing

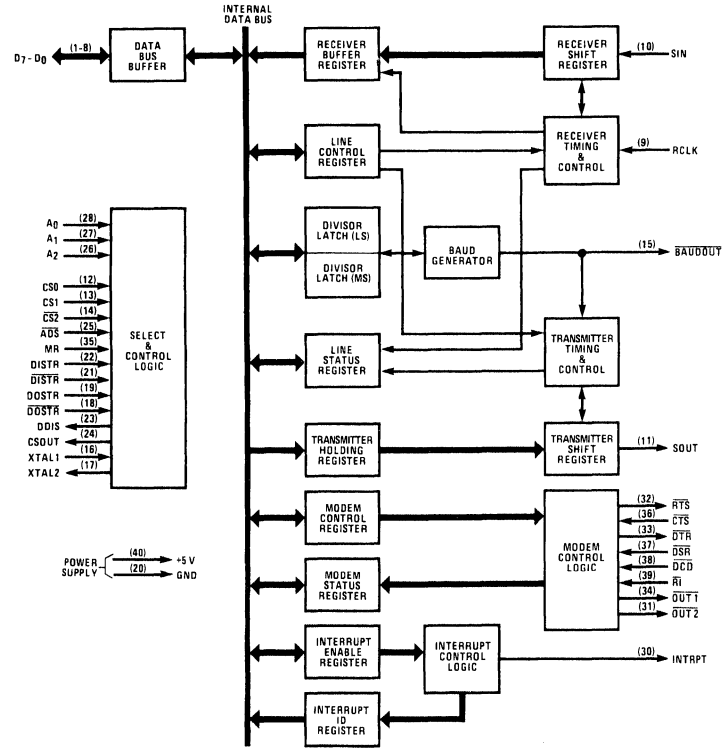


TL/C/8401-9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

Block Diagram



TL/C/8401-10

Note: Applicable pinout numbers are included within parenthesis.

Functional Pin Description

The following describes the function of all NS16450, NS16C450 and INS8250A, INS82C50A input and output pins. Some of these descriptions reference internal circuits.

Note: In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

INPUT SIGNALS

Chip Select (CS0, CS1, $\overline{CS2}$), Pins 12–14: When CS0 and CS1 are high and $\overline{CS2}$ is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) Address Strobe ($\overline{ADS}$) input. This enables communication between the ACE and the CPU.

Data Input Strobe (DISTR, $\overline{DISTR}$), Pins 22 and 21: When DISTR is high or $\overline{DISTR}$ is low while the chip is selected, it allows the CPU to read status information or data from a selected register of the ACE.

Note: Only an active DISTR or $\overline{DISTR}$ input is required to transfer data from the ACE during a read operation. Therefore, tie either the DISTR input permanently low or the $\overline{DISTR}$ input permanently high, if not used.

Data Output Strobe (DOSTR, $\overline{DOSTR}$), Pins 19 and 18: When DOSTR is high or $\overline{DOSTR}$ is low while the chip is selected, allows the CPU to write data or control words into a selected register of the ACE.

Note: Only an active DOSTR or $\overline{DOSTR}$ input is required to transfer data to the ACE during a write operation. Therefore, tie either the DOSTR input permanently low or the $\overline{DOSTR}$ input permanently high, if not used.

Address Strobe ($\overline{ADS}$), Pin 25: When low, provides latching for the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, $\overline{CS2}$) signals.

Note: An active $\overline{ADS}$ input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the $\overline{ADS}$ input permanently low.

Register Select (A0, A1, A2), Pins 26–28: These three inputs are used during a read or write operation to select an ACE register to read from or write into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain ACE registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

Functional Pin Description (Continued)

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Master Reset (MR), Pin 35: This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis. When high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the ACE. Also, the state of various output signals (SOUT, INTRPT, $\overline{OUT\ 1}$, $\overline{OUT\ 2}$, $\overline{RTS}$, $\overline{DTR}$) are affected by an active MR input. (Refer to Table I.)

Receiver Clock (RCLK), Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

Clear to Send ($\overline{CTS}$), Pin 36: The $\overline{CTS}$ signal is a MODEM control function input whose conditions can be tested by the CPU by reading bit 4 (CTS) of the MODEM Status Register. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{CTS}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Set Ready ($\overline{DSR}$), Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link and transfer data with the ACE. The $\overline{DSR}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 5 (DSR) of the MODEM Status Register. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Carrier Detect ($\overline{DCD}$), Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{DCD}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 7 (DCD) of the MODEM Status Register. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{DCD}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{DCD}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Ring Indicator ($\overline{RI}$), Pin 39: When low, indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 6 (RI) of the MODEM Status Register. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI bit of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Register is enabled.

V_{CC}, Pin 40: +5V supply.

V_{SS}, Pin 20: Ground (0V) reference.

OUTPUT SIGNALS

Data Terminal Ready ($\overline{DTR}$), Pin 33: When low, informs the MODEM or data set that the ACE is ready to communicate. The $\overline{DTR}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. The $\overline{DTR}$ signal is set high upon a Master Reset operation. The $\overline{DTR}$ signal is forced to its inactive state (high) during loop mode operation.

Request to Send ($\overline{RTS}$), Pin 32: When low, informs the MODEM or data set that the ACE is ready to transmit data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. The $\overline{RTS}$ signal is set high upon a Master Reset operation. The $\overline{RTS}$ signal is forced to its inactive state (high) during loop mode operation.

Output 1 ($\overline{OUT\ 1}$), Pin 34: User-designated output that can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. The $\overline{OUT\ 1}$ signal is set high upon a Master Reset Operation. The $\overline{OUT\ 1}$ signal is forced to its inactive state (high) during loop mode operation. In the XMOS parts this will achieve TTL levels.

Output 2 ($\overline{OUT\ 2}$), Pin 31: User-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. The $\overline{OUT\ 2}$ signal is set high upon a Master Reset Operation. The $\overline{OUT\ 2}$ signal is forced to its inactive state (high) during loop mode operation. In the XMOS parts this will achieve TTL levels.

Chip Select Out (CSOUT), Pin 24: When high, indicates that the chip has been selected by active, CS0, CS1, and $\overline{CS2}$ inputs. No data transfer can be initiated until the CSOUT signal is a logic 1. CSOUT goes low when chip is deselected.

Driver Disable (DDIS), Pin 23: Goes low whenever the CPU is reading data from the ACE. A high-level DDIS output can be used to disable an external transceiver (if used between the CPU and ACE on the D_7–D_0 Data Bus) at all times, except when the CPU is reading data.

Baud Out ($\overline{BAUDOUT}$), Pin 15: 16 × clock signal for the transmitter section of the ACE. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The $\overline{BAUDOUT}$ may also be used for the receiver section by tying this output to the RCLK input of the chip.

Functional Pin Description (Continued)

Interrupt (INTRPT), Pin 30: Goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTRPT signal is reset low upon the appropriate interrupt service or a Master Reset operation.

Serial Output (SOUT), Pin 11: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

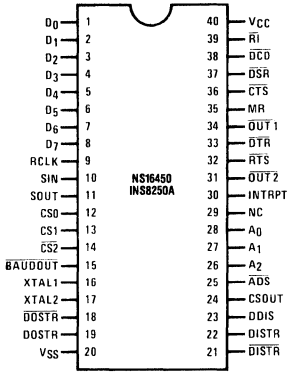
INPUT/OUTPUT SIGNALS

Data (D_7–D_0) Bus, Pins 1–8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the ACE and the CPU. Data, control words, and status information are transferred via the D_7–D_0 Data Bus.

External Clock Input/Output (XTAL 1, XTAL 2) Pins 16 and 17: These two pins connect the main timing reference (crystal or signal clock) to the ACE.

Connection Diagrams

Dual-In-Line Package

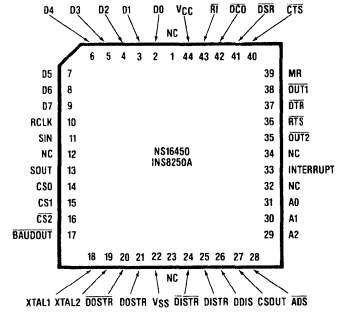


TL/C/8401–11

Top View

Order Number NS16450N, NS-16450N, INS8250AN, NS16C450N or INS82C50AN
See NS Package N40A

PCC Package



TL/C/8401–18

Top View

Order Number NS16450V, NS-16450V, INS8250A, NS16C450V or INS82C50AV
See NS Package V44A

TABLE I. ACE Reset Functions

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	All Bits Low (0–3 forced and 4–7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High, Bits 1 and 2 Low Bits 3–7 are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	All Bits Low, Except Bits 5 and 6 are High
MODEM Status Register	Master Reset	Bits 0–3 Low Bits 4–7—Input Signal
SOUT	Master Reset	High
INTRPT (RCVR Errs)	Read LSR/MR	Low
INTRPT (RCVR Data Ready)	Read RBR/MR	Low
INTRPT (THRE)	Read IIR/Write THR/MR	Low
INTRPT (Modem Status Changes)	Read MSR/MR	Low
$\overline{OUT\ 2}$	Master Reset	High
$\overline{RTS}$	Master Reset	High
$\overline{DTR}$	Master Reset	High
$\overline{OUT\ 1}$	Master Reset	High

Accessible Registers

The system programmer may access or control any of the ACE registers summarized in Table II via the CPU. These registers are used to control ACE operations and to transmit and receive data.

LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange via the Line Control Register. In addition to controlling the format, the programmer may retreive the contents of the Line Control Register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the Line Control Register are indicated in Table II and are described below.

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits in each transmitted character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a

Table II. Summary of Accessible Registers

Bit No.	Register Address										
	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Latch (MS)
	RBR	THR	IER	IIR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0*	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

*Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Accessible Registers (Continued)

half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1.

Bit 6: This bit is the Break Control bit. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s, pad character, in response to THRE.
2. Set break after the next THRE.
3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

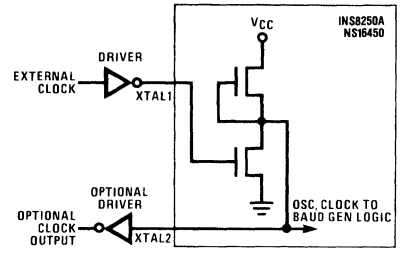
During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

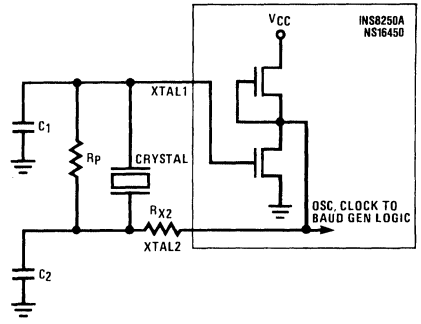
Table III. Baud Rates Using 1.8432 MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

Typical Clock Circuits



TL/C/8401–12



TL/C/8401–13

Typical Crystal Oscillator Network

CRYSTAL	R_P	R_{X2}	C_1	C_2
3.1 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF
1.8 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF

Table IV. Baud Rates Using 3.072 MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

4

Accessible Registers (Continued)

PROGRAMMABLE BAUD GENERATOR

The ACE contains a programmable Baud Generator that is capable of taking any clock input (DC to 3.1 MHz) and dividing it by any divisor from 1 to $2^{16}-1$). The output frequency of the Baud Generator is 16 × the Baud [divisor # = (frequency input) ÷ (baud rate × 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure desired operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded. This prevents long counts on initial load.

Tables III and IV illustrate the use of the Baud Generator with crystal frequencies of 1.8432 MHz and 3.072 MHz respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen.

Note: The maximum operating frequency of the Baud Generator is 3.1 MHz. However, when using divisors of 3 and below, the maximum frequency is equal to the divisor in MHz. For example, if the divisor is 1, then the maximum frequency is 1 MHz. In no case should the data rate be greater than 56k Baud.

LINE STATUS REGISTER

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the Line Status Register are indicated in Table 2 and are described below.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register. Bit 0 is reset to a logic 0 by reading the data in the Receiver Buffer Register.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is reset whenever the CPU reads the contents of the Line Status Register.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status indicator.

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status indicator.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected.

Bit 5: The bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the ACE is ready to accept a new character for transmission. In addition, this bit causes the ACE to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to a logic 0 whenever either the THR or TSR contains a data character.

Bit 7: This bit is permanently set to logic 0.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is used for factory testing.

TABLE V. Interrupt Control Functions

Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 2	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

Accessible Registers (Continued)

INTERRUPT IDENTIFICATION REGISTER

The ACE has an on-chip interrupt capability that allows for flexibility in interfacing popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the ACE prioritizes interrupts into four levels. The four levels of interrupt conditions are as follows: Receiver Line Status (priority 1); Received Data Ready (priority 2); Transmitter Holding Register Empty (priority 3); and MODEM Status (priority 4).

Information indicating that a prioritized interrupt is pending and the type of that interrupt are stored in the Interrupt Identification Register (IIR). When addressed during chip-select time, the IIR freezes the highest priority interrupt pending and no other interrupts change the IIR, even though they are recorded, until the particular interrupt is serviced by the CPU. The contents of the IIR are indicated in Table II and are described below.

Bit 0: This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending and polling (if used) continues.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table V.

Bits 3 through 7: These five bits of the IIR are always logic 0.

INTERRUPT ENABLE REGISTER

This 8-bit register enables the four types of interrupts of the ACE to separately activate the chip interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register. Similarly, by setting the appropriate bits of this register to a logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the Interrupt Identification Register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. The contents of the Interrupt Enable Register are indicated in Table II and are described below.

Bit 0: This bit enables the Received Data Available Interrupt when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

MODEM CONTROL REGISTER

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table II and are described below.

Bit 0: This bit controls the Data Terminal Ready ($\overline{DTR}$) output. When bit 0 is set to a logic 1, the $\overline{DTR}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{DTR}$ output is forced to a logic 1.

Note: The $\overline{DTR}$ output of the ACE may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{RTS}$) output. Bit 1 affects the $\overline{RTS}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{OUT\ 1}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{OUT\ 1}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{OUT\ 2}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{OUT\ 2}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the ACE. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{CTS}$, $\overline{DSR}$, $\overline{DCD}$, and $\overline{RI}$) are disconnected; and the four MODEM Control outputs ($\overline{DTR}$, $\overline{RTS}$, $\overline{OUT\ 1}$, and $\overline{OUT\ 2}$) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and received-data paths of the ACE.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

MODEM STATUS REGISTER

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

Accessible Registers (Continued)

The contents of the MODEM Status Register are indicated in Table II and are described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{DSR}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{RI}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{DCD}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{CTS}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{DSR}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent of DTR in the MCR.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

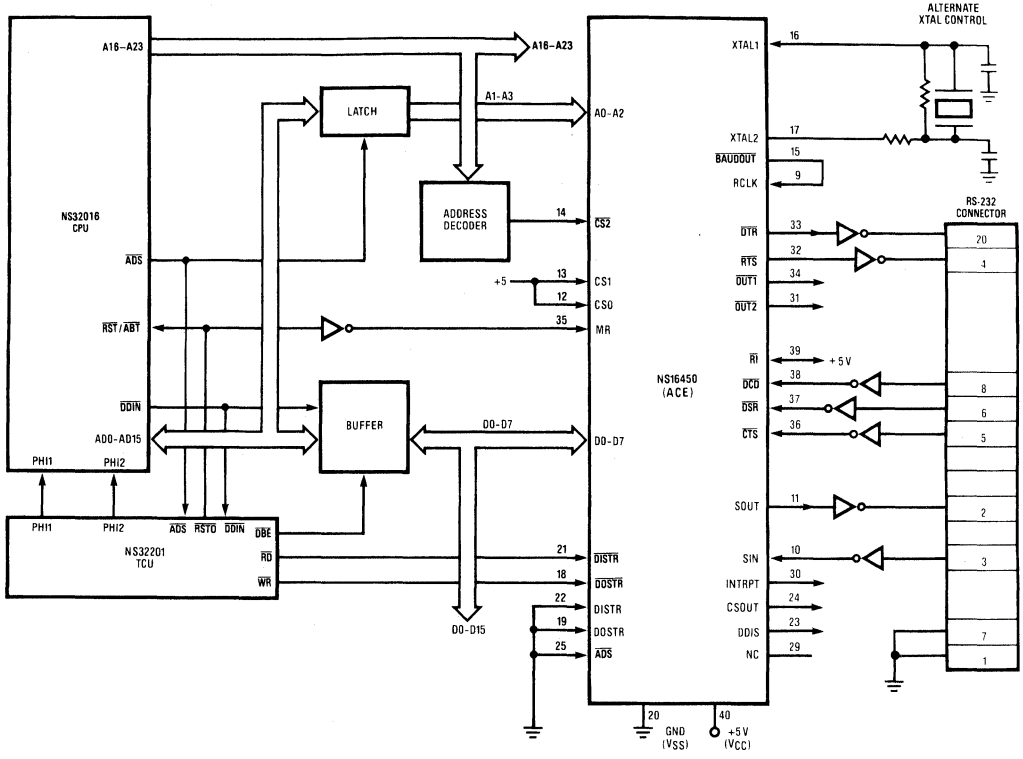
Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{DCD}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 of the MCR.

SCRATCHPAD REGISTER

This 8-bit Read/Write Register does not control the ACE in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

Typical Applications

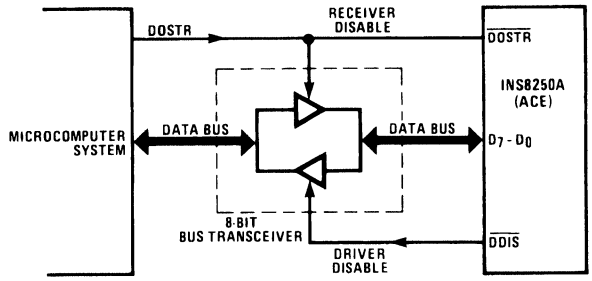
This shows the basic connections of an NS16450 to an NS32016 CPU



TL/C/8401–14

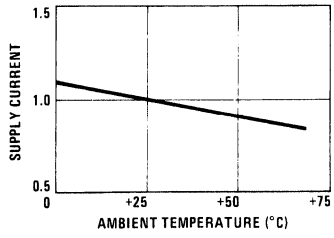
Typical Applications (Continued)

Typical Interface for a High-Capacity Data Bus



TL/C/8401–16

Typical Supply Current vs. Temperature, Normalized



TL/C/8401–17

Ordering Information

Order Number	Description
Plastic Dip Package	
NS16450N or NS-16450N	high speed part
INS8250AN	V_{CC} = 5V ±5%
NS16C450N*	CMOS high speed part
INS82C50AN*	CMOS V_{CC} = 5V ±5%
Plastic Chip Carrier Package	
NS16450V or NS-16450V	high speed part
INS8250A	V_{CC} = 5V ± 5%
NS16C450V*	CMOS high speed part
INS82C50AV*	CMOS V_{CC} = 5V ±5%

*The CMOS parts will be available after 6/85.

4

National Semiconductor Corporation

PRELIMINARY

NS16550 Asynchronous Communications Element with FIFOs

General Description

The NS16550 is an improved version of the NS16450 Asynchronous Communications Element (ACE). The improved specifications ensure compatibility with the NS32016 and other state-of-the-art CPUs. Functionally identical to the NS16450 on powerup (CHARACTER mode)* the NS16550 can be put into an alternate mode (FIFO mode) to relieve the CPU of excessive overhead when the data rate is high.

In this mode internal FIFOs are activated allowing 16 bytes (plus 3 bits per byte of error data in the RCVR FIFO) to be stored in both receive and transmit modes. All the logic is on chip to minimize system overhead and maximize system efficiency. Two pin functions have been added to allow signalling of DMA transfers.

The ACE performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the ACE at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the ACE, as well as any error conditions (parity, overrun, framing, or break interrupt).

The ACE includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to ($2^{16}-1$), and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. The ACE has complete MODEM-control capability, and a processor-interrupt system that may be software tailored to the user's requirements, minimizing the computing required to handle the communications link.

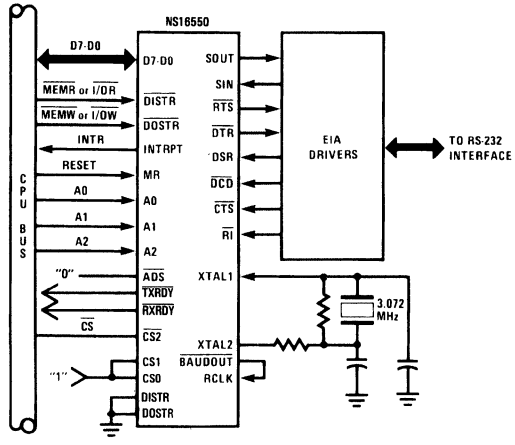
The ACE is fabricated using National Semiconductor's advanced scaled N-channel silicon-gate MOS process, XMOS.

*Can also be reset to (Character) mode under software control.

Features

- Capable of running all existing 16450 software.
- Pin for pin compatible with the existing 16450 except for CSOUT (24) and NC (29). The former CSOUT and NC pins will be TXRDY and RXRDY, respectively.
- After reset, all registers are identical to the 16450 register set.
- In the FIFO mode transmitter and receiver are each buffered with 16 byte FIFO's to reduce the number of interrrupts presented to the CPU.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data.
- Full double buffering in the CHARACTER mode eliminates need for precise synchronization.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator divides any input clock by 1 to ($2^{16}-1$) and generates the 16 × clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no-parity bit generation and detection
  - 1-, 1½-, or 2-stop bit generation
  - Baud generation (DC to 256k baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE® TTL drive for the data and control buses.
- Line break generation and detection.
- Internal diagnostic capabilities:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation.
- Full prioritized interrupt system controls.

Basic Configuration



TL/C/8652-1

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}	−0.5V to +7.0V
Power Dissipation	1W

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2.0	V_{CC}	V
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{CC}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all (Note 1)		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1.0 mA (Note 1)	2.4		V
I_{CC}(AV)	Avg. Power Supply Current (V_{CC})	V_{CC} = 5.25V No Loads on output SIN, DSR, RLSD, CTS, RI = 2.0V All other inputs = 0.8V		160 (Note 2) 140 (Note 3)	mA mA
I_{IL}	Input Leakage	V_{CC} = 5.25V, V_{SS} = 0V All other pins floating. V_{IN} = 0V, 5.25V		±10	μA
I_{CL}	Clock Leakage			±10	μA
I_{OZ}	TRI-STATE Leakage	V_{CC} = 5.25V, V_{SS} = 0V V_{OUT} = 0V, 5.25V 1) Chip deselected 2) WRITE mode, chip selected		±20	μA
V_{ILMR}	MR Schmitt V_{IL}			0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2.0		V

Note 1: Does not apply to XTAL2

Note 2: T_A = 25°C

Note 3: T_A = 70°C

Capacitance T_A = 25°C, V_{CC} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XTAL_2}	Clock Input Capacitance	f_c = 1 MHz Unmeasured pins returned to V_{SS}		15	20	pF
C_{XTAL_1}	Clock Output Capacitance			20	30	pF
C_{IN}	Input Capacitance			6	10	pF
C_{OUT}	Output Capacitance			10	20	pF

4

AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$

Symbol	Parameter	Conditions	Min	Max	Units
t_{AH}	Address Hold Time		0		ns
t_{AR}	$\overline{DISTR}$/DISTR Delay from Address	(Note 1)	30		ns
t_{AS}	Address Setup Time		60		ns
t_{AW}	Address Strobe Width		60		ns
t_{CH}	Chip Select Hold Time		0		ns
t_{CS}	Chip Select Setup Time		60		ns
t_{CSR}	$\overline{DISTR}$/DISTR Delay from Chip Select	(Note 1)	50		ns
t_{CSW}	$\overline{DOSTR}$/DOSTR Delay from Select	(Note 1)	50		ns
t_{DD}	$\overline{DISTR}$/DISTR to Driver Enable/Disable	@100 pF loading (Note 3)		60	ns
t_{DDA}	$\overline{DOSTR}$/DOSTR Delay from Address	(Note 1)	30		ns
t_{DDD}	Delay from $\overline{DISTR}$/DISTR to Data	@100 pF loading		125	ns
t_{DH}	Data Hold Time		30		ns
t_{DIW}	$\overline{DISTR}$/DISTR Strobe Width		125		ns
t_{DOW}	$\overline{DOSTR}$/DOSTR Strobe Width		100		ns
t_{DS}	Data Setup Time		30		ns
t_{HZ}	$\overline{DISTR}$/DISTR to Floating Data Delay	@100 pF loading (Note 3)	0	100	ns
t_{MRW}	Master Reset Pulse Width		5		μs
t_{RA}	Address Hold Time from $\overline{DISTR}$/DISTR	(Note 1)	20		ns
t_{RC}	Read Cycle Delay		125		ns
t_{RCS}	Chip Select Hold Time from $\overline{DISTR}$/DISTR	(Note 1)	20		ns
t_{WA}	Address Hold Time from $\overline{DOSTR}$/DOSTR	(Note 1)	20		ns
t_{WC}	Write Cycle Delay		150		ns
t_{WCS}	Chip Select Hold Time from $\overline{DOSTR}$/DOSTR	(Note 1)	20		ns
t_{XH}	Duration of Clock High Pulse	External Clock (8.0 MHz Max.)	55		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (8.0 MHz Max.)	55		ns
RC	Read Cycle = $t_{AR} + t_{DIW} + t_{RC}$		280		ns
WC	Write Cycle = $t_{DDA} + t_{DOW} + t_{WC}$		280		ns
Baud Generator					
N	Baud Divisor		1	$2^{16}-1$	
t_{BHD}	Baud Output Positive Edge Delay	100 pF Load		175	ns
t_{BLD}	Baud Output Negative Edge Delay	100 pF Load		175	ns
t_{HW}	Baud Output Up Time	$f_X = 8.0$ MHz, ÷2, 100 pF Load	75		ns
t_{LW}	Baud Output Down Time	$f_X = 8.0$ MHz, ÷2, 100 pF Load	100		ns
Receiver					
t_{RINT}	Delay from $\overline{DISTR}$/DISTR (RD RBR/RDLSR) to Reset Interrupt	100 pF Load		1	μs
t_{SCD}	Delay from RCLK to Sample Time			2	μs
t_{SINT}	Delay from Stop to Set Interrupt	(Note 2)	1	1	RCLK Cycles

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 2: In the FIFO mode (FCR0 = 1) the trigger level and timeout interrupts, the receiver data available indication, the active RXRDY indication and the overrun error indication will be delayed 9 RCLKs. Status indicators (PE, FE, BI) will be delayed 9 RCLKs after the first byte has been received. For subsequently received bytes these indicators will be updated immediately after RDRBR goes inactive.

Note 3: Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

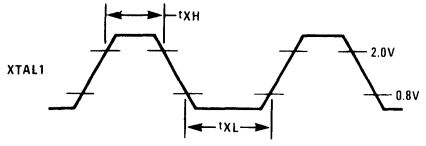
AC Electrical Characteristics (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
Transmitter					
t_{HR}	Delay from $\overline{DOSTR}$/DOSTR (WR THR) to Reset Interrupt	100 pF Load		175	ns
t_{IR}	Delay from $\overline{DISTR}$/DISTR (RD IIR) to Reset Interrupt (THRE)	100 pF Load		250	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		8	24	RCLK Cycles
t_{SI}	Delay from Initial Write to Interrupt	(Note 1)	16	32	RCLK Cycles
t_{STI}	Delay from Stop to Interrupt (THRE)	(Note 1)	8	8	RCLK Cycles
t_{SXA}	Delay from Start to TXRDY active	100 pF Load		8	RLCK Cycles
t_{WXI}	Delay from Write to TXRDY inactive	100 pF Load		195	ns
Modem Control					
t_{MDO}	Delay from $\overline{DOSTR}$/DOSTR (WR MCR) to Output	100 pF Load		200	ns
t_{RIM}	Delay to Reset Interrupt from $\overline{DISTR}$/DISTR (RD MSR)	100 pF Load		250	ns
t_{SIM}	Delay to Set Interrupt from MODEM Input	100 pF Load		250	ns

Note 1: This delay will be lengthened by 1 character time, minus the last stop bit time if the transmitter interrupt delay circuit is active. (See FIFO Interrupt Mode Operation).

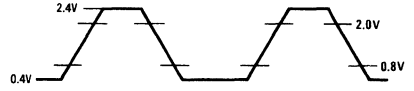
Timing Waveforms (All timings are referenced to valid 0 and valid 1)

External Clock Input (8.0 MHz Max.)



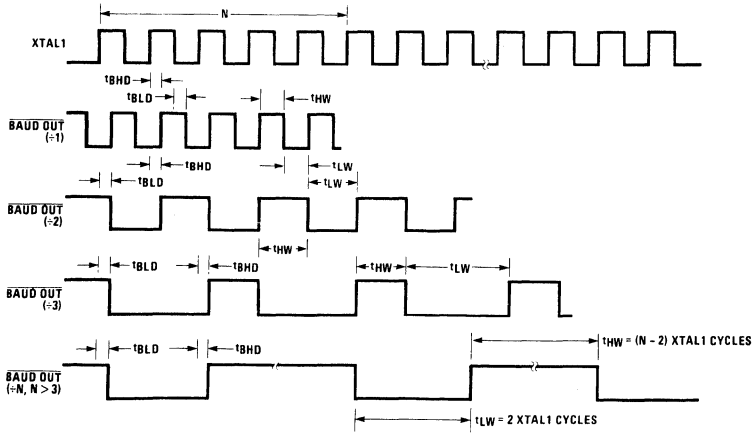
TL/C/8652-2

AC Test Points



TL/C/8652-3

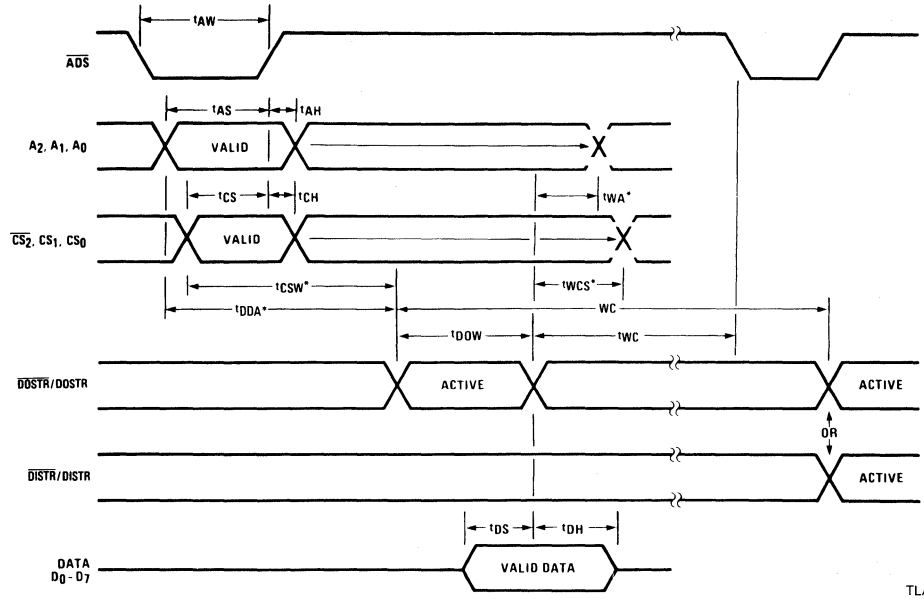
$\overline{BAUDOUT}$ Timing



TL/C/8652-4

Timing Waveforms (Continued)

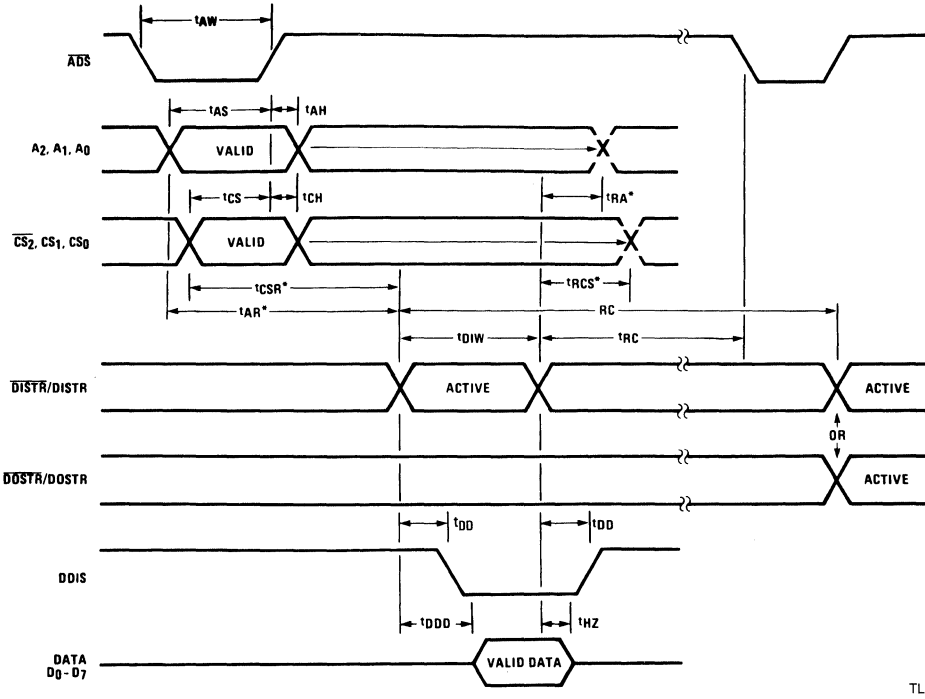
Write Cycle



TL/C/8652–5

*Applicable Only When $\overline{ADS}$ is Tied Low.

Read Cycle

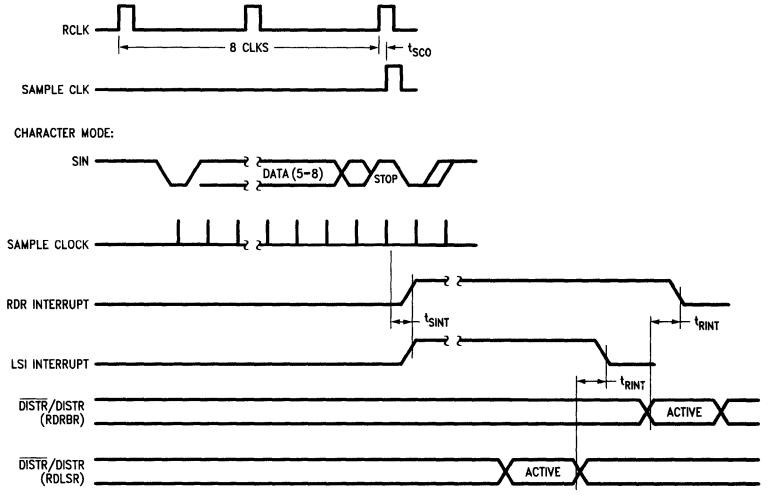


TL/C/8652–6

*Applicable Only When $\overline{ADS}$ is Tied Low.

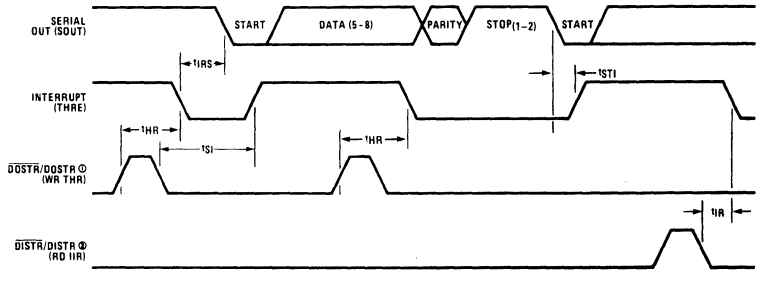
Timing Waveforms (Continued)

Receiver Timing



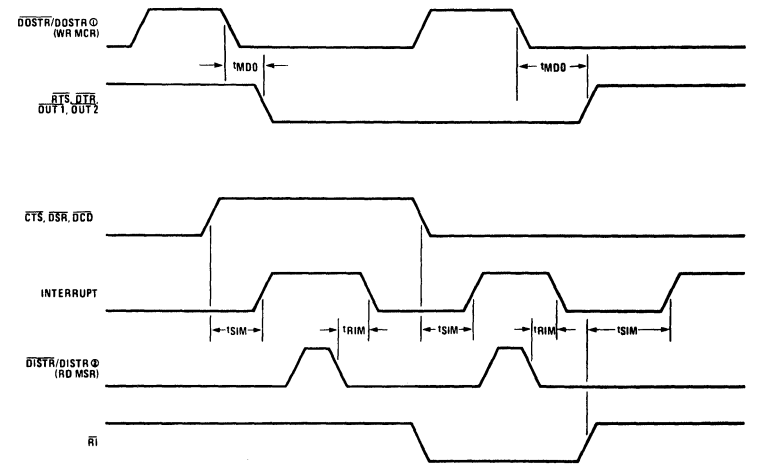
TL/C/8652-7

Transmitter Timing



TL/C/8652-8

MODEM Controls Timing



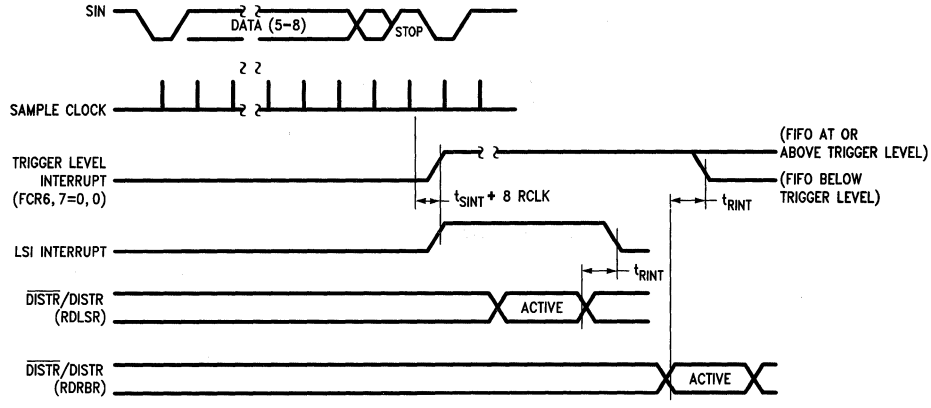
TL/C/8652-9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

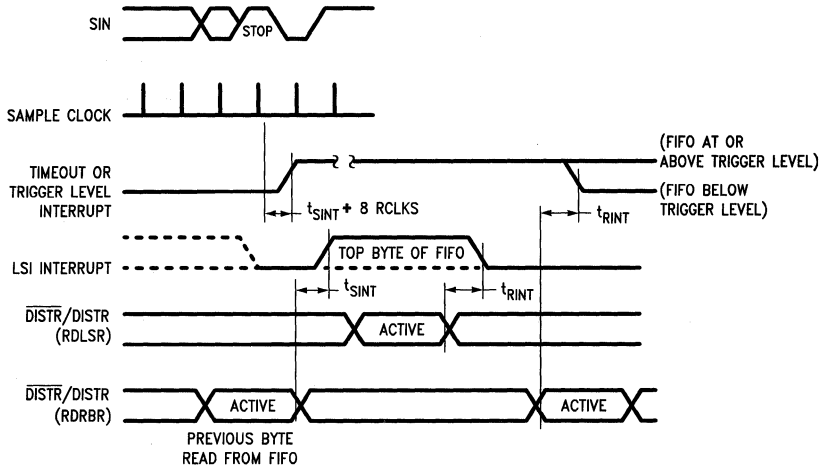
Timing Waveforms (Continued)

RCVR FIFO First Byte (This Sets RDR)



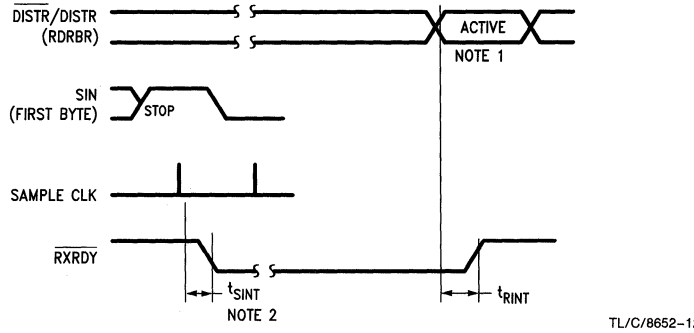
TL/C/8652–10

RCVR FIFO Bytes Other Than the First Byte (RDR Is Already Set)



TL/C/8652–11

Receiver Ready (Pin 29) FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)



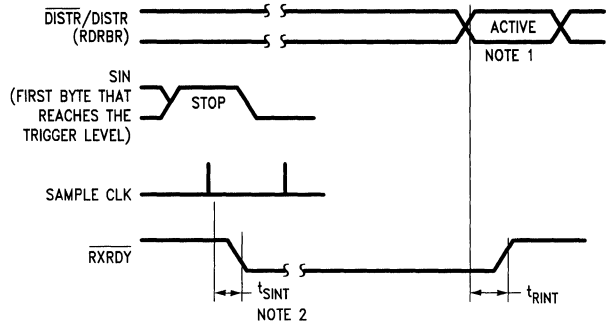
TL/C/8652–12

Note 1: This is the reading of the last byte in the FIFO.

Note 2: If FCR0 = 1, t_{SINT} = 9 RCLKs.

Timing Waveforms (Continued)

Receiver Ready (Pin 29) FCR0 = 1 and FCR3 = 1 (Mode 1)

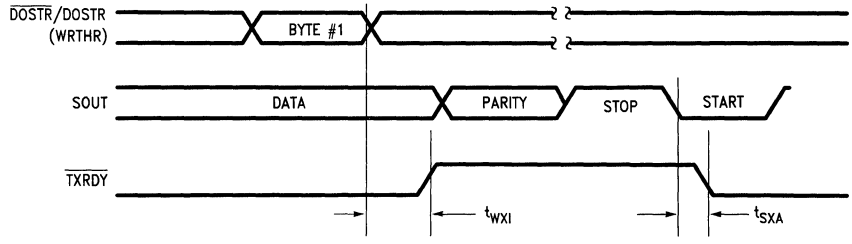


TL/C/8652–13

Note 1: This is the reading of the last byte in the FIFO.

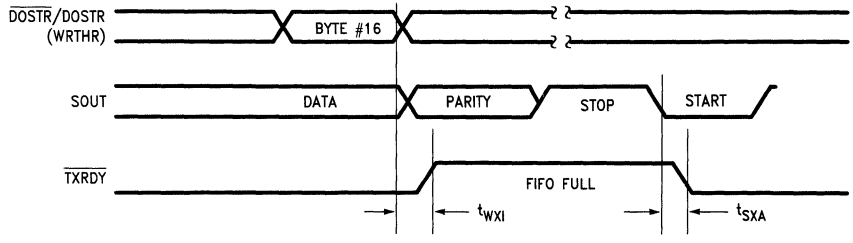
Note 2: If FCR0 = 1, t_{SINT} = 9 RCLKs.

Transmitter Ready (Pin 24) FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)



TL/C/8652–14

Transmitter Ready (Pin 24) FCR0 = 1 and FCR3 = 1 (Mode 1)



TL/C/8652–15

Functional Pin Description

The following describes the function of all NS16550 input and output pins. Some of these descriptions reference internal circuits.

Note: In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

INPUT SIGNALS

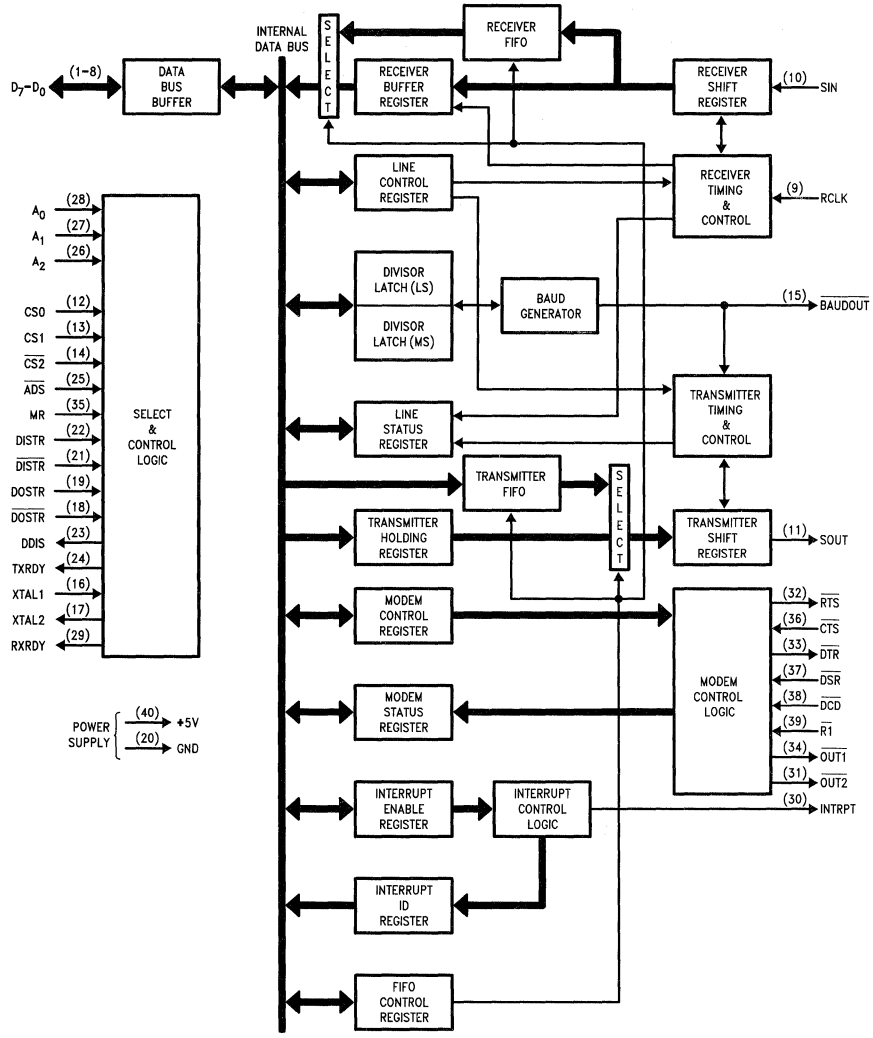
Chip Select (CS0, CS1, $\overline{CS2}$), Pins 12–14: When CS0 and CS1 are high and $\overline{CS2}$ is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) Address Strobe ($\overline{ADS}$) input. This enables communication between the ACE and the CPU.

Data Input Strobe (DISTR, $\overline{DISTR}$), Pins 22 and 21: When DISTR is high or $\overline{DISTR}$ is low while the chip is selected, it allows the CPU to read status information or data from a selected register of the ACE.

Note: Only an active DISTR or $\overline{DISTR}$ input is required to transfer data from the ACE during a read operation. Therefore, tie either the DISTR input permanently low or the $\overline{DISTR}$ input permanently high, if not used.

Data Output Strobe (DOSTR, $\overline{DOSTR}$), Pins 19 and 18: When DOSTR is high or $\overline{DOSTR}$ is low while the chip is selected, allows the CPU to write data or control words into a selected register of the ACE.

Block Diagram



TL/C/8652–16

Note: Applicable pinout numbers are included within parenthesis.

Functional Pin Description (Continued)

Note: Only an active DOSTR or $\overline{\text{DOSTR}}$ input is required to transfer data to the ACE during a write operation. Therefore, tie either the DOSTR input permanently low or the $\overline{\text{DOSTR}}$ input permanently high, if not used.

Address Strobe ($\overline{\text{ADS}}$), Pin 25: When low, provides latching for the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, $\overline{\text{CS2}}$) signals.

Note: An active $\overline{\text{ADS}}$ input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the $\overline{\text{ADS}}$ input permanently low.

Register Select (A0, A1, A2), Pins 26–28: These three inputs are used during a read or write operation to select an ACE register to read from or write into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain ACE registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

Master Reset (MR), Pin 35: This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis. When high, it clears all the registers and FIFOs (except the Receiver Buffer, Transmitter Holding, Scratch Pad Register and Divisor Latches), and the control logic of the ACE. Also, the state of various output signals (SOUT, INTRPT, $\overline{\text{OUT 1}}$, $\overline{\text{OUT 2}}$, $\overline{\text{RTS}}$, $\overline{\text{DTR}}$) are affected by an active MR input. (Refer to Table I.)

Functional Pin Description (Continued)

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read)
X	0	1	0	FIFO Control (write)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Receiver Clock (RCLK), Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

Clear to Send ($\overline{CTS}$), Pin 36: The $\overline{CTS}$ signal is a MODEM control function input whose conditions can be tested by the CPU by reading bit 4 (CTS) of the MODEM Status Register. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{CTS}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Set Ready ($\overline{DSR}$), Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link and transfer data with the ACE. The $\overline{DSR}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 5 (DSR) of the MODEM Status Register. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Carrier Detect ($\overline{DCD}$), Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{DCD}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 7 (DCD) of the MODEM Status Register. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{DCD}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{DCD}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Ring Indicator ($\overline{RI}$), Pin 39: When low, indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 6 (RI) of the MODEM Status Register. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI bit of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Register is enabled.

V_{CC}, Pin 40: +5V supply.

V_{SS}, Pin 20: Ground (0V) reference.

OUTPUT SIGNALS

Data Terminal Ready ($\overline{DTR}$), Pin 33: When low, informs the MODEM or data set that the ACE is ready to communicate. The $\overline{DTR}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. The $\overline{DTR}$ signal is set high upon a Master Reset operation. The $\overline{DTR}$ signal is forced to its inactive state (high) during loop mode operation.

Request to Send ($\overline{RTS}$), Pin 32: When low, informs the MODEM or data set that the ACE is ready to transmit data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. The $\overline{RTS}$ signal is set high upon a Master Reset operation. The $\overline{RTS}$ signal is forced to its inactive state (high) during loop mode operation.

Output 1 ($\overline{OUT\ 1}$), Pin 34: User-designated output that can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. The $\overline{OUT\ 1}$ signal is set high upon a Master Reset Operation. The $\overline{OUT\ 1}$ signal is forced to its inactive state (high) during loop mode operation.

Output 2 ($\overline{OUT\ 2}$), Pin 31: User-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. The $\overline{OUT\ 2}$ signal is set high upon a Master Reset Operation. The $\overline{OUT\ 2}$ signal is forced to its inactive state (high) during loop mode operation.

TXRDY, RXRDY, Pins 24, 29: Transmitter and Receiver DMA signalling is available through two pins (24 and 29). When operating in the FIFO mode, one of two types of DMA signalling per pin can be selected via FCR3. When operating as in the Character Mode, only DMA mode 0 is allowed. Mode 0 supports interleaved DMA where a transfer is made between CPU bus cycles. Mode 1 supports burst DMA where multiple transfers are made continuously until the RCVR FIFO has been emptied or the XMIT FIFO has been filled.

RXRDY Mode 0: When in the Character Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) and there is at least 1 character in the RCVR FIFO or RCVR holding register, the RXRDY pin (29) will be low active. Once it is activated the RXRDY pin will go inactive when there are no more characters in the FIFO or holding register.

RXRDY Mode 1: In the FIFO Mode (FCR0 = 1) when the FCR3 = 1 and the trigger level or the timeout has been reached, the RXRDY pin will go low active. Once it is activated it will go inactive when there are no more characters in the FIFO or holding register.

TXRDY Mode 0: In the Character Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 1) and there are no characters in the XMIT FIFO or XMIT holding register, the TXRDY pin (24) will be low active. Once it is activated the TXRDY pin will go inactive after the first character is loaded into the XMIT FIFO.

TXRDY Mode 1: In the FIFO Mode (FCR = 1) when FCR3 = 1 and there is at least one unfilled position in the XMIT FIFO, it will go low active. This pin will become inactive when the XMIT FIFO is completely full.

Driver Disable (DDIS), Pin 23: Goes low whenever the CPU is reading data from the ACE. A high-level DDIS output can be used to disable an external transceiver (if used between the CPU and ACE on the D_7–D_0 Data Bus) at all times, except when the CPU is reading data.

Functional Description (Continued)

Baud Out (BAUDOUT), Pin 15: 16 × clock signal for the transmitter section of the ACE. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The BAUDOUT may also be used for the receiver section by tying this output to the RCLK input of the chip.

Interrupt (INTRPT), Pin 30: Goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available: timeout (FIFO Mode only); Transmitter Holding Register Empty; and MODEM Status. The INTRPT signal is reset low upon the appropriate interrupt service or a Master Reset operation.

Serial Output (SOUT), Pin 11: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

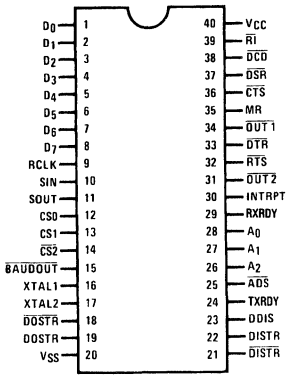
INPUT/OUTPUT SIGNALS

Data (D_7–D_0) Bus, Pins 1–8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the ACE and the CPU. Data, control words, and status information are transferred via the D_7–D_0 Data Bus.

External Clock Input/Output (XTAL 1, XTAL 2) Pins 16 and 17: These two pins connect the main timing reference (crystal or signal clock) to the ACE.

Connection Diagrams

Dual-In-Line Package

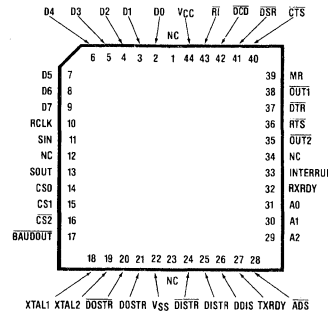


TL/C/8652–17

Top View

Order Number NS16550N
See NS Package Number N40A

Chip Carrier Package



TL/C/8652–18

Top View

Order Number NS16550V
See NS Package Number V44A

TABLE I. ACE Reset Configuration

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	All Bits Low (0–3 forced and 4–7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High, Bits 1–3, 7 forced low Bits 4–6 are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	All Bits Low, Except Bits 5 and 6 are High
MODEM Status Register	Master Reset	Bits 0–3 Low, Bits 4–7—Input Signal
SOUT	Master Reset	High
INTRPT (RCVR Errs)	Read LSR/MR	Low
INTRPT (RCVR Data Ready)	Read RBR/MR	Low
INTRPT (THRE)	Read IIR/Write THR/MR	Low
INTRPT (Modem Status Changes)	Read MSR/MR	Low
OUT 2	Master Reset	High
RTS	Master Reset	High
DTR	Master Reset	High
OUT 1	Master Reset	High
RCVR FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low
XMIT FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low
FIFO Control	Master Reset	All Bits Low

TABLE II. Summary of Registers

Bit No.	Register Address											
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	2	3	4	5	6	7	0 DLAB = 1	1 DLAB = 1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFO Enable (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Note 2: These bits are always 0 in the Character mode.

Registers

The system programmer may access or control any of the ACE registers summarized in Table II via the CPU. These registers are used to control ACE operations and to transmit and receive data.

LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange via the Line Control Register. In addition to controlling the format, the programmer may retreive the contents of the Line Control Register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the Line Control Register are indicated in Table II and are described below.

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits in each transmitted character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked by the receiver as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1.

Bit 6: This bit is the Break Control bit. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

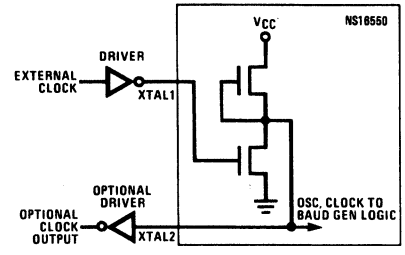
Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load in all 0s, a pad character, in response to THRE.
2. Set break after the next THRE.
3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

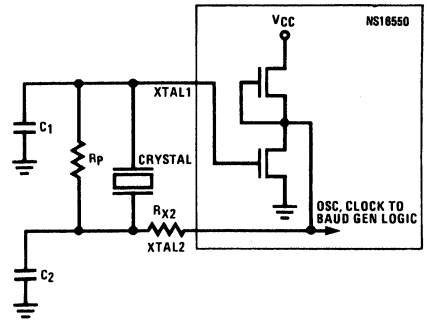
During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

Typical Clock Circuits



TL/C/8652–19



TL/C/8652–20

Typical Crystal Oscillator Network

CRYSTAL	R_P	R_{X2}	C_1	C_2
3.1 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF
1.8 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF

TABLE III. Baud Rates Using 1.8432 MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

Registers (Continued)

PROGRAMMABLE BAUD GENERATOR

The ACE contains a programmable Baud Generator that is capable of taking any clock input (DC to 8.0 MHz) and dividing it by any divisor from 2 to $2^{16}-1$). 4 MHz is the highest input clock frequency recommended when the divisor = 1. The output frequency of the Baud Generator is 16 × the Baud [divisor # = (frequency input) ÷ (baud rate × 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure desired operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded. This prevents long counts on initial load.

Tables III, IV and V illustrate the divisors for use with crystal frequencies of 1.8432 MHz, 3.072 MHz and 8 MHz, respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen.

LINE STATUS REGISTER

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the Line Status Register are indicated in Table II and are described below.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to a logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO.

TABLE IV. Baud Rates Using 3.072 MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status indicator. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO.

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status indicator. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes to the marking state and receives the next valid start bit.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected.

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the ACE is ready to accept a new character for transmission. In addition, this bit causes the ACE to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the

TABLE V. Baud Rates Using 8 MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	10000	—
75	6667	0.005
110	4545	0.010
134.5	3717	0.013
150	3333	0.010
300	1667	0.020
600	833	0.040
1200	417	0.080
1800	277	0.080
2000	250	—
2400	208	0.160
3600	139	0.080
4800	104	0.160
7200	69	0.644
9600	52	0.160
19200	26	0.160
38400	13	0.160
56000	9	0.790
128000	4	2.344
256000	2	2.344

Registers (Continued)

TABLE VI. Interrupt Control Functions

FIFO Mode Only	Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	0	1	—	None	None	—
0	1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
0	1	0	0	Second	Received Data Available	Receiver Data Available or Trigger Level Reached	Reading the Receiver Buffer Register or the FIFO Drops Below the Trigger Level
1	1	0	0	Second	Character Timeout Indication	No Characters Have Been Removed From or Input to the RCVR FIFO During the Last 4 Char. Times and There Is at Least 1 Char. in It During This Time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU. In the FIFO mode this bit is set when the XMIT FIFO is empty; it is cleared when at least 1 byte is written to the XMIT FIFO.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to a logic 0 whenever either the THR or TSR contains a data character. In the FIFO mode this bit is set to one whenever the transmitter FIFO and shift register are both empty.

Bit 7: In the Character mode this is a 0. In the FIFO mode LSR7 is set when there is at least one parity error, framing error or break indication in the FIFO. LSR7 is cleared when the CPU reads the LSR, if there are no subsequent errors in the FIFO.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is used for factory testing.

FIFO CONTROL REGISTER

This is a write only register at the same location as the IIR (the IIR is a read only register). This register is used to enable the FIFOs, clear the FIFOs, set the RCVR FIFO trigger level, and select the type of DMA signalling.

Bit 0: Writing a 1 to FCR0 enables both the XMIT and RCVR FIFOs. Resetting FCR0 will clear all bytes in both FIFOs. When changing from FIFO Mode to Character Mode and vice versa, data is automatically cleared from the FIFOs. This bit must be a 1 when other FCR bits are written to or they will not be programmed.

Bit 1: Writing a 1 to FCR1 clears all bytes in the RCVR FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 2: Writing a 1 to FCR2 clears all bytes in the XMIT FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 3: Setting FCR3 to a 1 will cause the RXRDY and TXRDY pins to change from mode 0 to mode 1 if FCR0 = 1 (see description of RXRDY and TXRDY pins).

Bit 4, 5: FCR4 to FCR5 are reserved for future use.

Bit 6, 7: FCR6 and FCR7 are used to set the trigger level for the RCVR FIFO interrupt.

7	6	RCVR FIFO Trigger Level (Bytes)
0	0	01
0	1	04
1	0	08
1	1	14

INTERRUPT IDENTIFICATION REGISTER

The ACE has an on-chip interrupt capability that allows for flexibility in interfacing popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the ACE prioritizes interrupts into four levels. The four levels of interrupt conditions are as follows: Receiver Line Status (priority 1); Received Data Ready (priority 2); Timeout (priority 2, FIFO Mode only); Transmitter Holding Register Empty (priority 3); and MODEM Status (priority 4).

Registers (Continued)

Information indicating that a prioritized interrupt is pending and the type of that interrupt is stored in the Interrupt Identification Register (IIR). When addressed during chip-select time, the IIR freezes the highest priority interrupt pending and no other interrupts change the IIR even though they are recorded, until that particular interrupt is serviced by the CPU. The contents of the IIR are indicated in Table II and are described below.

Bit 0: This bit can be used in either a prioritized interrupt or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending and polling (if used) continues.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table VI.

Bit 3: In the Character mode this bit is 0. In the FIFO mode this bit is set along with bit 2 when a timeout interrupt is pending.

Bits 4 through 6: These three bits of the IIR are always logic 0.

Bit 7: This bit is set when FCR0 = 1.

INTERRUPT ENABLE REGISTER

This 8-bit register enables the four types of interrupts of the ACE to separately activate the chip interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register. Similarly, by setting the appropriate bits of this register to a logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the Interrupt Identification Register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. The contents of the Interrupt Enable Register are indicated in Table II and are described below.

Bit 0: This bit enables the Received Data Available Interrupt (and timeout interrupts in the FIFO mode) when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

MODEM CONTROL REGISTER

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table II and are described below.

Bit 0: This bit controls the Data Terminal Ready ($\overline{DTR}$) output. When bit 0 is set to a logic 1, the $\overline{DTR}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{DTR}$ output is forced to a logic 1.

Note: The $\overline{DTR}$ output of the ACE may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{RTS}$) output. Bit 1 affects the $\overline{RTS}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{OUT\ 1}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{OUT\ 1}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{OUT\ 2}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{OUT\ 2}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the ACE. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{CTS}$, $\overline{DSR}$, $\overline{DCD}$, and $\overline{RI}$) are disconnected; and the four MODEM Control outputs ($\overline{DTR}$, $\overline{RTS}$, $\overline{OUT\ 1}$, and $\overline{OUT\ 2}$) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and received-data paths of the ACE.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. Their sources are external to the part. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

MODEM STATUS REGISTER

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The contents of the MODEM Status Register are indicated in Table II and described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{DSR}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{RI}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{DCD}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{CTS}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{DSR}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent of DTR in the MCR.

Registers (Continued)

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{DCD}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 of the MCR.

SCRATCHPAD REGISTER

This 8-bit Read/Write Register does not control the ACE in anyway. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

FIFO INTERRUPT MODE OPERATION

When the RCVR FIFO and receiver interrupts are enabled (FCR0 = 1, IER0 = 1) RCVR interrupts will occur as follows:

A. The receive data available interrupt will be issued to the CPU when the FIFO has reached its programmed trigger level; it will be cleared as soon as the FIFO drops below its programmed trigger level.

B. The IIR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt it is cleared when the FIFO drops below the trigger level.

C. The receiver line status interrupt (IIR-06), as before, has higher priority than the received data available (IIR = 04) interrupt.

D. The data ready bit (LSR0) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset when the FIFO is empty.

When RCVR FIFO and receiver interrupts are enabled, RCVR FIFO timeout interrupts will occur as follows:

A. A FIFO timeout interrupt will occur, if the following conditions exist:

- — at least one character is in the FIFO
- — the most recent character received was longer than 4 continuous character times ago (if 2 stop bits are programmed the second one is included in this time delay).
- — the most recent CPU read of the FIFO was longer than 4 continuous character times ago.

This will cause a maximum character received to interrupt issued delay of 160 ms at 300 BAUD with a 12 bit character.

B. Character times are calculated by using the RCLK input for a clock signal (this makes the delay proportional to the baudrate).

C. When a timeout interrupt has occurred it is cleared and the timer reset when the CPU reads one character from the RCVR FIFO.

D. When a timeout interrupt has not occurred the timeout timer is reset after a new character is received or after the CPU reads the RCVR FIFO.

When the XMIT FIFO and transmitter interrupts are enabled (FCR0 = 1, IER1 = 1), XMIT interrupts will occur as follows:

A. The transmitter holding register interrupt (02) occurs when the XMIT FIFO is empty; it is cleared as soon as the transmitter holding register is written to (1 to 16 characters may be written to the XMIT FIFO while servicing this interrupt) or the IIR is read.

B. The transmitter FIFO empty indications will be delayed 1 character time minus the last stop bit time whenever the following occurs: THRE = 1 and there have not been at least two bytes at the same time in the transmit FIFO, since the last THRE = 1. The first transmitter interrupt after changing FCR0 will be immediate, if it is enabled.

Character timeout and RCVR FIFO trigger level interrupts have the same priority as the current received data available interrupt; XMIT FIFO empty has the same priority as the current transmitter holding register empty interrupt.

FIFO POLLED MODE OPERATION

With FCR0 = 1 resetting IER0, IER1, IER2, IER3 or all to zero puts the ACE in the FIFO Polled Mode of operation. Since the RCVR and XMITTER are controlled separately either one or both can be in the polled mode of operation.

In this mode the user's program will check RCVR and XMITTER status via the LSR. As stated previously:

LSR0 will be set as long as there is one byte in the RCVR FIFO.

LSR1 to LSR4 will specify which error(s) has occurred. Character error status is handled the same way as when in the interrupt mode, the IIR is not affected since IER2 = 0.

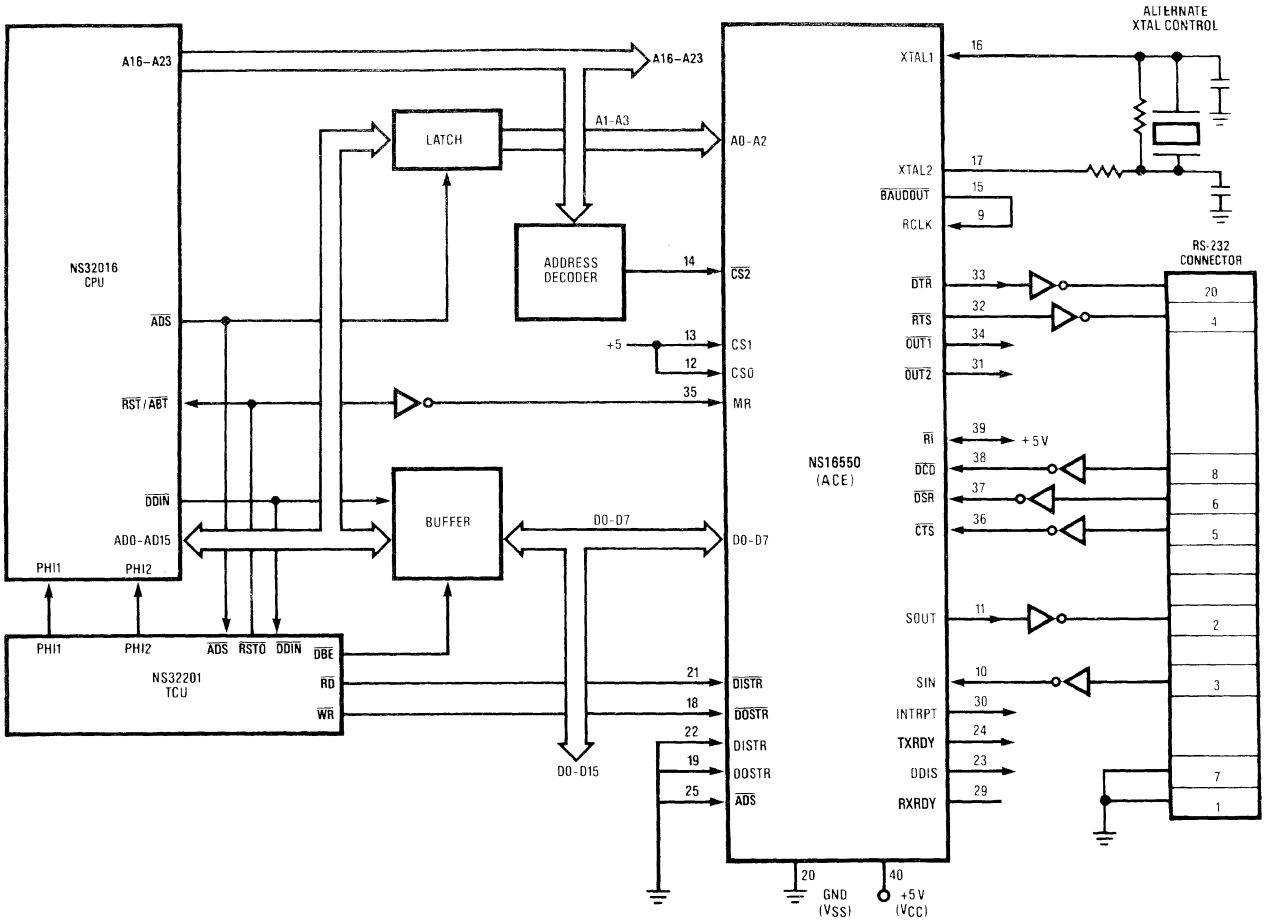
LSR5 will indicate when the XMIT FIFO is empty.

LSR6 will indicate that both the XMIT FIFO and shift register are empty.

LSR7 will indicate whether there are any errors in the RCVR FIFO.

There is no trigger level reached or timeout condition indicated in the FIFO Polled Mode, however, the RCVR and XMIT FIFOs are still fully capable of holding characters.

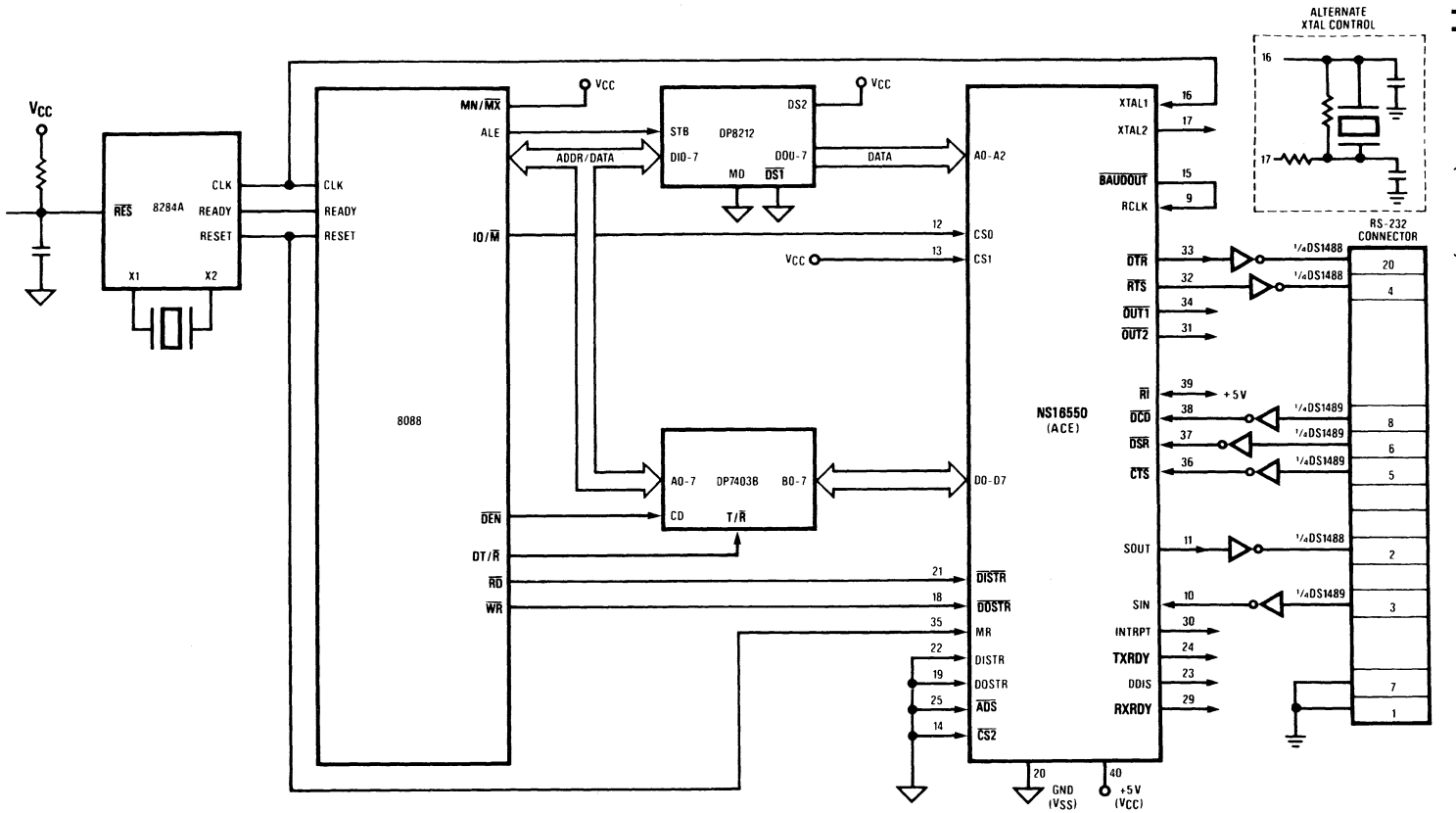
This shows the basic connections of an NS16550 to an NS32016 CPU



TL/C/8652–21

Typical Applications (Continued)

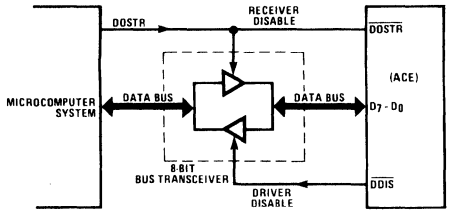
This shows the basic connections of an NS16550 to an 8088 CPU



TL/C/8652–22

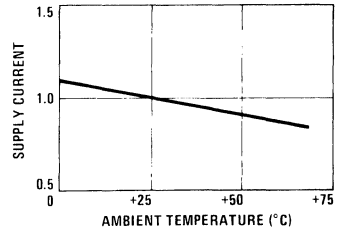
Typical Applications (Continued)

Typical Interface for a High-Capacity Data Bus



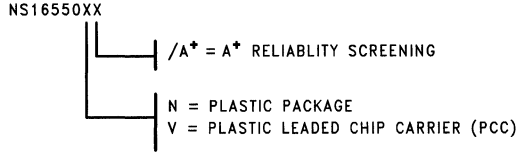
TL/C/8652-23

Typical Supply Current vs. Temperature, Normalized



TL/C/8652-24

Ordering Information



TL/C/8652-25

Section 5

Data Communications and LANs

Section 5 Contents

HPC16040 High-Performance Microcontroller 5-3
NS405/NS32405—Series Display/Terminal Management Processor 5-26
DP8390/NS32490 Network Interface Controller 5-65
DP8391/NS32491 Serial Network Interface 5-113
DP8392/NS32492 Coaxial Transceiver Interface 5-122
DP8340/NS32440 Serial Bi-Phase Transmitter/Encoder 5-130
DP8341/NS32441 Serial Bi-Phase Receiver/Decoder 5-139
DP8342/NS32442 High-Speed Serial Transmitter/Encoder 5-150
DP8343/NS32443 High-Speed Serial Receiver/Decoder 5-160

National Semiconductor Corporation

PRELIMINARY

HPC16040 High-Performance Microcontrollers

General Description

The HPC16040 is a member of the HPC™ family of high-performance microcontrollers. The HPC16040 is designed 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.

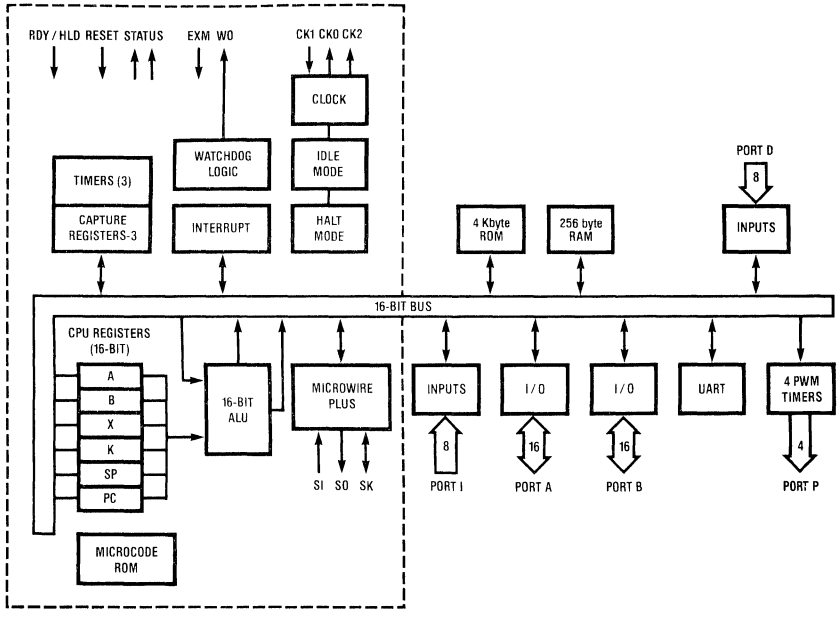
The HPC16040 is a complete microcomputer on a single chip. All system timing, internal logic, ROM, RAM, and I/O are provided on the chip to produce a cost effective solution for high-performance applications. On-chip functions such as UART, 16-bit timers, and MICROWIRE/PLUS™ provide a high level of system integration. The ability to address up to 64 kbytes of external memory enables the HPC16040 to be used in powerful applications typically performed by microprocessors and costly peripheral chips.

The microCMOS process results in very low current drain and enables the user to select the optimum speed/power product for his system. The HPC16040 is available in 68-pin PCC and LCC packages, and a 48-pin ceramic DIP package.

Features

- 16-bit architecture: both byte and word
- CMOS — very low power with two power save modes: IDLE and HALT (2 mA, 25 μA)
- FAST!—240 ns for register instructions when using 17.0 MHz clock
- 64 kbytes of external memory addressing
- UART — full duplex, programmable baud rate
- WATCHDOG logic monitors processor
- MICROWIRE/PLUS serial I/O interface
- 16 x 16-bit multiply and divide
- 16-bit data bus, ALU, and registers
- 4 kbyte ROM, 256 byte RAM
- ROMless versions available
- High code efficiency: most instructions are single byte
- Eight vectored interrupt sources
- 52 general purpose I/O lines
- Powerful set of 16-bit timer/counters:
  Ten timer synchronous outputs
  Three input capture registers
- Wide voltage supply range: 3 to 5.5V
- Industrial (−40°C to +85°C) and extended (−55°C to +125°C) temperature ranges

Block Diagram



TL/DD/8340-1

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

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

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 HPC46040, −40°C to +85°C for HPC36040, −40°C to +105°C for HPC26040, −55°C to +125°C for HPC16040

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
I_{CC_1}	Supply Current	V_{CC} = 5.0V, f_{in} = 17.0 MHz*		20		mA
		V_{CC} = 5.0V, f_{in} = 2.0 MHz		2.4		mA
I_{CC_2}	IDLE Mode Current	V_{CC} = 5.0V, f_{in} = 17.0 MHz, T_A = 25°C*		2		mA
		V_{CC} = 5.0V, f_{in} = 2.0 MHz, T_A = 25°C		0.2		mA
I_{CC_3}	HALT Mode Current	V_{CC} = 5.0V, f_{in} = 0 kHz, T_A = 25°C*		25		μA
		V_{CC} = 2.5V, f_{in} = 0 kHz, T_A = 25°C		10		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
C_I	Input Capacitance			10		pF
C_{IO}	I/O Capacitance			20		pF
OUTPUT VOLTAGE LEVELS CMOS OPERATION						
V_{OH_1}	Logic High	I_{OH} = −10 μA		V_{CC} − 0.2		V
V_{OL_1}	Logic Low	I_{OH} = 10 μA		0.2		V
V_{OH_2}	Port A/B Drive (A_0–A_{15}, B_{10}, B_{11}, B_{12}, B_{15})	I_{OH} = −7 mA, V_{CC} = 5.0V		2.4		V
V_{OL_2}		I_{OL} = 3 mA		0.4		V
V_{OH_3}	Other Port Pin Drive (B_0–B_9, B_{13}, B_{14}, P_0–P_3)	I_{OH} = −1.6 mA, V_{CC} = 5.0V		2.4		V
V_{OL_3}		I_{OL} = 0.5 mA		0.4		V
V_{OL_4}	WO (Watchdog Out) Drive	I_{OL} = 0.5 mA, V_{CC} = 5.0V		0.4		V
V_{OH_6}	CK2 Drive	I_{OH} = −12 mA, V_{CC} = 5.0V		2.4		V
V_{OL_6}		I_{OL} = 3.5 mA		0.4		V
V_{OH_7}	ST1 and ST2 Drive	I_{OH} = −6 mA, V_{CC} = 5.0V		2.4		V
V_{OL_7}		I_{OL} = 1.6 mA		0.4		V
V_{RAM}	RAM Keep-Alive Voltage			2.5		V
I_{OZ}	TRI-STATE Leakage Current			±5		μA

***Note:** 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).

AC Electrical Characteristics V_{CC} = 5.0V ±10%, f_C = 16.78 MHz, T_A = 0°C to +70°C for HPC46040, −40°C to +85°C for HPC36040, −40°C to +105°C for HPC26040, −55°C to +125°C for HPC16040

Symbol	Parameter	Min	Typ	Max	Units
f_C = CKI freq.	Operating Frequency		16.78	17.0	MHz
$t_{C1} = 1/f_C$	Clock Period		60		ns
$t_C = 2/f_C$	Timing Cycle		120		ns
$t_{LL} = ½\ t_C$	ALE Pulse Width		60		ns
$t_{ST} = ¼\ t_C$	Address Valid to ALE Trailing Edge		30		ns
$t_{WAIT} = t_C = WS$	Wait State Period		120		ns
$f_{XIN} = 1/19\ t_{C1}$	External Timer Input Frequency		877		kHz
$t_{XIN} = 3\ t_{C1}$	Pulse Width for Timer Inputs		180		ns
$f_{XOUT} = 1/16\ t_{C1}$	Timer Output Frequency		1.04		MHz
$f_{MW} = 1/19\ t_{C1}$	External MICROWIRE/PLUS Clock Input Frequency		877		kHz
$f_U = 1/19\ t_{C1}$	External UART Clock Input Frequency		877		kHz

Read Cycle Timing f_C = 16.78 MHz with One Wait State

Symbol	Parameter	Min	Typ	Max	Units
$t_{ARR} = ¼\ t_C + 5$	ALE Trailing Edge to $\overline{RD}$ Falling Edge		35		ns
$t_{RW} = ½\ t_C + WS$	$\overline{RD}$ Pulse Width		180		ns
t_{DR}	Data Valid before Trailing Edge of $\overline{RD}$		15		ns
$t_{ACC} = t_C + WS - 55$	Address Valid to Input Data Valid		185		ns
$t_{RD} = ½\ t_C + WS - 65$	$\overline{RD}$ Falling Edge to Data in Valid		115		ns
$t_{RDA} = WS = t_C$	$\overline{RD}$ Falling Edge to Address Valid		120		ns
$t_{VPR} = ¼\ t_C + 5$	Address Valid from ALE Trailing Edge Prior to $\overline{RD}$		35		ns
$t_{HZ} = ¾\ t_C - 10$	End of $\overline{RD}$ to Input Data Float		80		ns

Write Cycle Timing f_C = 16.78 MHz with One Wait State

Symbol	Parameter	Min	Typ	Max	Units
$t_{ARW} = ½\ t_C$	ALE Trailing Edge to $\overline{WR}$ Falling Edge		60		ns
$t_{WW} = ¾\ t_c + WS + 5$	$\overline{WR}$ Pulse Width		215		ns
t_{HW}	Data Hold after Trailing Edge of $\overline{WR}$		20		ns
$t_V = ½\ t_C + WS + 5$	Data Valid before Trailing Edge of $\overline{WR}$		185		ns
$t_{VPW} = ¼\ t_C + 25$	Address Valid from Trailing Edge Prior to $\overline{WR}$		55		ns

Ready/Hold Timing f_C = 16.78 MHz with One Wait State

Symbol	Parameter	Min	Typ	Max	Units
$t_{DAR} = \frac{1}{4} t_C + WS - 50$	Falling Edge of ALE to Falling Edge of $\overline{RDY}$		100		ns
$t_{RWP} = t_C$	$\overline{RDY}$ Pulse Width		120		ns
$t_{SALE} = \frac{1}{4} t_C + 40$	Falling Edge of $\overline{HLD}$ to Rising Edge of ALE		70		ns
$t_{HWP} = t_C + 10$	$\overline{HLD}$ Pulse Width		130		ns
t_{HAD}	Rising Edge on $\overline{HLD}$ to Rising Edge on $\overline{HLDA}$		120		ns
$t_{HAE} = t_C + 100$	Falling Edge on $\overline{HLD}$ to Falling Edge on $\overline{HLDA}$		220	*	ns
$t_{BF} = t_C + 30$	Bus Float before Falling Edge on $\overline{HLDA}$		150		ns
$t_{BE} = 2 t_C + 50$	Bus Enable from Rising Edge of $\overline{HLD}$		290		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.

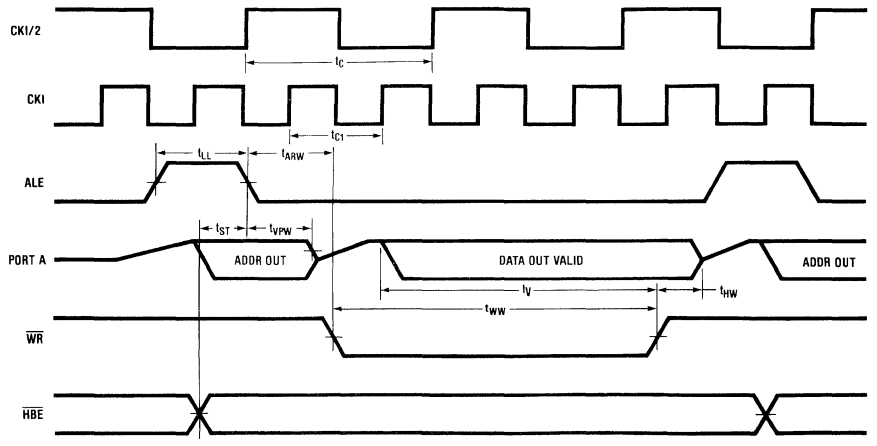
Status Timing f_C = 16.78 MHz

Symbol	Parameter	Min	Typ	Max	Units
$t_{SRS2} = 40 - (\frac{1}{4} t_C + 25)$	Setup Time for ST2 on Rising Edge of ALE		−15		ns
$t_{HRS2} = \frac{3}{4} t_C - 15$	Hold Time for ST2 on Rising Edge of ALE		75		ns
$t_{SFS2} = 40 - (\frac{1}{4} t_C + 25)$	Setup Time for ST2 on Falling Edge of ALE		−15		ns
$t_{HFS2} = \frac{3}{4} t_C - 15$	Hold Time for ST2 on Falling Edge of ALE		75		ns
t_{SFS1}	Setup Time for ST1 on Falling Edge of $\overline{RD}$		20		ns
$t_{HRS1} = \frac{1}{2} t_C - 15$	Hold Time for ST1 on Rising Edge of $\overline{RD}$		45		ns

UPI Read/Write Timing

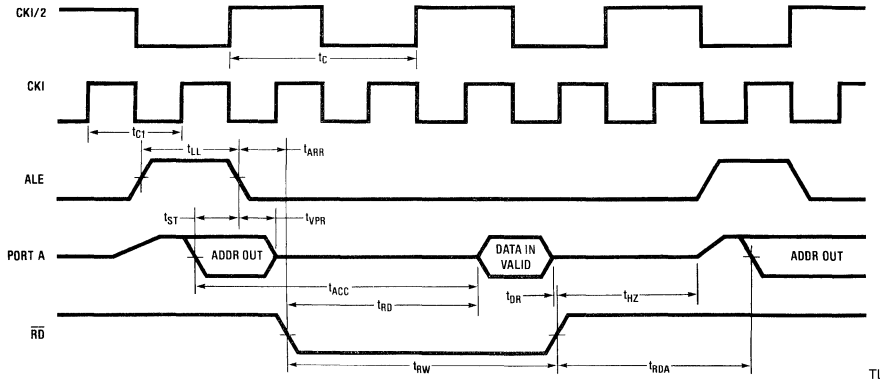
Symbol	Parameter	Min	Typ	Max	Units
t_{UAS}	Address Setup Time to Falling Edge of $\overline{UPIRD}$		5		ns
t_{UAH}	Address Hold Time from Rising Edge of $\overline{UPIRD}$		5		ns
t_{RPW}	$\overline{UPIRD}$ Pulse Width		100		ns
t_{OE}	$\overline{UPIRD}$ Falling Edge to Data Out Valid		60		ns
t_{OD}	End of $\overline{UPIRD}$ to Data Out Valid		35		ns
t_{DRDY}	$\overline{RDRDY}$ Delay from Trailing Edge of $\overline{UPIRD}$		70		ns
t_{WDW}	$\overline{UPIWR}$ Pulse Width		40		ns
t_{UDS}	Data in Valid before Trailing Edge of $\overline{UPIWR}$		10		ns
t_{UDH}	Data in Hold after Trailing Edge of $\overline{UPIWR}$		15		ns
t_A	$\overline{WRRDY}$ Delay from Trailing Edge of $\overline{UPIWR}$		70		ns

Timing Waveforms



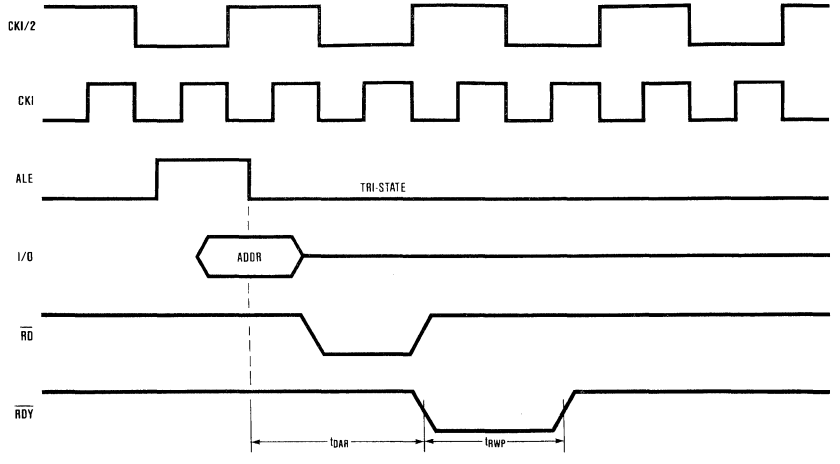
TL/DD/8340–2

FIGURE 1. Write Cycle



TL/DD/8340–3

FIGURE 2. Read Cycle

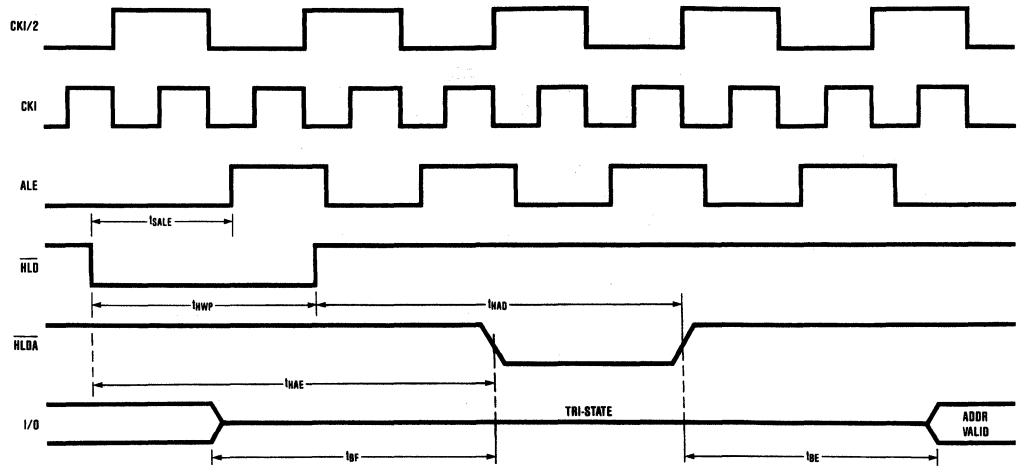


TL/DD/8340–4

FIGURE 3. Ready Mode Timing

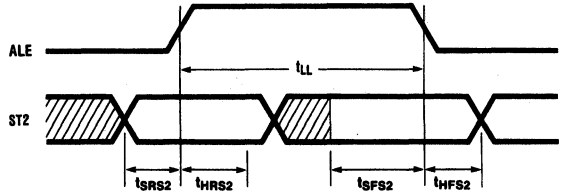
5

Timing Waveforms (Continued)

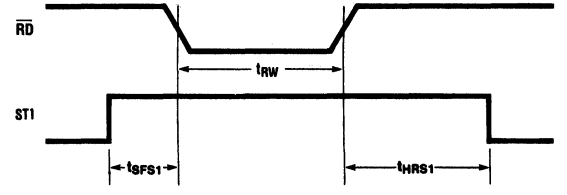


TL/DD/8340-5

FIGURE 4. Hold Mode Timing

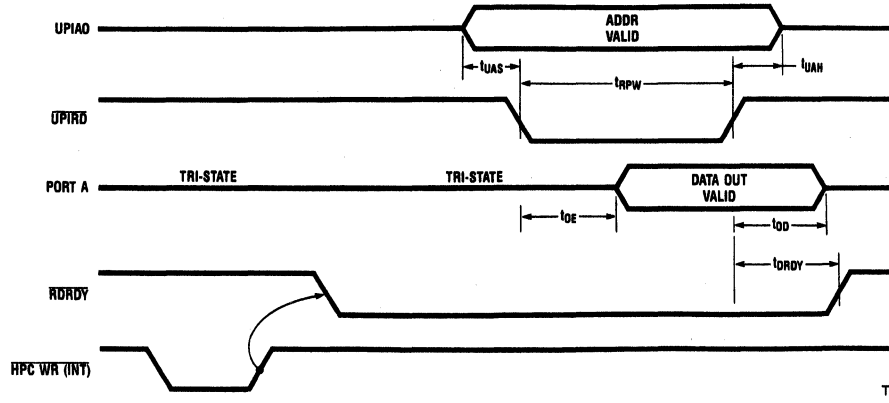


TL/DD/8340-6



TL/DD/8340-7

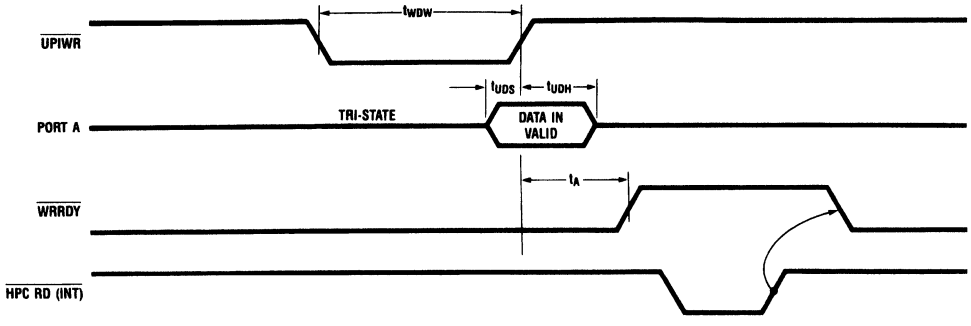
FIGURE 5. Status Timing



TL/DD/8340-8

FIGURE 6. UPI Read Timing

Timing Waveforms (Continued)



TL/DD/8340–9

FIGURE 7. UPI Write Timing

Pin Descriptions

The HPC16040 is available in 68-pin PCC and LCC packages, and a 48-pin ceramic DIP.

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:	WR	Write Output
B12:	$\overline{\text{HBE}}$	High Byte Enable Output
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 through 7 in order to generate frequency, duty cycle and pulse width modulated outputs.

POWER SUPPLY PINS

V_{CC}	Positive Power Supply (3V to 5.5V)
GND	Ground for On-Chip Logic
DGND	Ground for Output Buffers

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

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 and interrupt).

5

Pin Descriptions (Continued)

RESET is an active low input that forces the chip to restart and sets the ports in a TRI-STATE® mode.

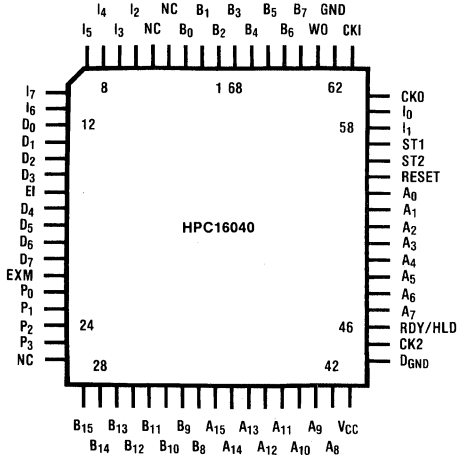
RDY/HLD has two uses, selected by a software bit. It's either a READY input to extend the bus cycle for slower memories, or a HOLD request input to put the bus in a high impedance state for DMA purposes.

NC (no connection) unused at this time.

EXM External memory enable (active high) disables internal ROM and maps it to external memory.

EI External interrupt with vector address FFF1:FFF0.

Plastic and Leadless Chip Carriers



TL/DD/8340–10

Top View

Order Number HPC16040E or V
See NS Package Number E68A or V68A

48-Pin Package Description

A 48-pin version of the HPC16040 called the HPC16041 will be available for applications where the surface mount 68-pin package is not practical. The 48-pin HPC16041 includes all of the most powerful features of the 68-pin HPC16040 such as:

Port A: (A0–A15) All of Port A has been retained.

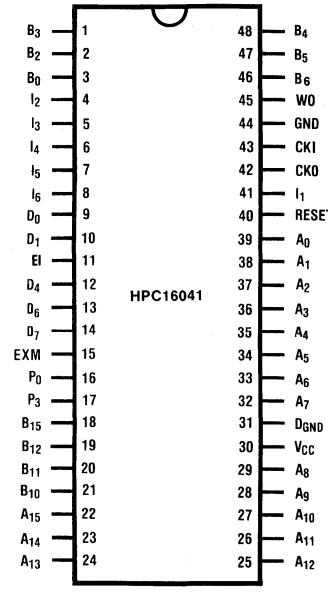
Port B: (B0, B2–B6, B10–B12, B15) These pins retain functions such as UART, timer I/O, MICROWIRE/PLUS, memory expansion, and UPI.

Port D: (D4–D7) Provides four outputs.

Port I: (I1–I6) Maintains all external interrupt capability, UART, and MICROWIRE/PLUS.

Port P: (P0, P3) Provides two synchronous timer outputs.

48-Pin Ceramic DIP



TL/DD/8340–11

Top View

Order Number HPC16041D
See NS Package Number D48A

The 48-pin package pin arrangement is subject to change without notice.

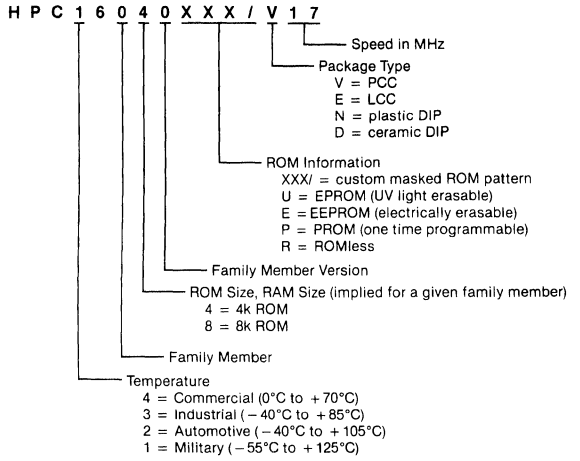
Part Selection

The HPC family includes devices with many different options and configurations to meet various application needs. Table I shows the parts that are represented by this data sheet. The name HPC16040 is used throughout this data sheet to represent the parts listed.

The following chart identifies the various options available when ordering HPC family members.

Note: All options may not currently be available.

Part Selection (Continued)



TL/DD/8340-12

FIGURE 8. HPC Family Part Numbering Scheme

TABLE I. Part Numbers Represented by This Device Description

Part Number	Program Memory	Special Function	Package	Temperature Range
HPC16040XXX/V17	4k ROM	UART, 8 Timers	68-Pin PCC	−55°C to +125°C
HPC16040XXX/E17	4k ROM	UART, 8 Timers	68-Pin LCC	−55°C to +125°C
HPC16040RV17	ROMless	UART, 8 Timers	68-Pin PCC	−55°C to +125°C
HPC16040RE17	ROMless	UART, 8 Timers	68-Pin LCC	−55°C to +125°C
HPC26040XXX/V17	4k ROM	UART, 8 Timers	68-Pin PCC	−40°C to +105°C
HPC26040XXX/E17	4k ROM	UART, 8 Timers	68-Pin LCC	−40°C to +105°C
HPC26040RV17	ROMless	UART, 8 Timers	68-Pin PCC	−40°C to +105°C
HPC26040RE17	ROMless	UART, 8 Timers	68-Pin LCC	−40°C to +105°C
HPC36040XXX/V17	4k ROM	UART, 8 Timers	68-Pin PCC	−40°C to +85°C
HPC36040XXX/E17	4k ROM	UART, 8 Timers	68-Pin LCC	−40°C to +85°C
HPC36040RV17	ROMless	UART, 8 Timers	68-Pin PCC	−40°C to +85°C
HPC36040RE17	ROMless	UART, 8 Timers	68-Pin LCC	−40°C to +85°C
HPC46040XXX/V17	4k ROM	UART, 8 Timers	68-Pin PCC	0°C to +70°C
HPC46040XXX/E17	4k ROM	UART, 8 Timers	68-Pin LCC	0°C to +70°C
HPC46040RV17	ROMless	UART, 8 Timers	68-Pin PCC	0°C to +70°C
HPC46040RE17	ROMless	UART, 8 Timers	68-Pin LCC	0°C to +70°C
HPC16041XXX/D17	4k ROM	UART, 8 Timers	48-Pin Ceramic DIP	−55°C to +125°C
HPC16041RD17	ROMless	UART, 8 Timers	48-Pin Ceramic DIP	−55°C to +125°C
HPC26041XXX/D17	4k ROM	UART, 8 Timers	48-Pin Ceramic DIP	−40°C to +105°C
HPC26041RD17	ROMless	UART, 8 Timers	48-Pin Ceramic DIP	−40°C to +105°C
HPC36041XXX/D17	4k ROM	UART, 8 Timers	48-Pin Ceramic DIP	−40°C to +85°C
HPC36041RD17	ROMless	UART, 8 Timers	48-Pin Ceramic DIP	−40°C to +85°C
HPC46041XXX/D17	4k ROM	UART, 8 Timers	48-Pin Ceramic DIP	0°C to +70°C
HPC46041RD17	ROMless	UART, 8 Timers	48-Pin Ceramic DIP	0°C to +70°C

Ports A and B

The highly flexible A and B ports are similarly structured. The Port A (see *Figure 9*), consists of a data register and a direction register. Port B (see *Figure 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.

Operating Modes

To offer the user a variety of I/O and expanded memory options, the HPC16040 has four operating modes. The four modes are Single-Chip, Expanded, Single-Chip ROMless, and Expanded ROMless. The four modes are determined by the state of both the External Memory (EXM) pin and the External Access (EA) bit in the PSW Register. The HPC16040 System bus consists of port A and four bits of port B. Port A is defined as the address/data bus and the four bits of port B are referred to as the control bus.

SINGLE-CHIP MODE

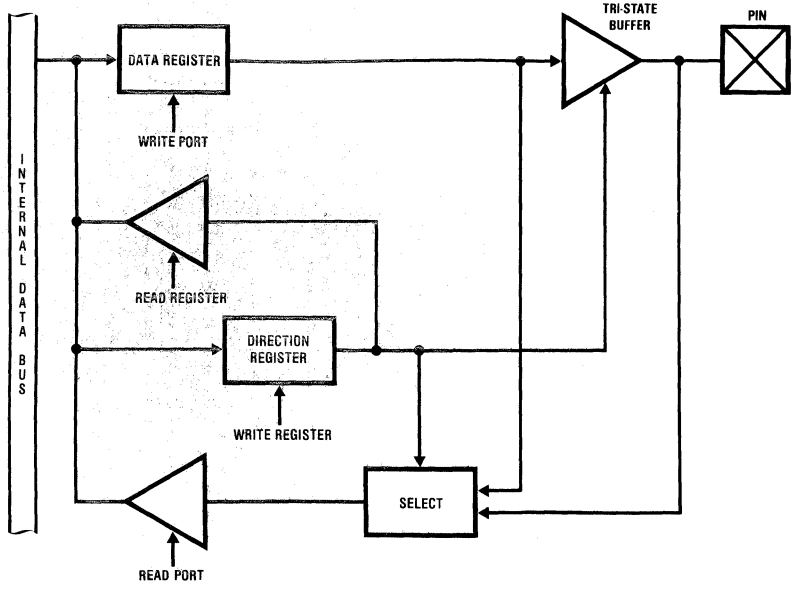
In this mode, the HPC16040 functions as a self-contained microcomputer. It can address internal memory consisting of 256 bytes of RAM and 4 kbytes of ROM. All ports are configured as memory mapped I/O ports. The HPC16040 reads 8 bits or 16 bits of data from the ports, depending on whether a byte or word format instruction is used (see *Figure 11*). The EXM pin and EA bit of the PSW Register are both logic "0" during Single-Chip mode signifying that on-chip ROM is being addressed and the range is limited to 4k (see Table II).

TABLE II. Operating Modes

External Memory Pin (EXM)	External Access Bit (EA)	Operation Mode
0	0	Single Chip
0	1	Expanded
1	0	Single Chip ROMless
1	1	Expanded ROMless

EXPANDED MODE

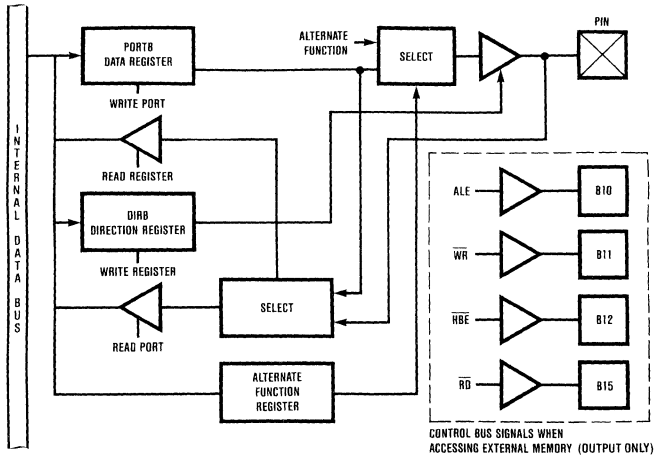
The Expanded mode (see *Figures 12* and *13*) is entered by setting the EA bit in the PSW Register. The HPC16040 can operate within the full 64 kbytes of address space. The 64 kbytes of addressable memory includes all on-chip memory because the EXM pin is grounded during this mode. The external memory may be any combination of RAM and ROM. External memory can be accessed with the data bus defined as either 8 bits wide or 16 bits wide. The System bus may be configured in the 8-bit mode by pulling the $\overline{\text{HBE}}$ pin high at reset. Upon entering the expanded mode, port A



TL/DD/8340–13

FIGURE 9. Port A: I/O Structure

Operating Modes (Continued)



TL/DD/8340-14

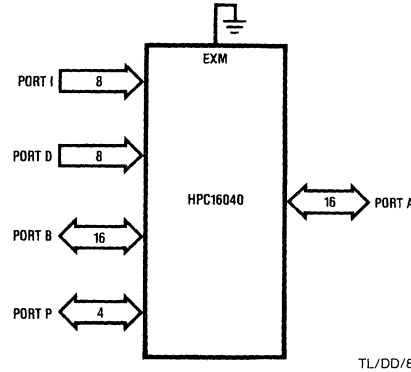
FIGURE 10. Port B: I/O Structure

becomes the Address/Data bus. Four bits of port B become ALE, $\overline{WR}$, $\overline{HBE}$ and $\overline{RD}$ signals. The $\overline{RD}$ and $\overline{WR}$ signals are generated only if the selected address is off-chip. The $\overline{HBE}$ is generated only in the 16-bit bus configuration.

ROMless MODES

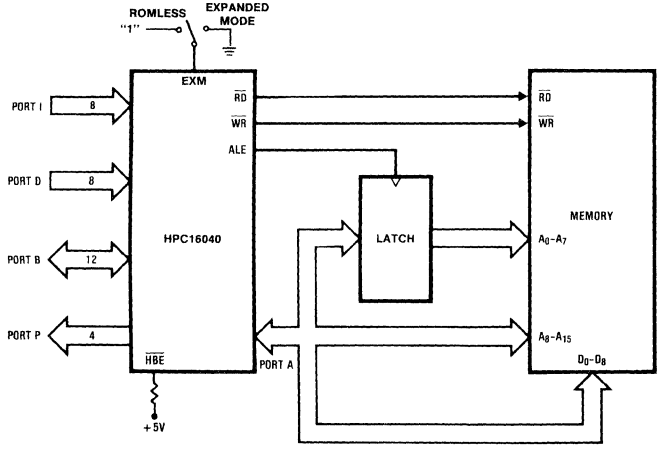
There are two ROMless modes; Single-Chip ROMless and Expanded ROMless. Both ROMless modes are entered by pulling the EXM pin high, (see *Figure 12* and *13*). The EA bit in the PSW Register determines whether the HPC16040 addresses the Single-Chip memory range of 4 kbytes or the Expanded range of 64 kbytes, (see Table II for this information). In both ROMless modes, the HPC16040 continues to use the internal 256 bytes of RAM. The external 4k or 64k of addressed memory may be any combination of RAM and ROM. The address space corresponding to internal ROM is mapped into external memory.

Note: The HPC16040 uses 16-bit words for stack memory. Therefore, when using the 8-bit mode, User's Stack must be in internal RAM.



TL/DD/8340-15

FIGURE 11. Single-Chip Mode

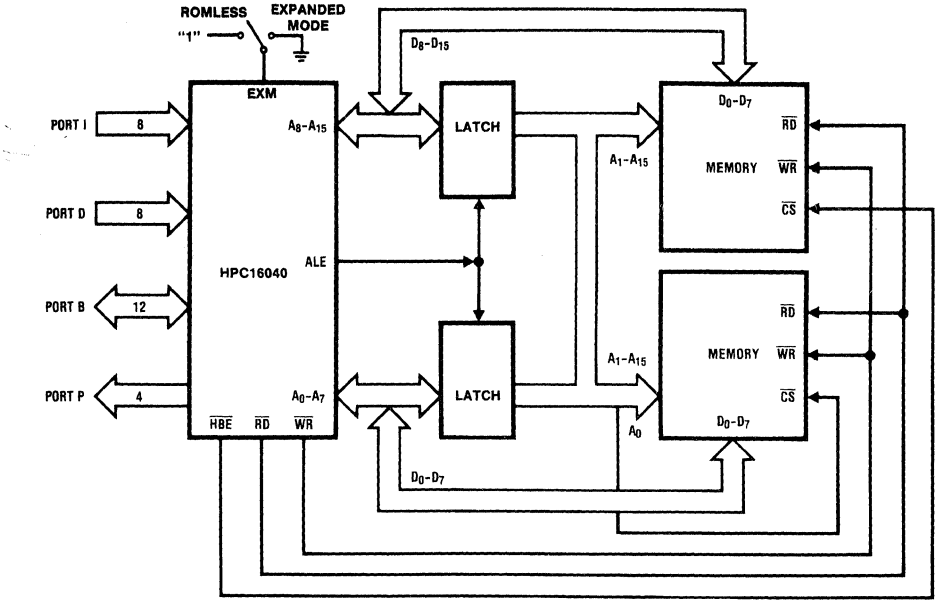


TL/DD/8340-16

FIGURE 12. 8-Bit External Memory

5

Operating Modes (Continued)



TL/DD/8340–17

FIGURE 13. 16-Bit External Memory

Wait States

The HPC16040 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 HPC16040: 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 HPC16040 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 HPC16040 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 RESET or the NMI. The 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 HPC16040 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 HPC16040 resumes normal operation upon timer T0 overflow. As with the HALT mode, the processor is returned to full operation by the RESET or NMI inputs, but without waiting for oscillator stabilization.

HPC16040 Interrupts

Complex interrupt handling is easily accomplished by the HPC16040's vectored interrupt scheme. There are eight possible interrupt sources as shown in Table III.

TABLE III. 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 on the UART transmit/receive complete	6
$FFF1:FFF0	External interrupt on EI pin	7

Interrupt Arbitration

The HPC16040 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 III. 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 is a level-sensitive interrupt. 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.

Interrupt Control Registers

The HPC16040 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 be serviced, 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 HPC16040 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)

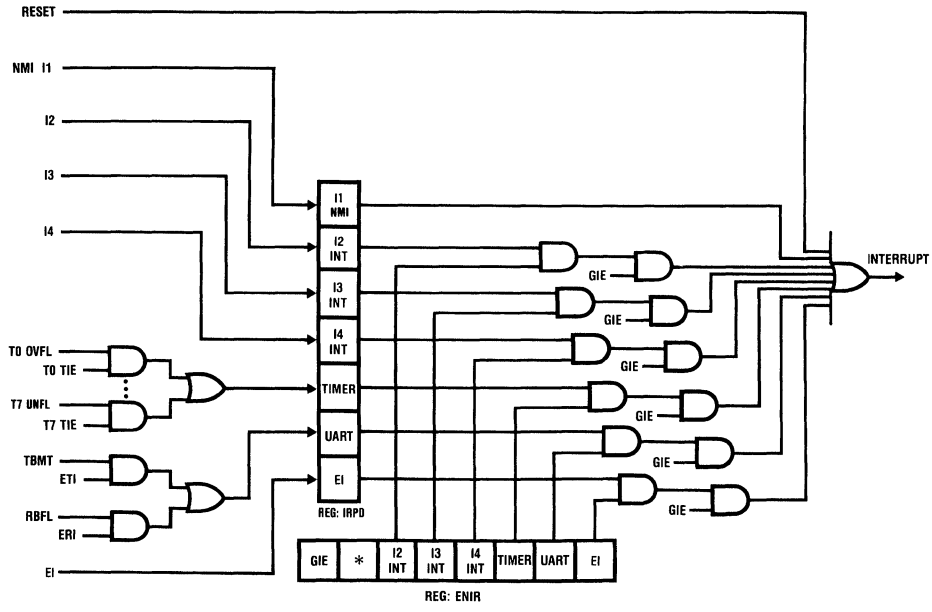
Three bits of the register select the input polarity of the external interrupt on I2, I3, and I4.

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 14* shows the Interrupt Enable Logic.

Reset

The RESET input initializes the processor and sets ports A, B, and P 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.



TL/DD/8340–18

FIGURE 14. Interrupt Enable Logic

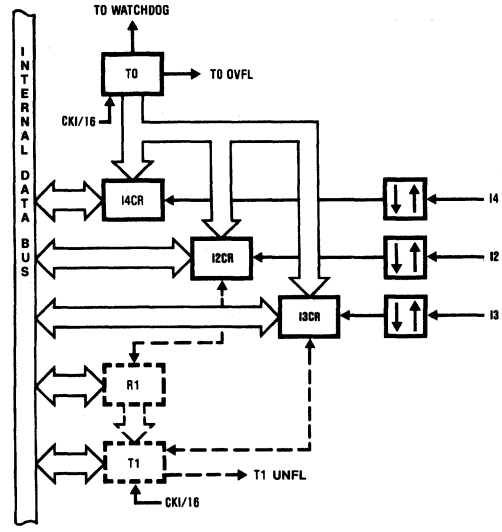
Timer Overview

The HPC16040 contains a powerful set of flexible timers enabling the HPC16040 to perform extensive timer functions; not usually associated with microcontrollers.

The HPC16040 contains eight 16-bit timers. Each timer has 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 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 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.



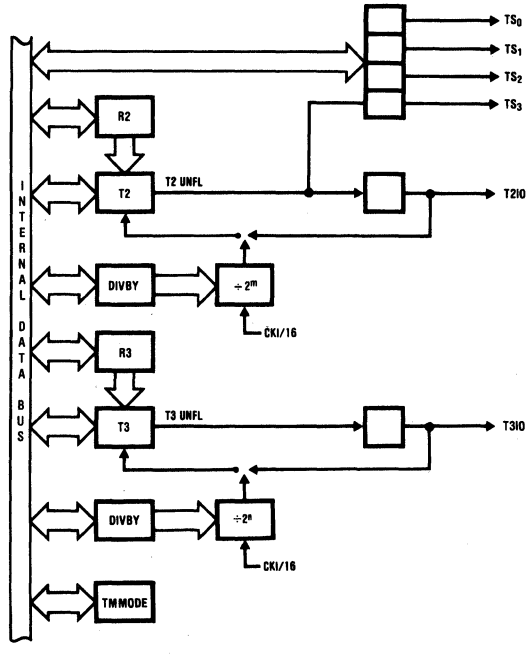
TL/DD/8340-19

FIGURE 15. Timers T0–T1 Block

SYNCHRONOUS OUTPUTS

The flexible timer structure of the HPC16040 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*).

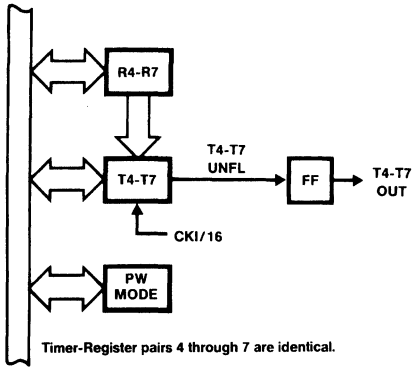
Timer/register pairs 4–7 form four identical units which can generate synchronous outputs on port P (see *Figure 17*).



TL/DD/8340-20

FIGURE 16. Timers T2–T3 Block

Timer Overview (Continued)



TL/DD/8340-21

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 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 HPC16040.

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.

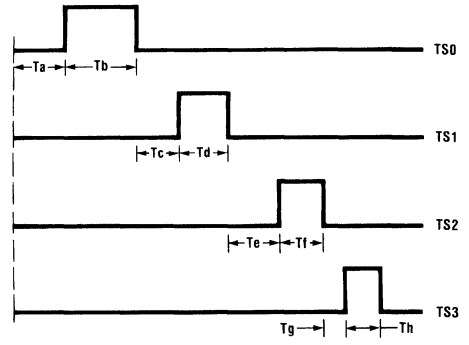
TL/DD/8340-22

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.

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 and illegal addresses. Should the Watch Dog register not be written to before Timer T0 overflows



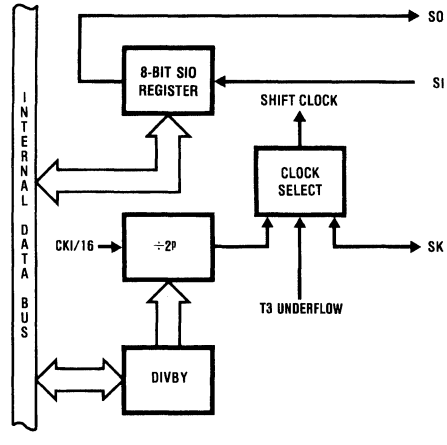
TL/DD/8340-23

FIGURE 19. Synchronous Pulse Generation

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 off-chip address when in the Single-Chip mode. 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 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/8340-24

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

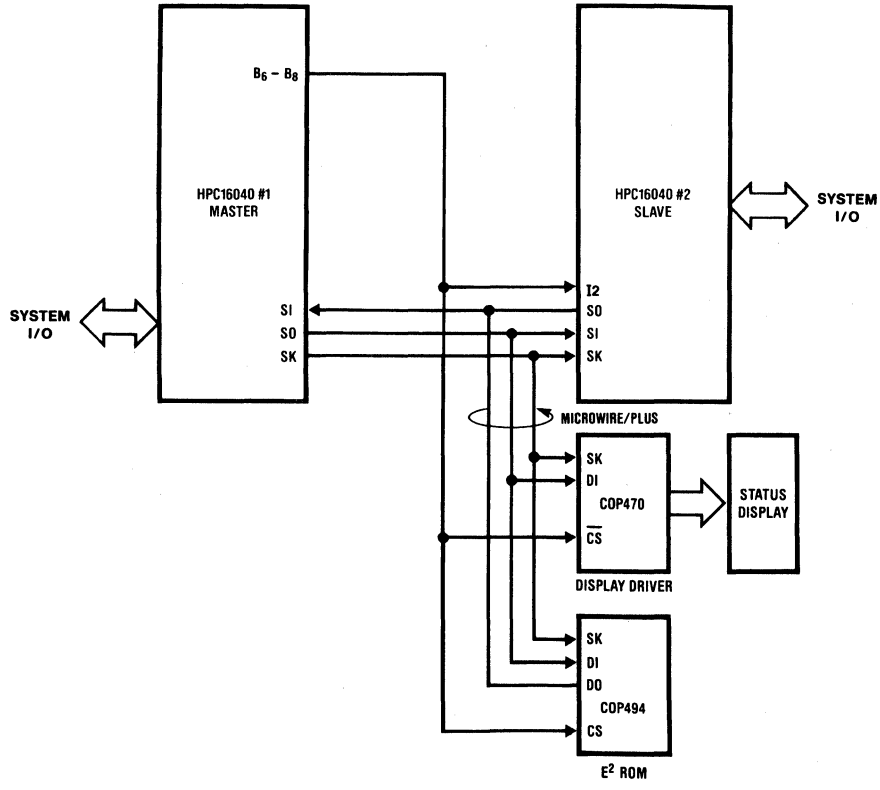
The HPC16040 can enter the MICROWIRE/PLUS mode as the master or a slave. A control bit in the IRCD register determines whether the HPC16040 is the master or slave. The shift clock is generated when the HPC16040 is configured as a master. An externally generated shift clock on the SK pin is used when the HPC16040 is configured as a slave. When the HPC16040 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 17.0 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 HPC16040 microcontrollers interconnected to other MICROWIRE peripherals. HPC16040 #1 is set up as the master and initiates all data transfers. HPC16040 #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 VF display controlled by the COP470 display driver. The data to be displayed is sent serially to the COP470 over the MICROWIRE/PLUS link. Data such as accumulated mileage could be stored and retrieved from the EEPROM COP494. The slave HPC16040 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/8340–25

FIGURE 21. MICROWIRE/PLUS Application

HPC16040 UART

The HPC16040 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.

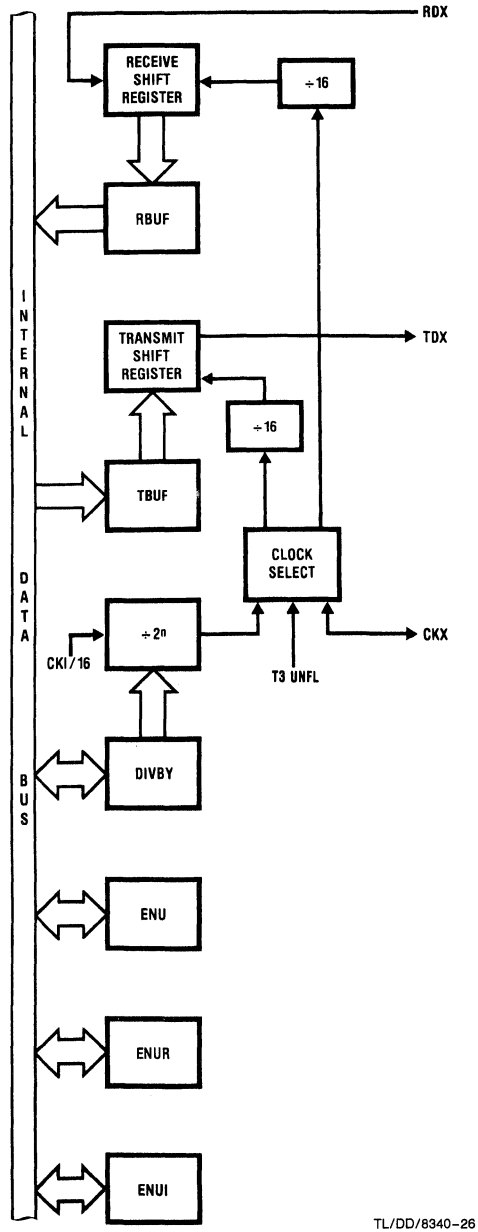
The baud rate clock for the Receiver and Transmitter can be selected for either an internal or external source using two bits in the ENUI register. The internal baud rate is programmed by the DIVBY register. The baud rate may be selected from a range of 8 Hz to 64 kHz in binary steps or T3 underflow. By selecting a 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 HPC16040 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 HPC16040 UART features a Wake-up Mode of operation. This mode of operation enables the HPC16040 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 HPC16040 with an interrupt. The processor then examines the content of the receiver buffer to decide whether it has been addressed and whether to accpet subsequent data.



TL/DD/8340–26

FIGURE 22. UART Block Diagram

Universal Peripheral Interface

The Universal Peripheral Interface (UPI) allows the HPC16040 to be used as an intelligent peripheral to another processor. The UPI could thus be used to tightly link two HPC16040's and set up systems with very high data exchange rates. Another area of application could be where a HPC16040 is programmed as an intelligent peripheral to a host system such as the Series 32000 microprocessor. *Figure 23* illustrates how a HPC16040 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 ($\overline{URD}$), a Write Strobe ($\overline{UWR}$), a Read Ready Line ($\overline{RDRDY}$), a Write Ready Line ($\overline{WRRDY}$) and one Address Input (UA0). The data bus can be either eight or sixteen bits wide.

The $\overline{URD}$ and $\overline{UWR}$ inputs may be used to interrupt the HPC16040. The $\overline{RDRDY}$ and $\overline{WRRDY}$ 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 HPC16040 is the data bus. UPI can only be used if the HPC16040 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 HPC16040 supports shared memory access with two pins. The pins are the RDY/$\overline{HLD}$ input pin and the $\overline{HLDA}$ output pin. The user can software select either the Hold or Ready function by the state of a control bit. The $\overline{HLDA}$ output is multiplexed onto port B.

The host uses DMA to interface with the HPC16040. The host initiates a data transfer by activating the $\overline{HLD}$ input of the HPC16040. In response, the HPC16040 places its system bus in a TRI-STATE Mode, freeing it for use by the host. The host waits for the acknowledge signal ($\overline{HLDA}$) from the HPC16040 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 HPC16040 resumes normal operations.

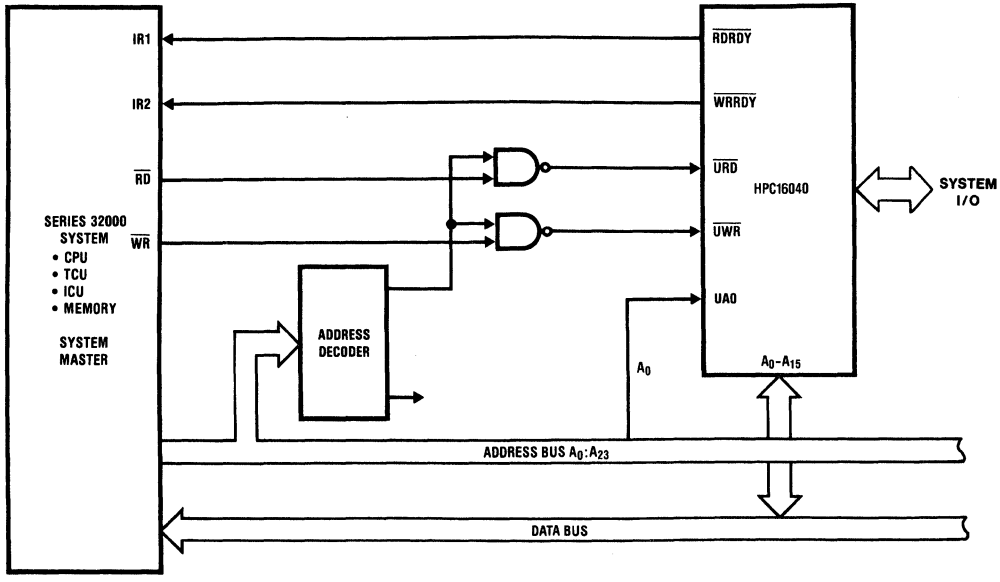
Figure 24 illustrates an application of the shared memory interface between the HPC16040 and a Series 32000 system.

Memory

The HPC16040 has been designed to offer flexibility in memory usage. A total address space of 64 kbytes can be addressed with 4096 bytes 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 HPC16040 uses memory-mapped organization to support registers, I/O and on-chip peripheral functions.

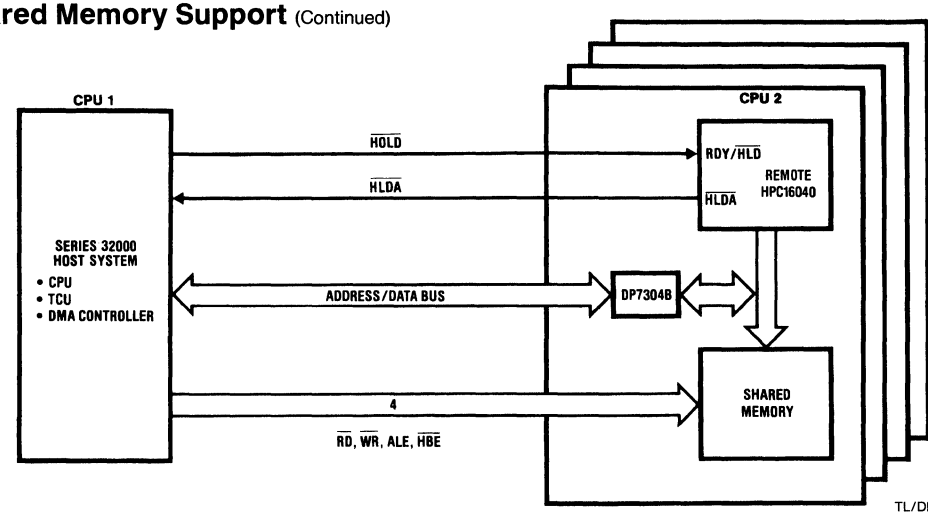
The HPC16040 memory address space extends to 64 kbytes and registers and I/O are mapped as shown in Table IV.



TL/DD/8340–27

FIGURE 23. HPC16040 as a Peripheral: (UPI Interface to Series 32000 Application)

Shared Memory Support (Continued)



TL/DD/8340–28

FIGURE 24. Shared Memory Application: HPC16040 Interface to Series 32000 System

TABLE IV. Memory Map

FFFF:FFF0	Interrupt Vectors	USER MEMORY
FFEF:FFD0	JSRP Vectors	
FFCF:FFCE		
: :	On-Chip ROM	
F001:F000		
EFFF:EFFE		
: :	External Expansion	
0201:0200	Memory	
01FF:01FE		
: :	On-Chip RAM	
01C1:01C0		
0195:0194	Watchdog Address	Watchdog Logic
0192	T1CON 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	
0153:0152	Port P Register	Timer Block T4:T7
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	
00E7:00E6	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	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)	
00C1:00C0	PSW Register	
00BF:00BE	On-Chip RAM	USER RAM
: :		
0001:0000		

5

HPC16040 CPU

The HPC16040 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 HPC16040 (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
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
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		
SET	Set bit	1 → Mem.bit (bit is 0 to 7 immediate)
RESET	Reset bit	0 → Mem.bit
IF	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	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
SET C	Set C	1 → C
RESET C	Reset C	0 → C
IF C	IF C	Do next if C = 1
IFN C	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 +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
MemI is 8- or 16-bit memory or 8 or 16-bit immediate data
imm is 8-bit or 16-bit immediate data

5

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
SET	1	2	3(4)	3	4(5)	1
RESET	1	2	3(4)	3	4(5)	1
IF	1	2	3(4)	3	4(5)	1
DDSZ	3	3	2(4)	3	4(5)	
INCD	3	3	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
SET	C	1
RESET	C	1
IF	C	1
IFN	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 HPC16040 has been designed to be extremely code-efficient. The HPC16040 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 HPC16040, and the code savings over other popular microcontrollers has been considerable—often the jobs take less than one-half the memory!

Reasons for this saving of code include the following:

SINGLE BYTE INSTRUCTIONS

The majority of instructions on the HPC16040 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 HPC16040 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 the B register
3. Compare the B register versus the K register
4. Generate a conditional skip if B is greater than 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 HPC16040 supplies 8-bit byte capability for 2-digit variables and literal variables.

MULTIPLY AND DIVIDE INSTRUCTIONS

The HPC16040 has 16-bit multiply and 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

The MOLE (Microcontroller On-Line Emulator) is a low cost development system and emulator for all microcontroller products. These include COPs, TMP, 8050U and the HPC Family of Products. The MOLE consists of a BRAIN Board, Personality Board and optional host software.

The purpose of a MOLE is to provide the user with a tool to write and assemble code, emulate code for the target microcontroller and assist in both the software & 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 because multiple ports are usually needed to optionally connect to a terminal, a host system, a printer or 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 selected host systems, i.e., those using CP/M or PC-DOS. Communicating via RS-232 port.

Dial-A-Helper is a service provided by the MOLE applications group. If a user is having difficulty in getting a MOLE to operate in a particular mode or it is acting peculiar, he can contact us via his system and a modem. He can leave messages on our electronic bulletin board which we will respond to, or he can arrange for us to actually take control of his system via modem for debugging purposes.

The applications group can then cause his system to execute various commands and try to resolve the customer's problem by actually getting the customer's system to respond. 99% of the time the problem is resolved. This allows us to respond in minutes instead of days when applications help is needed.

National Semiconductor Corporation

PRELIMINARY

NS405/NS32405-Series Display/Terminal Management Processor (TMP)

General Description

The NS405 is a CRT terminal controller on a chip. It is a microcomputer system which replaces the following LSI circuits commonly found in a CRT data terminal:

- Microcomputer
- Baud Rate Generator
- CRT Controller
- Interrupt Controller
- DMA Controller
- Parallel I/O Controller
- Character Generator
- Timer
- UART

In addition the NS405 includes powerful attribute logic, two graphics display modes, and fast video output circuits.

The NS405 is primarily intended for use in low-cost terminals, but contains many features which make it a superior building block for "smart" terminals and word processing systems.

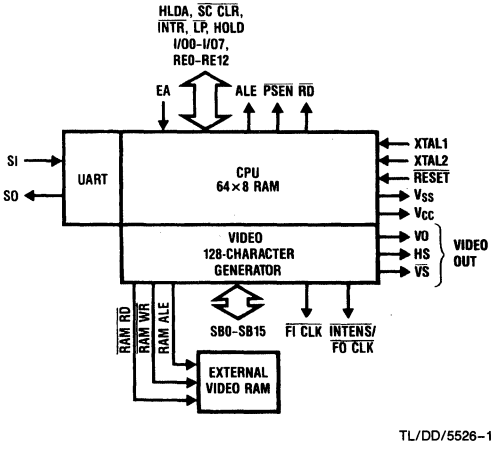
The NS405 interfaces easily to the display monitor, keyboard, display memory, and I/O ports. The architecture and instruction set are derived from the 8048-series microcontrollers. The instruction set has been enhanced and the architecture tailored to allow the NS405 CPU to efficiently manage a large display memory and an extensive interrupt environment.

The TMP can be used to easily and inexpensively add a display to many systems where it was previously impractical, it is not limited to terminal applications.

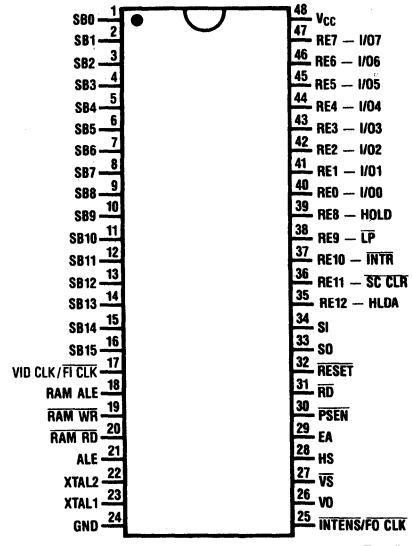
Features

- Enhanced 8048 instruction set and architecture
- Up to 8k x 8 ROM external with ROM expand bus
- On-board RAM 64 x 8
- Programmable display format
- On-board video memory management unit
- 16-bit bidirectional display memory bus (direct video and attribute RAM interface)
- Built-in timer
- Real-time clock (may be programmed for 1 Hz)
- Video control signals
- Eight independent attributes
- Pixel and block graphics display modes
- Programmable cursor characteristics
- Programmable CRT refresh rate
- Light pen feature
- UART, programmable baud rate up to 19.2k baud
- Character generator (128 characters 7 x 11 max)
- Single 5-volt supply @ 110 mA (typ)
- Up to 18 MHz video dot rate (12 MHz CPU clock)
- 48-pin package
- 8-bit parallel I/O port (multiplexed with external ROM)
- Extensive I/O expansion capabilities

Block and Connection Diagrams



TL/DD/5526-1



Top View

TL/DD/5526-2

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}*	−0.5V to +7.0V
Power Dissipation	1.5W

*EA, SI and VSYNC may be subjected to V_{SS} + 15V.

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operations should be limited to those conditions specified under DC Electrical Characteristics.*

ESD rating is to be determined.

DC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
V_{IL1}	Input Low Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)		−0.5		0.8	V
V_{IH1}	Input High Voltage (All Except XTAL1, XTAL2, $\overline{RESET}$)		2.0		V_{CC}	V
V_{IL2}	Input Low Voltage (XTAL1, XTAL2, $\overline{RESET}$)		−0.5		0.6	V
V_{IH2}	Input High Voltage (XTAL1, XTAL2, $\overline{RESET}$)		3.8		V_{CC}	V
V_{OL}	Output Low Voltage (All Except INTENS, VO)	I_{OL} = 2.0 mA			0.4	V
V_{OH}	Output High Voltage (All Except INTENS, VO)	I_{OH} = −125 μA	2.4		V_{CC}	V
V_{OL}	Output Low Voltage (INTENS, VO)	I_{OL} = 5.0 mA			0.4	V
V_{OH}	Output High Voltage (INTENS, VO)	I_{OH} = −500 μA	2.4			V
I_{IL}	Input Leakage Current (EA, $\overline{INT}$, SI)	$V_{SS} \le V_{IN} \le V_{CC}$			±10	μA
I_{OL}	Output Leakage Current (ROM Expand Bus, High Impedance State)	$V_{CC} \ge V_{IN} \ge V_{SS}$ + 0.45			±10	μA
I_{OL}	Output Leakage Current (System Bus, High Impedance State)	$V_{CC} \ge V_{IN} \ge V_{SS}$ + 0.45			±100	μA
I_{CC}	Total Supply Current	T_A = 25°C		110	150	mA

AC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified

Symbol	Parameter	Min	Typ	Max	Units
CPU AND ROM EXPAND BUS TIMING					
F_{XTAL}	Crystal Frequency	3		18	MHz
F_{CPU}	CPU Frequency	3		12	MHz
t_{CY}	CPU Cycle Time	1.25		7.5	μs
t_{DF}	Video Dot Time	55.5		333.3	ns
t_{LL}	ALE Pulse Width (Note 1)	125			ns
t_{AL}	Address Setup to ALE (Note 1)	55			ns
t_{LA}	Address Hold from ALE (Note 1)	40			ns
t_{CC}	Control Pulse Width $\overline{PSEN}$, $\overline{RD}$ (Note 1)	250			ns
t_{DR}	Data Hold (Notes 1, 4)	0		100	ns
t_{RD}	$\overline{PSEN}$, $\overline{RD}$ to Data In (Note 1)			220	ns
t_{AD}	Address Setup to Data In (Note 1)			360	ns
t_{AFC}	Address Float to $\overline{RD}$, $\overline{PSEN}$ (Note 1)		0		ns
t_{CAF}	$\overline{PSEN}$ to Address Float (Note 1)		±10		ns
t_{DAL}	Data Setup to ALE (RE11, 12) (Note 1)	55			ns
t_{ALD}	Data Hold from ALE (RE11, 12) (Note 1)	40			ns
t_{CIS}	Control Input Setup to ALE (RE8, 9, 10) (Note 1)	240			ns
t_{CIH}	Control Input Hold from ALE (RE8, 9, 10) (Notes 1, 4)	0		125	ns

AC Electrical Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±10%, V_{SS} = 0V, unless otherwise specified (Continued)

Symbol	Parameter	Min	Typ	Max	Units
SYSTEM BUS TIMING					
t_{EL}	RAM ALE Low Time (Note 1)	250	280		ns
t_{EH}	RAM ALE High Time (Note 1)	100	120		ns
t_{AS}	Address Setup to RAM ALE (Note 1)	20	70		ns
t_{AH}	Address Hold from RAM ALE (Note 1)	10	50		ns
t_{RCY}	Read or Write Cycle Time (Note 1)		420*		ns
t_{RR}	$\overline{\text{RAM RD}}$ Width (Note 1)	210	240		ns
t_{AR}	Address Setup to $\overline{\text{RAM RD}}$ (Note 1)	80	120		ns
t_{RRD}	Data Access from $\overline{\text{RAM RD}}$ (Note 1)			140	ns
t_{RDR}	Data Hold from $\overline{\text{RAM RD}}$ (Notes 1, 4)	0		60	ns
t_{WFI}	$\overline{\text{FIFO In Clock}}$ Width (Note 1)	210	240		ns
t_{WW}	$\overline{\text{RAM WR}}$ Strobe Width (Note 1)	140	160		ns
t_{AW}	Address Setup to $\overline{\text{RAM WR}}$ (Note 1)	120	200		ns
t_{DW}	Data Setup to $\overline{\text{RAM WR}}$ (Note 1)	10	40		ns
t_{WD}	Data Hold from $\overline{\text{RAM WR}}$ (Note 1)	20	40		ns
VIDEO TIMING					
t_{DF}	Dot Period = $\frac{1}{f_c}$ (Note 1)		55		ns
t_{VID}	Video Blank Time $\frac{t_{DF}}{5}$ (Note 1)		±10		ns
t_{VI}	Skew, $\overline{\text{Intensity}}$ to Dot 0 (Note 1)		±15		ns
t_{FOV}	$\overline{\text{FIFO Out Clock}}$ to Dot 0 (Note 1)		15		ns
t_{WFOH}	$\overline{\text{FIFO Out Clock}}$ Width High (Note 1, Note 2)		55–165**		ns

*1/3 CPU cycle.

**1 Dot time is 55 ns.

Note 1: Control outputs C_L = 80 pF; ROM Expand Bus outputs C_L = 150 pF; System Bus outputs C_L = 100 pF; F_{XTAL} = 18 MHz; F_{CPU} = 12 MHz.

Note 2: $\overline{\text{FO CLK}}$ duty cycle is shown above.

Note 3: Hold request is latched. It is honored at the start of the next vertical retrace.

Note 4: Max spec. listed for user information only, to prevent bus contention.

Character Cell Width	$\overline{\text{FIFO Out}}$ HIGH	$\overline{\text{FIFO Out}}$ LOW
6	1 dot	5 dots
7	2 dots	5 dots
8	2 dots	6 dots
9	3 dots	6 dots
10	3 dots	7 dots

Input Hold Times

T_A = 25°C, V_{CC} = +5V ±10%, V_{SS} = 0V

Input	Min Active Time
Reset	50 ms (power up) 5 CPU Cycles (after power up)
$\overline{\text{External Interrupt}}$	2 CPU Cycle
$\overline{\text{Light Pen}}$	1 CPU Cycle
I/O Input	1 CPU Cycle
Hold Request	1 CPU Cycle (Note 3)

FIFO

Fall through should not be greater than 4 character times (character time = $1/f_{XTAL} \times$ #dots/cell).

Throughput rate must be at least the character rate (character rate = 1/character time).

Capacitance $T_A = 25°C, V_{CC} = V_{SS} = 0V$

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
C_{IN}	Input Capacitance	$F_C = 1$ MHz		6	10	pF
C_{OUT}	Output and Reset	Unmeasured Pins Returned to V_{SS}		10	20	pF

AC Electrical Characteristics in CPU Cycle Time

CPU AND ROM EXPAND BUS TIMING

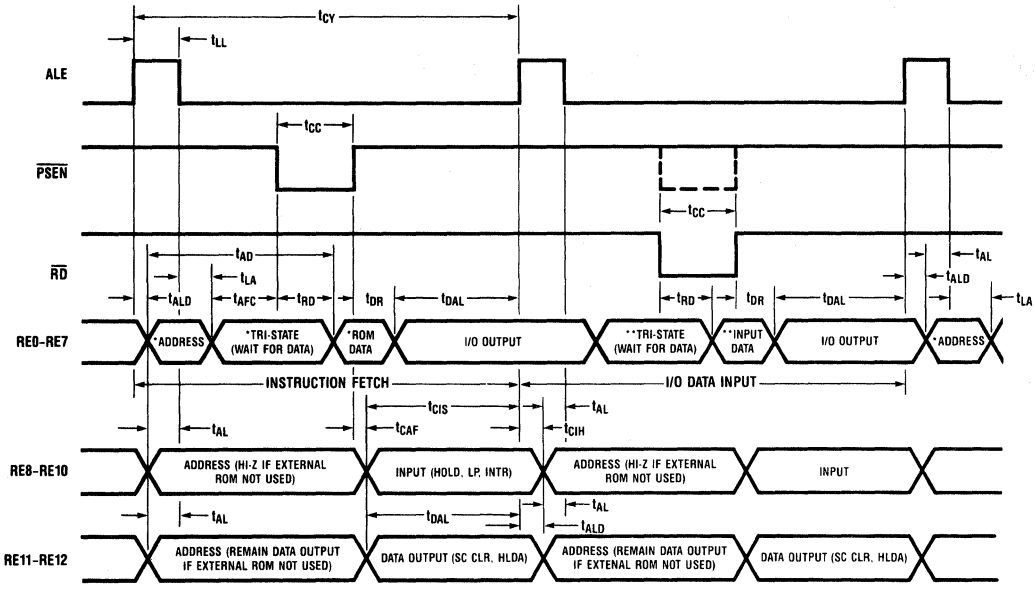
Symbol	Parameter		Typ
t_{LL}	ALE Pulse Width		$14\ t_{CY/60}$
t_{AL}	Address Setup to ALE		$8\ t_{CY/60}$
t_{LA}	Address Hold from ALE		$6\ t_{CY/60}$
t_{CC}	Control Pulse Width	$\overline{PSEN}$ $\overline{RD}$	$24\ t_{CY/60}$ $36\ t_{CY/60}$
t_{CY}	CPU Cycle Time		$60\ t_{CY/60} = 15/f_{CPU} = \frac{15}{f_{XTAL} \div 1 \text{ or } \div 1.5}$
t_{DR}	Data Hold		$-2\ t_{CY/60}$
t_{RD}	Control Pulse to Data In	$\overline{PSEN}$ $\overline{RD}$	$18\ t_{CY/60}$ $30\ t_{CY/60}$
t_{AD}	Address Setup to Data In		$32\ t_{CY/60}$
t_{AFC}	Address Float to	$\overline{PSEN}$ $\overline{RD}$	$2\ t_{CY/60}$ $2\ t_{CY/60}$
t_{CAF}	PSEN to Address Float		$0\ t_{CY/60}$
t_{DAL}	Data Setup to ALE	RE0–7 RE8–10 RE11–12	$6\ t_{CY/60}$ $-2\ t_{CY/60}$ $16\ t_{CY/60}$
t_{ALD}	Data Hold from ALE	RE0–7 RE8–12	$2\ t_{CY/60}$ $6\ t_{CY/60}$

SYSTEM BUS TIMING

Symbol	Parameter	Ticks		
		Min	Typ	Max
t_{EL}	RAM ALE Low Time	$14\ t_{CY/60} - 42$ ns	$14\ t_{CY/60}$	
t_{EH}	RAM ALE High Time	$6\ t_{CY/60} - 25$ ns	$6\ t_{CY/60}$	
t_{AS}	Address Setup to RAM ALE	$4\ t_{CY/60} - 60$ ns	$4\ t_{CY/60}$	
t_{AH}	Address Hold from RAM ALE	$2\ t_{CY/60} - 40$ ns	$2\ t_{CY/60}$	
t_{RCY}	Read or Write Cycle Time		$20\ t_{CY/60}$	
t_{RR}	$\overline{RAM\ RD}$ Width	$12\ t_{CY/60} - 40$ ns	$12\ t_{CY/60}$	
t_{AR}	Address Setup to $\overline{RAM\ RD}$	$6\ t_{CY/60} - 45$ ns	$6\ t_{CY/60}$	
t_{RRD}	Data Access from $\overline{RAM\ RD}$		$10\ t_{CY/60}$	$10\ t_{CY/60} - 70$ ns
t_{RDR}	Data Hold from $\overline{RAM\ RD}$			
t_{WFI}	$\overline{FIFO\ In\ Clock}$ Width	$12\ t_{CY/60} - 40$ ns	$12\ t_{CY/60}$	
t_{WW}	$\overline{RAM\ WR}$ Strobe Width	$8\ t_{CY/60} - 27$ ns	$8\ t_{CY/60}$	
t_{AW}	Address Setup to $\overline{RAM\ WR}$	$10\ t_{CY/60} - 90$ ns	$10\ t_{CY/60}$	
t_{DW}	Data Setup to $\overline{RAM\ WR}$	$2\ t_{CY/60} - 30$ ns	$2\ t_{CY/60}$	
t_{WD}	Data Hold from $\overline{RAM\ WR}$	$2\ t_{CY/60} - 20$ ns	$2\ t_{CY/60}$	

Timing Waveforms

ROM Expand Bus Timing
(In Port Instruction is Shown)

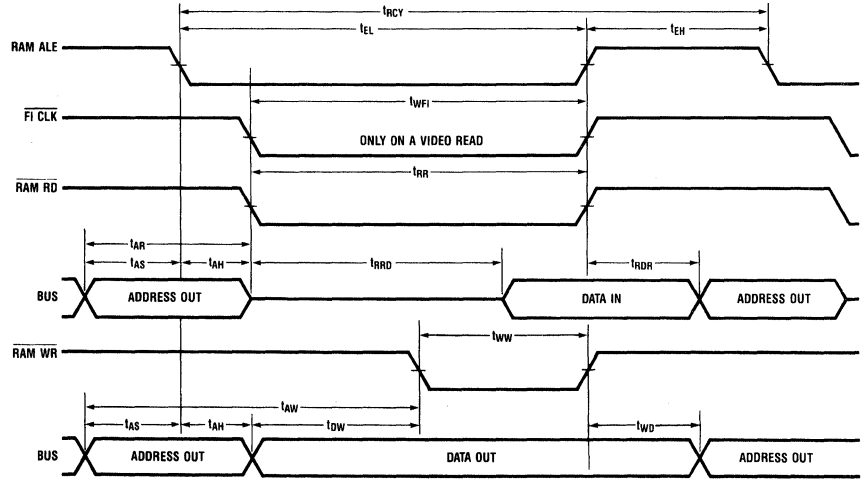


TL/DD/5526–3

*Remain I/O OUTPUT if External ROM not used.

**I/O Data input or 2nd ROM byte of 2 byte instruction. Otherwise remain I/O OUTPUT.

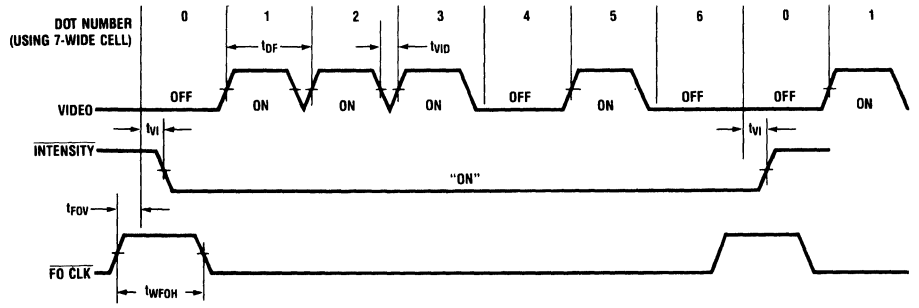
System Bus Timing



TL/DD/5526–4

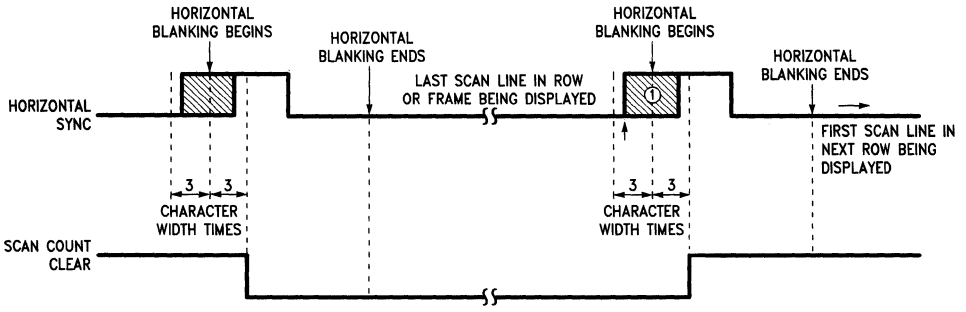
Timing Waveforms (Continued)

Video Timing



TL/DD/5526–5

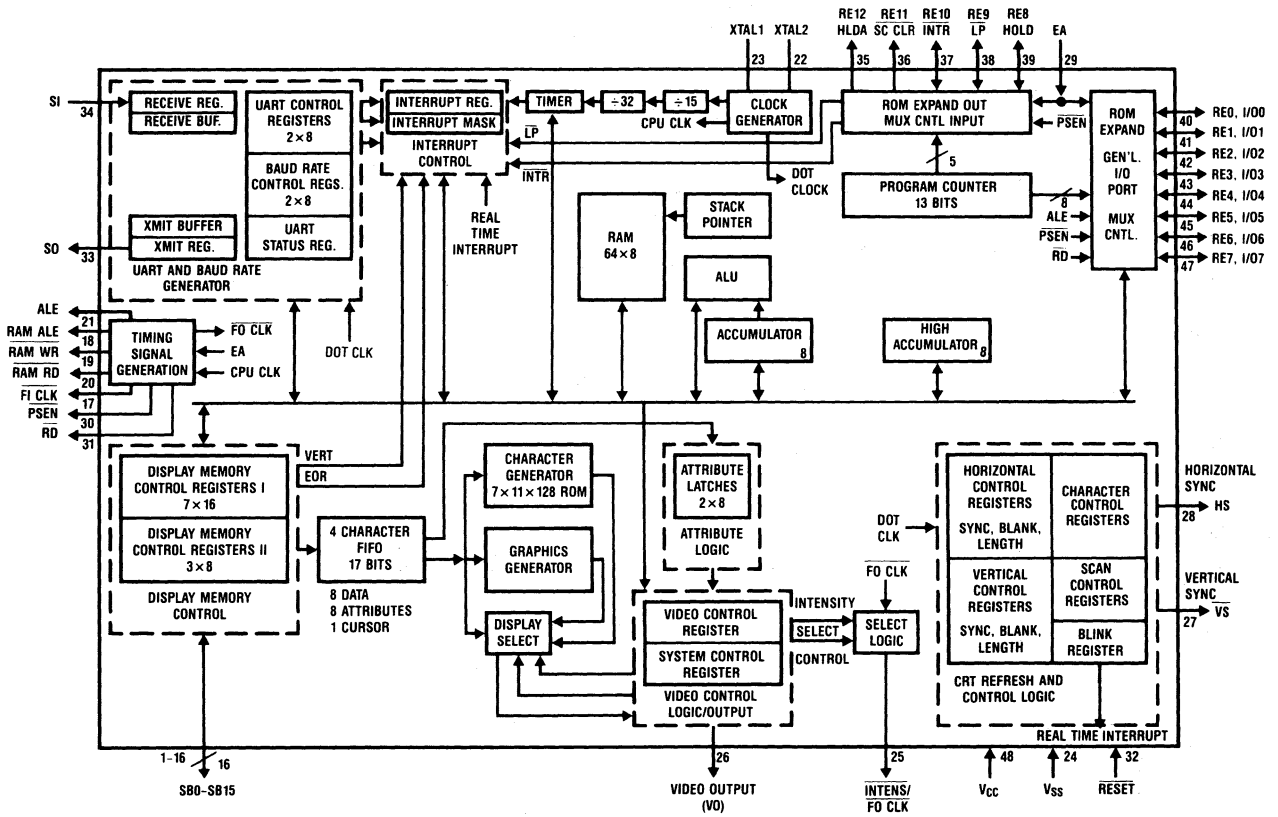
Scan Count Clear Timing



TL/DD/5526–6

For external character generation this edge is used to clock CLEAR into scan line counter. The edge must come before Scan Count Clear goes away, but not before the video controller has brought in all necessary display information for the last scan line.

NS405-Series Detailed



TL/DD/5526–7

1.0 Functional Pin Descriptions

1.1 SUPPLIES

Pin	Name	Function
48	V_{CC} — Power	5V ±10%
24	V_{SS} — Ground Reference	

1.2 INPUT SIGNALS

23, 22	XTAL1, XTAL2 — Crystal 1, 2:	Crystal connections for clock oscillator (3–18 MHz).
29	EA — External Access:	Pull HIGH (V_{IH_2})
32	$\overline{\text{RESET}}$	An active low input that initializes the processor. The $\overline{\text{RESET}}$ input is also used for internal ROM verification.
34	SI — Serial Input:	Drives receiver section of UART (true data).

1.3 OUTPUT SIGNALS

33	SO — Serial Output:	Driven by transmitter section of UART (true data).
21	ALE — Address Latch Enable:	ROM address is available on the ROM Expand Bus and may be latched on the falling edge of ALE. Port output data may be latched on the rising edge of ALE. ALE pulses are always present, even if EA is tied low.
30	$\overline{\text{PSEN}}$ — Program Store Enable:	Enable external ROM output drivers when low. $\overline{\text{PSEN}}$ is idle (high) when the CPU fetches from internal ROM.
31	$\overline{\text{RD}}$ — Read Port Data:	Accept Port input data on ROM Expand Bus RE0–RE7 while low. ROM Expand Bus is in high impedance state while $\overline{\text{RD}}$ is low.
28	HS — Horizontal Sync	The rising edge of HS is controlled by the Horizontal Sync Begin Register and the falling edge is controlled by the Horizontal Sync End Register. HS is disabled (low) if bit 5 of the Video Control Register = 0.
27	$\overline{\text{VS}}$ — Vertical Sync Output:	The falling edge of $\overline{\text{VS}}$ is controlled by the Vertical Sync Begin Register and the rising edge is controlled by the Vertical Sync End Register. $\overline{\text{VS}}$ is tri-stated if bit 5 of the Video Control Register = 0.
26	VO — Video Output:	High = beam on, low = beam off. VO is disabled (low) if bit 5 of the Video Control Register = 0.
25	$\overline{\text{INTENS}}$/$\overline{\text{FO CLK}}$	(Shared pin) $\overline{\text{INTENS}}$ Signal under attribute control may be used to switch the bistable brightness of display characters. $\overline{\text{FIFO Out Clock}}$ may be used to clock data from an external FIFO in synchronism with data from the internal FIFO. Both CANNOT be used simultaneously.
17	VID CLK/$\overline{\text{FI CLK}}$ — Video Dot Clock Out/ FIFO IN CLOCK	(Shared pin) The rising edge of the Video Dot Clock may be used to clock the data out of the video output pin. FIFO In Clock may be used to clock data from an extended attribute RAM into an external FIFO in synchronism with the data loaded into the internal FIFO. Both CANNOT be used simultaneously.
18	RAM ALE — RAM Address Latch Enable:	RAM address is available on the System Bus and may be latched on the falling edge of RAM ALE. Only operational when Display RAM accesses being performed. Otherwise high.
20	$\overline{\text{RAM RD}}$ — RAM Read:	Enable display RAM data onto the System Bus when $\overline{\text{RAM RD}}$ is low.
19	$\overline{\text{RAM WR}}$ — RAM Write:	Data to RAM is available on the System Bus and may be written at the rising edge of $\overline{\text{RAM WR}}$.

1.4 BUS — I/O

1–8	SB0–SB7 — System Bus 0–7:	Display RAM address is output while RAM ALE is high and may be latched on the falling edge of RAM ALE. System Bus accepts data input while $\overline{\text{RAM RD}}$ is low and outputs data while $\overline{\text{RAM WR}}$ is low.
9–16	SB8–SB15 — System Bus 8–15:	Normally, Display RAM address is output and held on these pins for the full read or write cycle. However, if bit 4 of the System Control Register is set, these pins function bidirectionally like SB0–SB7 to allow 16-bit data words for attribute operation.
35–47	RE0–12 — ROM Expand Bus 0–12:	Used for program ROM expansion as described below. Time multiplexed with I/O port and system control signals. I/O port and system control signals only if no external ROM used.
40–47	RE0–RE7	Low order ROM address is output and may be latched on the falling edge of ALE. Enable ROM data to this Bus when $\overline{\text{PSEN}}$ is low. Enable I/O port input data to the Bus when $\overline{\text{RD}}$ is low. Use the rising edge of ALE to latch port output data.

1.0 Functional Pin Description (Continued)

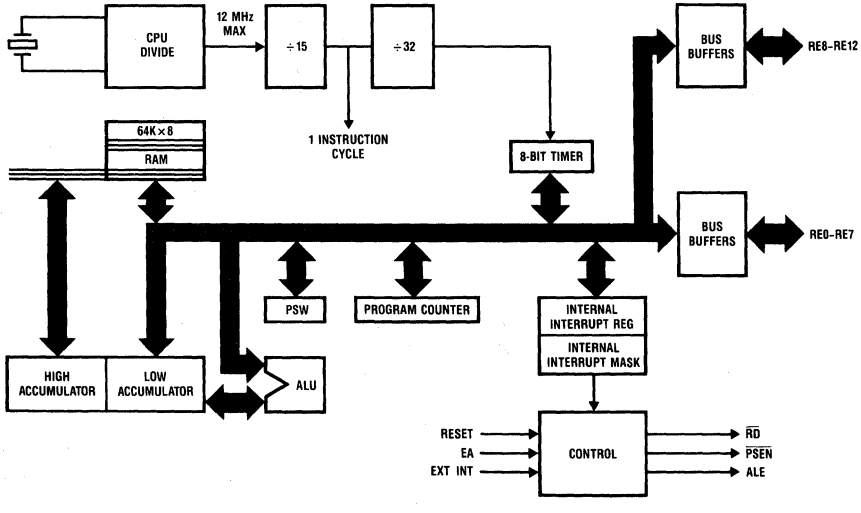
Pin	Name	Function
39–35	RE8–RE12	Five most significant bits of the ROM address are output during ALE and remain stable until data is read in during $\overline{PSEN}$. These pins are multiplexed with the HLDA, $\overline{INTR}$, $\overline{LP}$, $\overline{SC\ CLR}$, and HOLD signals.
37	$\overline{INTR}$ — Interrupt: RE10	An active low input that interrupts the processor if the external interrupt is enabled. Because it shares a pin with RE10, $\overline{INTR}$ may be driven directly only if no external ROM is used (EA is low). Otherwise must be driven through a 3.9k resistor.*
38	$\overline{LP}$ — Light Pen Interrupt: RE9	An active low input that interrupts the processor if internal interrupts are enabled and bit 5 in the Interrupt Mask Register is set. Because it shares a pin with RE9, $\overline{LP}$ may be driven directly only if EA is low. Otherwise, must be driven through a 3.9k resistor.*
39	HOLD — HOLD request: RE8	When high, requests that the NS405 enter the Hold mode. When in the Hold mode the System Bus will be in a high impedance state. The Hold mode is granted at the beginning of the next vertical retrace. Because it shares a pin with RE8, HOLD may be driven directly only if EA is low. Otherwise, must be driven through a 3.9k resistor.*
35	HLDA — Hold Acknowledge: RE12	This output is asserted in response to Hold and provides handshake capability with another processor (active high). For more detailed information see Section 3.1 Slave Processing. Because HLDA shares a pin with RE12, the HLDA state is preset only during the interval preceding the rising edge of ALE. However, if no external ROM is used, HLDA is a steady state output and need not be latched externally.
36	$\overline{SC\ CLR}$ — Scan Count Clear: RE11	This output clears an external scan counter when used with an external character generator. It is a low going pulse which occurs during the horizontal retrace preceding the first scan line of each character row. Because $\overline{SC\ CLR}$ shares a pin with the RE11, the correct $\overline{SC\ CLR}$ state is present only during the interval preceding the rising edge of ALE. However, if no external ROM is used, $\overline{SC\ CLR}$ is a steady state output and need not be latched externally.

*Unused control inputs must be terminated

2.0 Functional Description

2.1 CPU

The CPU of the NS405 is patterned after the 8048 single chip microcomputer (see *Figure 1).*



TL/DD/5526–8

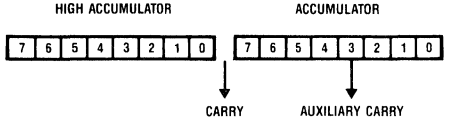
FIGURE 1. NS405 Series CPU Block Diagram

2.0 Functional Description (Continued)

2.1.1 Accumulator — High Accumulator

In addition to the regular 8-bit Accumulator, there is an 8-bit High Accumulator extension to facilitate the 16-bit operations required for display memory management. The HACC/ACC pair is usually used in conjunction with the 16-bit RAM pointer registers (RA, R0 and RB, R1, CURSOR, HOME, BEGD and ENDD) to effect video data transfers. In addition, external attribute memory is loaded in a 16-bit transfer operation. Any instruction which causes a carry or borrow out of the low accumulator will affect the high accumulator (see *Figure 2*).

Auxiliary carry is used only when converting the accumulator contents from binary to BCD (binary coded decimal) using the DA A instruction. The auxiliary carry flag can be cleared by moving a zero into bit 6 of the program status word.

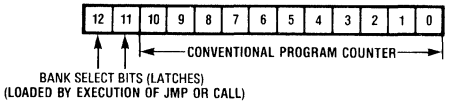


TL/DD/5526-9

FIGURE 2. CPU Accumulator

2.1.2 Program Counter (PC)

The Program Counter is a 13-bit wide register which provides program addressing for the CPU. The lower 11 bits operate like a conventional program counter while the upper 2 bits are actually latches. These 2 latches are automatically loaded from the bank select flip-flops (PSW bits 3, 4) whenever a JMP or CALL instruction is executed. The bank select flip-flops in turn are only modified upon the execution of a Select Memory Bank Instruction or modification of the PSW (see *Figure 3*).



TL/DD/5526-10

FIGURE 3. TMP Program Counter

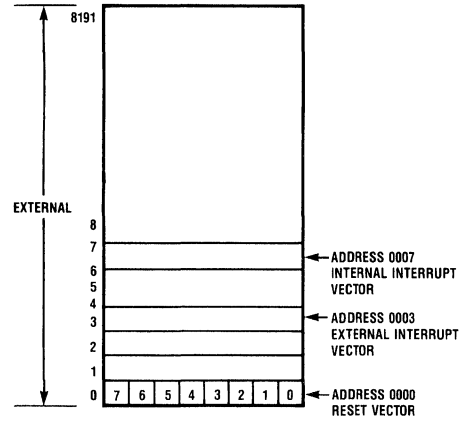
2.1.3 Program Memory

Memory is subdivided into 2k banks with accesses limited to the currently selected bank unless a Bank Change sequence has been executed. Upon reaching the end of a memory bank, the program counter will wrap around and point to the beginning of the current bank.

Each bank is further subdivided into pages of 256 bytes each, with 8 pages in every bank. The conditional JUMP instructions are restricted to operate within the memory page that they reside in.

Because of the sequence which the CALL instruction executes when pushing and loading the PC, it is possible to easily call and return from subroutines located in different memory banks (see *Figure 4*).

Upon executing an RET or RETR instruction for a call from one memory bank into another, a SEL MBx instruction should be excuted to restore the memory bank select flip-flops to their original bank. However, no SEL MBx is needed after an interrupt since the flip-flops were never modified.



TL/DD/5526-11

FIGURE 4. Program Memory Map

2.1.4 Program Status Word Bit Assignments

Bit Position	Contents
0	Stack Pointer Bit, S0
1	Stack Pointer Bit, S1
2	Stack Pointer Bit, S2
3*	Memory Bank Select Bit 0
4*	Memory Bank Select Bit 1
5*	Register Bank Select Bit (0 = Bank 0, 1 = Bank 1)
6*	Auxiliary Carry. A carry from Bit 3 to Bit 4 generated by an add operation. Used only by the decimal adjust (DA A) instruction.
7*	Carry. A bit indicating the preceding operation resulted in an overflow or an underflow from the 8-bit accumulator.

***Note 1:** Bits 3 through 7 are saved on the stack by subroutine calls or interrupts. Bits 3 and 4 are restored upon execution of an RET instruction, whereas all 5 bits are restored by RETR.

Note 2: F0 is not saved on the stack (as in an 8048).

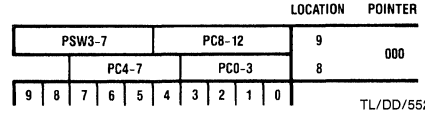
Note 3: Bits 0–5 cleared on a RESET.

2.1.5 Stack Pointer (SP)

The stack pointer is an independent 3-bit counter which points to designated locations in the internal RAM that holds subroutine return parameters. The stack itself is located in RAM locations 8–23 (see *Figure 5*).

Each entry in the stack takes up two bytes and contains both the PC and status bits. When reset to zero, the stack pointer actually points to locations 8 and 9 in RAM. Since the stack pointer is a simple up/down counter, an overflow will cause the deepest stack entry to be lost (the counter overflows from 111 to 000 and underflows from 000 to 111).

Note: If the level of subroutine nesting is less than eight (8), the unneeded stack locations may be used as RAM.



TL/DD/5526-12

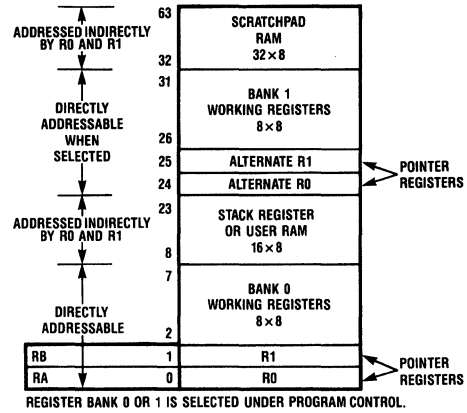
Note: The odd numbered RAM bytes in the stack area have two (2) extra bits to allow for storage of the bank select switch bits. This feature allows interrupt routines and subroutines to be located outside the current 2k program memory bank.

FIGURE 5. Typical Stack Composition

2.0 Functional Description (Continued)

2.1.6 Data Memory (On-Chip RAM)

The data memory nominally consists of 64 8-bit locations and is utilized for working registers, the subroutine stack, pointer registers and scratch pad. There are two sets of working/pointer registers (R0–R7) which are selected by the Select RAM Bank instruction. The stack area is located in locations 8–23. Locations 32–63 contain the scratch pad memory. To facilitate 16-bit Video Memory Management there are two 8-bit extension registers (RA and RB) which are associated with the R0 and R1 registers respectively of whichever RAM bank is currently selected (see *Figure 6*). i.e., There is only one RA register and only one RB register.

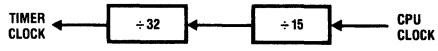


TL/DD/5526–13

FIGURE 6. RAM Memory Map

2.1.7 Timer

The On-Board Timer is an 8-bit up counter which sets the Timer Overflow Flag and generates an internal interrupt (if enabled) whenever it overflows from FF to zero. The Timer may be stopped, started, loaded and read from by the CPU. The Timer clock is derived from the CPU clock as shown in *Figure 7*. Whenever a Start Timer instruction is executed the ÷32 is initialized to its zero state to insure a full count measurement. After overflow the timer keeps counting until the next FF to zero overflow at which time the overflow flag will be set and another interrupt generated. The overflow flag can only be reset through the JTF and JNTF instructions.



TL/DD/5526–14

FIGURE 7. Timer Clock Generation

2.1.8 Interrupts

The interrupt circuitry handles two generic classes of interrupt conditions called Internal and External. Either class has its own master control which can be activated through software enable and disable instructions. On an interrupt service the currently executing instruction is completed, then two CPU cycles are used as the program counter and bits 3–7 of the PSW are pushed onto the stack and stack pointer is incremented.

Then the interrupt vector address (3 or 7) is loaded into the PC and service started. Whenever an interrupt condition is being serviced all other interrupts of either class are locked out until a RETR instruction is executed to conclude interrupt service. If both an external and internal interrupt arrive at the same time, the external interrupt is recognized first.

2.1.8.1 External Interrupt

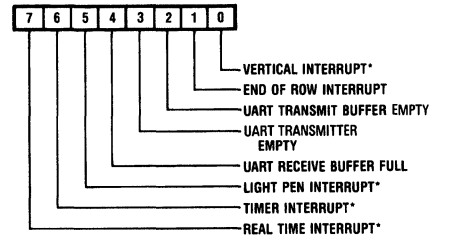
The External Interrupt consists solely of the shared $\overline{\text{INTR}}$/RE10 pin. External interrupts on this pin will be detected if the setup and hold times as shown in the timing diagrams are met. This pin is a level sampled interrupt which means that as long as the pin is low during the sampling window an interrupt will be generated. In addition, the $\overline{\text{INTR}}$ pin is the only external pin whose logic state can be directly tested through software.

2.1.8.2 Internal Interrupts

The Internal Interrupts consist of seven internal operational conditions plus the light pen arranged in an 8-bit wide register as shown in *Figure 8*. Activation of an internal interrupt condition causes a corresponding register bit to be set, *Figure 9*. Each internal interrupt may be individually masked out through the Interrupt Mask register which has the same bit assignments as the Interrupt register and can be loaded from the accumulator. A zero in the Interrupt Mask register inhibits the interrupt and a one enables it. Further interrupt processing is as shown. To determine which of the eight internal conditions caused the interrupt the CPU must read the Interrupt register into the accumulator. To acknowledge receipt of the interrupt certain bits are automatically cleared on a read while others are reset upon service of the particular interrupt. The conditions under which each of the interrupts are generated and cleared are as follows:

Bit

0 Vertical Interrupt—Generates an interrupt at the end of the display row designated by the Vertical Interrupt Register. Interrupt bit cleared on a CPU read of the interrupt register. If VIR > Vertical Length Register no interrupt will be generated.

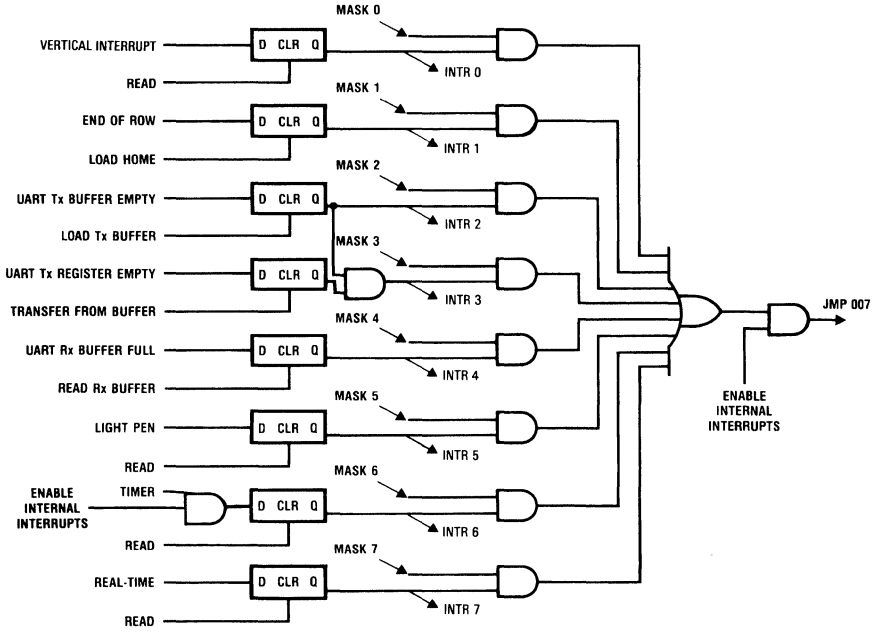


TL/DD/5526–16

Note: The interrupt flags indicated by an asterisk (*) are cleared when the Interrupt Register is read.

FIGURE 8. Internal Interrupt Register

2.0 Functional Description (Continued)



TL/DD/5526–15

FIGURE 9. Internal Interrupt Processing

Bit

1 End of Row Interrupt—Generates an interrupt at the end of each display row when the Current Row Start Register is updated for the next row. Used in conjunction with the Row Sequencing Control Bit (5) in the System Control Register to implement Row Pointer Look-Up Tables and Horizontally Split Screens. Interrupt bit cleared on a CPU write to the Home Register. Does not generate interrupts for those rows blanked during vertical blanking.

2 UART Transmit Buffer Empty—Generates an interrupt when the Transmit Buffer empties out after dumping a character into the Transmit Shift Register. Interrupt bit cleared on a CPU write to the Transmit Buffer.

3 Transmitter Empty—Generates an interrupt when BOTH the Transmit Buffer and Transmit Shift Register are empty. The interrupt bit is cleared when the CPU loads the transmit buffer.

4 UART Receiver Buffer Full—Generates an interrupt when the Receiver Buffer fills up with a character from the Receive Shift Register. Interrupt bit cleared on a CPU read of the Receiver Buffer.

5 Light Pen Interrupt—Generates an interrupt on each falling edge detected on the shared $\overline{LP}$/RE9 pin. Since only falling edges generate interrupts and the input is sampled each CPU Cycle, a high level must be sampled between falling edges in order to be considered a new interrupt. This interrupt is used to latch the light pen position registers. For further information see Light Pen Description. Interrupt bit cleared on a CPU read of the interrupt register.

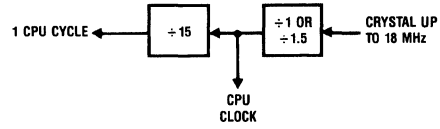
Bit

6 Timer Interrupt—Generates an interrupt when the internal 8-bit Timer overflows from FF to 00. Interrupt bit cleared on a CPU read of the interrupt register.

7 Real-Time Interrupt—Generates interrupts at a software programmable frequency that is generally in the Hertz range. (See CPU Clock Generation.) Thus permitting the implementation of a real-time clock or timer. Interrupt bit cleared on a CPU read of the interrupt register.

2.1.9 Clock Generation

All chip clocks are derived from the one external crystal connected between pins 22 and 23. This master clock also doubles as the video dot clock. The crystal frequency is constrained to lie within the range of 3 to 18 MHz. The CPU clock is derived from the crystal clock by either using it directly or by dividing down by a factor of 1.5 *(Figure 10)*.



TL/DD/5526–17

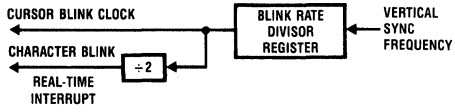
FIGURE 10. CPU Clock Generation

The choice is software programmable through bit 0 in the System Control Register. The exact selection is made in consideration of the fact that the CPU clock must lie within the range of 3 to 12 MHz. In addition, the choice of divide by modes will also impact the display character cell width due to the nature of the video controller. Specifically with ÷1.5

2.0 Functional Description (Continued)

the cell width must be ≥ 8 dots wide whereas with ÷1 the cell width must be ≥ 6 dots wide.

The low clock rates necessary to implement Cursor Blinking, Character Blinking and the Real-Time Interrupt are derived by passing the vertical sync frequency through a 5-bit Blink Rate Divisor Register, *(Figure 11)*. The resultant frequency is used as the Cursor Blink Clock. This clock is then further divided by 2 to yield the Character Blink and Real-Time Interrupt Clocks. For example, to get a 1 Hz real time interrupt, with a 60 Hz system, set the 5 bit Divisor Register to 30 in order to yield a 2 Hz signal which is then divided by 2.

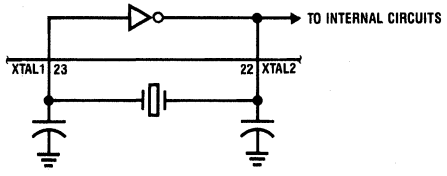


TL/DD/5526–18

FIGURE 11. Blink Clock Generation

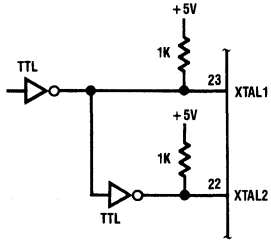
2.1.10 Oscillator Operation

The on-board oscillator circuit consists of a phase inverter which, when used with an external parallel resonant tank, *(Figure 12a)*, will yield the required oscillator clock. Crystals should be specified for AT cut and parallel resonant operation with the desired load capacitance (typically 20 pF). If one desires to externally generate the clock and input it to the chip, he may do so by driving XTAL1 (pin 23) and XTAL2 (pin 22) as shown in *Figure 12b*.



TL/DD/5526–19

FIGURE 12a. TMP Oscillator



TL/DD/5526–20

Note: Use AS TTL devices if faster than 12 MHz.

FIGURE 12b. External Oscillator Mode

2.2 DISPLAY MEMORY CONTROLLER

The video display data resides in the external Video Memory which is managed by the Display Memory Controller (DMC) through the System Bus. Either the CPU or the Video Controller may access the display memory by presenting its requests to the DMC. A maximum of three Video Memory accesses (Reads or Writes) can be performed by the DMC during each CPU instruction execution cycle. Because the CPU can access the Video Memory, one may expand CPU I/O or data memory by memory mapping into the Video Memory space. Up to 64k locations may be addressed over the 16-bit System Bus. Data word widths may be 8 or 16 bits depending upon whether external character attribute selection is used. The actual bus multiplexing mode is controlled by bit 4 in the System Control register. The Video Controller has the highest priority in obtaining Video Memory accesses with the CPU getting in on a space available basis. If all memory accesses are being taken by the Video Controller (rarely), the CPU is put into a wait state should it try to access video memory. To ease accessing requirements and boost throughput the Video Controller utilizes a 4-level data FIFO which is normally kept full of display data.

2.2.1 Display Memory Control Registers

In order to facilitate the management of video data for such features as a Screen scroll, memory paging and row lookup the DMC utilizes a number of registers which address the video RAM space. Each of these pointers is 16 bits wide and writable or readable from the 16-bit HACC/ACC pair as the case may be. There are 2 video data accessing modes as determined by bit 5 in the SCR, Sequential and Table Lookup. The functions of the pointer registers vary depending upon the accessing mode selected. Their designators are:

HOME = Home address register. Read and write.

BEGD = Beginning of diplay RAM. Write only.

ENDD = End of display RAM. Write only.

CURS = Cursor address register. Read, Write, Increment, Decrement.

SROW = Status section register. Write only.

CRSR = Current row start register. Not directly accessed.

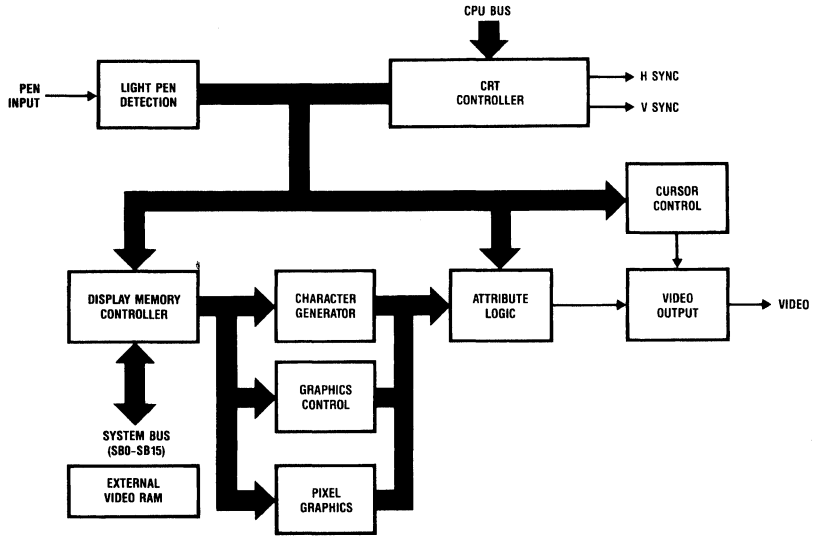
2.2.2 Sequential Access Mode

In this mode display data is accessed from sequential address locations in the video memory until the data requirements for the current screen field are fulfilled. The location from which the first display character is taken is the one pointed to by the HOME register. By modifying the contents of HOME one may implement a row scroll or paging operation. The BEGD and ENDD are used to control the wrap-around condition when HOME gets near the end of available display RAM as determined by ENDD. In this instance, when sequential accessing brings us to the end of memory as pointed to by ENDD, the controller wraps around by jumping back to the beginning of display memory as pointed to by BEGD. The value in ENDD should be the last location in display memory + 1. Also the size of the display memory between BEGD and ENDD (ENDD − BEGD) must be an integral number of display rows. The CURS in both accessing modes merely identifies the current cursor position in display memory so that the cursor characteristics can be inserted into the video at the appropriate character position.

In addition to the display of normal video data one may elect to have a special status section displayed using data from a separate section of video memory. The status section would consist of an integral number of display rows on the bottom of the screen. This feature operates by reloading the video RAM pointer with the contents of SROW when the desired row position at which to start the status section comes up. The particular row at which the status display starts is defined in the Timing Chain. Once the video RAM pointer is jumped to SROW, data accessing again proceeds sequentially from there until the data requirements for the current field are satisfied.

2.0 Functional Description (Continued)

TMP Video Section



TL/DD/5526-21

Whether a status section is used or not, upon accessing all of the data necessary to display a field, the video RAM pointer is reset to HOME in preparation for the display of a new field.

2.2.3 Table Lookup Mode

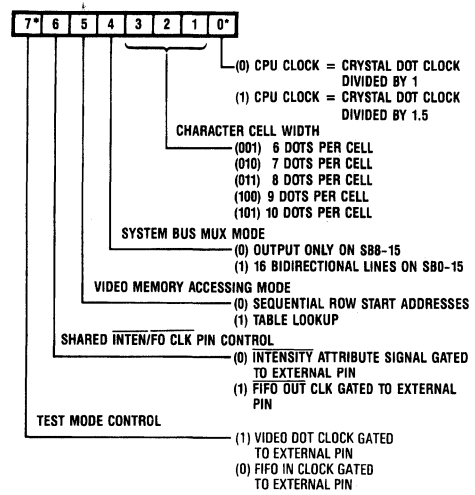
The CRSR (transparent to the user) is a pointer to the address of the first character in a display row. It is required because each time a scan line is displayed, all display characters in the row must be accessed anew. Since a row is made up of a number of scan lines, we must recover the address of the first character in the row for each scan in the row. After a row is done, the CRSR is normally advanced to point to the first character in the next row.

In table look-up mode the starting memory location of the next row is loaded into the CRSR from the HOME register at the end of each row. The HOME register was presumably updated by the CPU since the last end of row.

A CRSR load also generates the internal End of Row interrupt which the CPU will use as a signal to reload HOME. Finally, reloading HOME will clear out the End of Row interrupt. If the status section feature is used, upon reaching the begin status row location the CRSR will be loaded with SROW instead of HOME for that row. After which CRSR will revert back to load from HOME for the remaining rows on the screen.

2.3 SYSTEM CONTROL REGISTER

Through the System Control Register (SCR) the user specifies several important chip operational conditions. It is an 8-bit write only register which is loaded from the CPU accumulator.



TL/DD/5526-22

*Bit 0 is set to 1 by RESET and bit 7 is set to 0 by RESET.

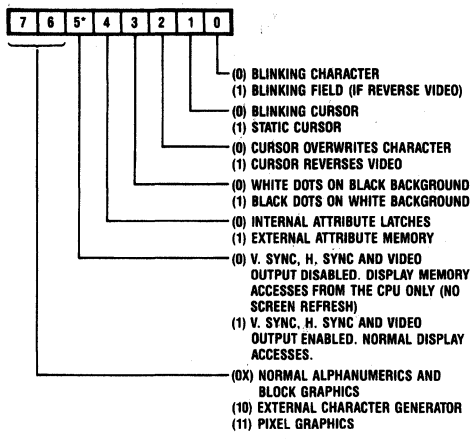
FIGURE 13. System Control Register

2.4 VIDEO CONTROL REGISTER

Through the Video Control Register (VCR) the user specifies several video display features to the chip. It is an 8-bit write only register which is loaded from the CPU accumulator.

5

2.0 Functional Description (Continued)



TL/DD/5526–23

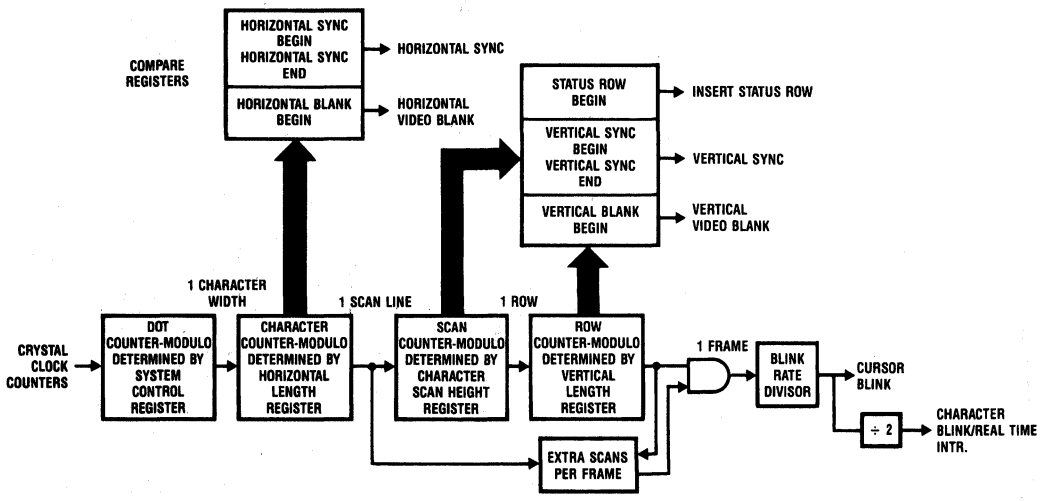
*Bit 5 is set to 0 by RESET.

FIGURE 14. Video Control Register

2.5 CRT REFRESH LOGIC

All video timing and clocking signals are derived from a series of counters and comparators called the Video Timing Chain. The chain is driven by the dot/crystal clock and ultimately divides down to the very slow blink clock, *(Figure 15)*. By having the program initialize the registers in the chain a user may specify all aspects of video generation.

The chain also controls the size and placement of the cursor and underline attribute within a character cell as well as the cell partitioning for block graphics display. All totaled, the chain consists of 14 wire only registers. They are loaded indirectly by using the Timing Chain Pointer (TCP), a 4-bit pointer to registers in the chain, and the MOV @TCP, A instruction.



TL/DD/5526–24

FIGURE 15. TMP Video Timing Chain

2.5.1 TMP Timing Chain Registers

TCP	Horizontal Timing
0	Horizontal Length Register — HLR 7 bits
	— Total number of character cells in a horizontal scan and retrace.
	— Enter desired count − 1
1	Horizontal Blank Begin Register — HBR 7 bits (Characters/Row)
	— Character position in horizontal scan after which horizontal blanking begins.
	— Enter desired number of displayed characters/row − 1.
2	Horizontal Sync Begin Register — HSBR 7 bits
	— Character position in horizontal scan after which horizontal sync begins (rising edge), HSBR ≤ HLR.
	— Enter desired count + 2.

2.0 Functional Description (Continued)

2.5.1 TMP Timing Chain Registers (Continued)

TCP — **Horizontal Timing**

3 Horizontal Sync End Register — HSER 7 bits
- — Character position in horizontal scan after which horizontal sync ends (falling edge), HSER ≤ HLR.
- — Enter desired count + 2.

Note: The polarity of the horizontal sync signal can be inverted by switching the values in the two horizontal sync registers.

TCP — **Character Height Definition**

4 High Nibble — Character Scan Height Register — CSHR 4 bits (see *Figure 16a*)
- — Scan line height of a character cell.
- — Enter desired number of scan lines − 1.

4 Low Nibble — Extra Scans/Frame — ES/F 4 bits
- — Number of extra scans to be added to a frame if desired.
- — Enter desired number of extra scans −1.
- — To get no extra scans make ES/F = CSHR. ES/F must be ≤ CSHR.

TCP — **Vertical Timing**

5 Vertical Length Register — VLR 5 bits
- — Total number of display and retrace rows in a frame.
- — Enter desired number of rows − 1.

6 Vertical Blank Register — VBR 5 bits (Rows/Screen)
- — Row position in vertical scan after which vertical blanking begins, VBR < VLR.
- — Enter desired number of displayed rows − 1.

7 High Nibble — Vertical Sync Begin Register — VSBR 4 bits
- — Scan line position in first blank row at which vertical sync begins (falling edge). Sync starts 1 char time after blanking for that line starts (except when VSBR = CSHR sync will start 1 char time after blanking of the last displayed scan line).
- — Enter desired scan line position − 1.

7 Low Nibble — Vertical Sync End Register — VSER 4 bits
- — Scan line position after start of vertical sync at which vertical sync ends (rising edge). Sync ends 1 char time after horizontal blanking for that scan line start.
- — Enter desired scan line position − 1.

Note: If VSER = VSBR there will be no vertical sync signal.

8 Status Row Begin Register — SRBR 5 bits
- — Row count after which the status row is inserted.
- — Enter desired row position − 1.

TCP — **Cursor and Graphics Control**

9 Upper 5 Bits — Blink Rate 5 bits
- — Divider driven by the vertical sync frequency to yield the slow cursor, character and real-time blink rates.
- — Enter desired divisor − 1.

9 Lower 3 Bits — Blink Duty Cycle 3 bits
- — Approximate ON time of blink signal.
- — 000 = shortest, 111 = longest (100 = 50% duty cycle).

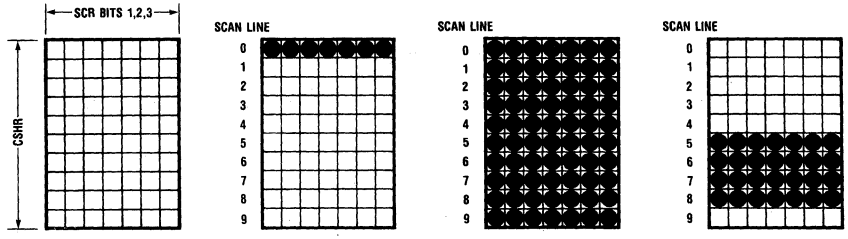
10 Graphics Column Register — GCR 8 bits
- — Assign dot positions to left, middle and right character cell columns for block graphics operation.

11 Graphics Row Register — GRR 8 bits
- — Defines scan count at which middle row for block graphics characters begins (upper nibble) and at which bottom row begins (lower nibble). The middle row (upper nibble) must be ≥ 1.
- — Enter desired scan count − 1.

12 Underline Size Register — USR 8 bits (see *Figures 16a, b, c*)
- — Defines the beginning (upper nibble) and ending (lower nibble) scan lines for the underline attribute. Values must be ≤ CSHR.

13 Cursor Size Register — CSR 8 bits (see *Figures 16a, b, c*)
- — Defines the beginning (upper nibble) and ending (lower nibble) scan lines for the cursor. Values must be ≤ CSHR.

2.0 Functional Description (Continued)



a. Character Cell Format Specification

b. Underline Size Register = 90

c. Cursor Size Register = 9A (0B, 0C, 0D, 0E, 0F) may also be used

d. Cursor Size Register = 48

TL/DD/5526–25

FIGURE 16. Underline and Cursor Register Operation

Note: The internal cursor flip-flop gets set to ON whenever a scan line corresponding to the begin cursor nibble is reached, and gets set to cursor OFF whenever a scan line corresponding to the end cursor nibble is reached. The cursor attributes are inserted whenever the character position being displayed corresponds to the one pointed to by the cursor address register. A similar situation applies for characters with the underline attribute selected. Therefore, care should be taken when setting the ES/F register and setting the cursor and underline sizes. In particular the ES/F value should not be between the upper nibble and lower nibble values of the underline size register or between the upper nibble and lower nibble values of the cursor size register. To use the cursor as a pointer without displaying it, set the lower nibble of the cursor size register to a value less than CSHR and the upper nibble to a value greater than CSHR.

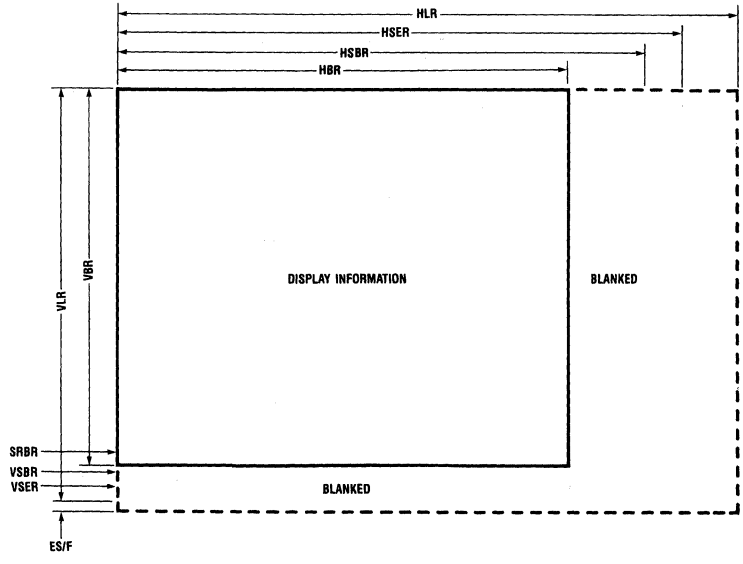
2.5.2 TIMING CHAIN LOAD VALUE EXAMPLE

It is desired to have a display field of 80 columns by 25 rows with the last screen row being a status row. It has been determined that 25 character width times will be necessary to complete horizontal retrace and that Horizontal sync should be positioned to start a full seven character times after blanking and end twenty characters after blanking to give us a total sync width of 13 character times. (See *Figure 17* for example.)

Additionally, vertical retrace will take 23 scan line times to complete with vertical sync starting three scan line times after vertical blanking begins and occupying a total period of 11 scan lines.

It is desired to make the character cells 12 scan lines tall. The cursor will be a block shape and occupy the bottom 11 scan lines in a cell. The underline attribute will actually be a strike through dash occupying the 4th scan line from the top in a cell.

Our line width is 80 displayed characters plus 25 for retrace making HLR = 80 + 25 − 1 = 104. Blanking will start after the 80th character so HBR = 80 − 1 = 79. To achieve seven character times after horizontal blanking, HSBR = 87 + 2 = 89. To achieve twenty character times after blanking HSER = 100 + 2 = 102 (note 102 − 89 = 13 total). Cell height is 12 lines so CSHR = 12 − 1 = 11. Since there are 12 scan lines per cell or row, vertical retrace will require 23/12 = 1 row and 11 scan lines. This makes our total row count VLR = 25 + 1 − 1 = 25 and ES/F = 11 − 1 = 10. Thus, timing chain location 4 would be coded: 1011 1010. We will display 25 rows so VBR = 25 − 1 = 24. Vertical sync will start at the beginning of the fourth scan



TL/DD/5526–26

FIGURE 17. Typical Video Screen Format Specification

2.0 Functional Description (Continued)

line of the row after blanking begins so VSBR = 4 − 1 = 3. It will run for 11 scan lines or specifically the 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2 ending at the beginning of the 3rd so VSER = 3 − 1 = 2. The status row will be after the 24th so SRBR = 24 − 1 = 23. To specify the underline and cursor sizes one must remember that the first scan line is numbered 0. To get our 11 line block cursor we begin after the 0 line and end at the end of the 11 line making CSR = 0000 1011. The underline dash will be USR = 0011 0100. Note that the CSHR determines the scan counter modulo and if a scan compare register value (ES/F, VSBR, VSER, USR, CSR) is never reached, the signal end or begin will never be initiated.

2.6 ATTRIBUTES

Eight independent attributes may be inserted itno the video dot stream to affect display characters on either an individual or global basis. The eight attributes along with their control word bit assignments are detailed in *Figure 18*. The scope with which a particular set of attributes affects the display depends upon whether attribute control is internal or external as determined by bit 4 in the VCR.

Attributes are present if the corresponding bit is a ZERO (low).

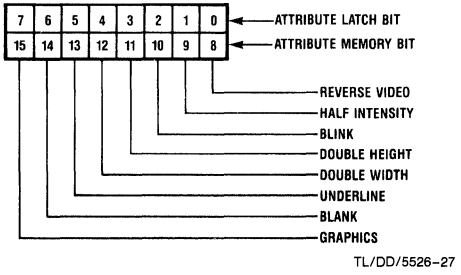


FIGURE 18. Attribute Bit Assignments

2.6.1 Internal Attribute Selection

In internal mode attribute control comes from one of two internal attribute latches designated AL0 and AL1, either of which is directly loadable from the CPU accumulator. The choice of which of the two is used for a particular display character is determined by bit 7 (MSB) in the display memory data byte with 0 = AL0 and 1 = AL1. (Characters are represented in display memory as ASCII values occupying the low 7 bits of each 8-bit byte thus leaving bit 7 free for attribute control.)

2.6.2 External Attribute Selection

In external mode each display character has associated with it, a dedicated attribute field in the form of a high 8-bit extension to the regular display memory character byte. To use this mode the system bus msut be configured for 16-bit bidirectional operation (SCR bit 4 = 1) and external attributes must be selected (VCR bit 4 = 1).

2.6.3 Attribute Processing

Each of the eight attributes may be independently enabled thus yielding a number of possible combinations. The exact processing involved is shown in *Figure 19*. Note that attributes are always present. Whether any of them are active depends upon the particular control bit being enabled in the latch or memory.

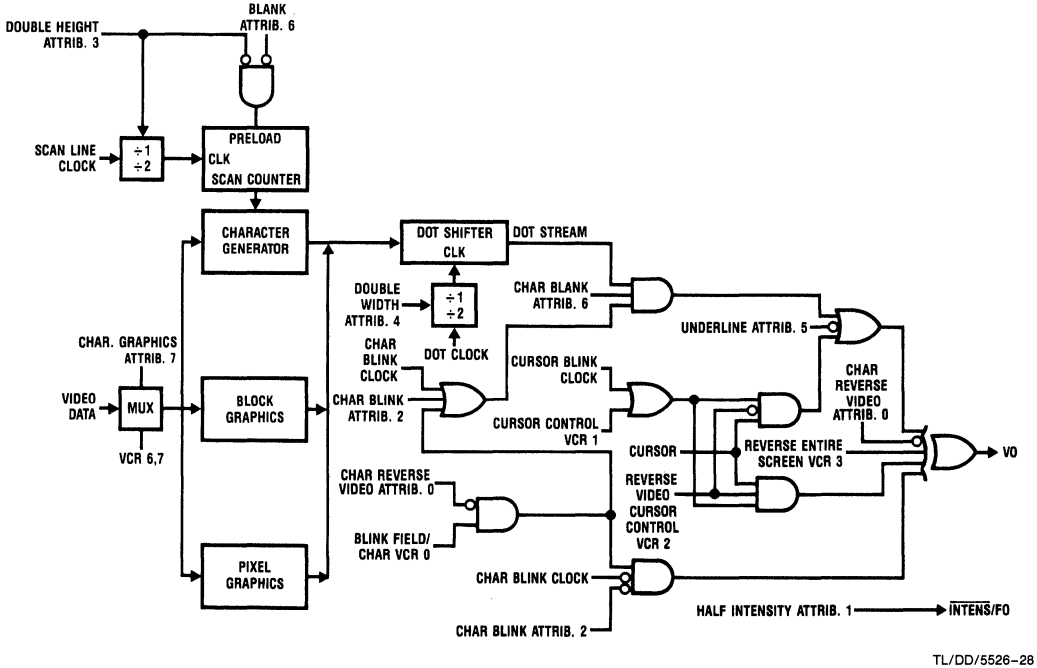


FIGURE 19. TMP Attribute Processing

2.0 Functional Description (Continued)

2.6.4 Attribute Operation

Reverse Video: A character and its surrounding cell are reversed in video from what was selected for the rest of the screen.

Half Intensity: To use the half intensity function the shared $\overline{\text{INTENSITY/FO CLK}}$ pin (25) must be selected for INTENSITY operation by setting SCR bit 6 low. In operation the half intensity pin will be low whenever a character for which the attribute is active is being displayed. To perform the actual attenuation function external circuitry must be connected between the $\overline{\text{INTEN}}$ and Video Output pins. In fact the signal may be used for another purpose such as switching between two colors.

Blink: A character or the field around it blinks as selected by VCR bit 0.

Double Height: A designated character is stretched out so that it will occupy a 2-row tall space. This attribute is implemented by slowing down by half the scan line stepping to the internal character generator. To use this attribute the desired double high character must be placed into the two display memory locations corresponding to the top and bottom row positions. For both locations the double high attribute is set. In addition the Blank attribute for the bottom character is also set to tell the controller it is the bottom half of a double high character. The double high attribute has no effect on element graphics or on pixel graphics displays. If an external character generator is used special circuitry must be employed to implement double high characters.

Double Width: A designated character is stretched out so that it will occupy a 2-character cell wide space. This attribute is implemented by slowing down by half the clock to the video dot shifter. To use this attribute the desired double wide character must be placed in the left character position and the double wide attribute bit set. The following character position (right) can have any character as it will be ignored.

Underline: If set this attribute causes the underline figure to be added to the video dot stream. Since the underline, like the cursor, can be specified as to position and size in the character cell, the underline can be an overline, block, strike through or any one of a number of effects. The underline overwrites any dot where it overlaps the character.

Blank/Double High Bottom: A character is inhibited from being displayed while still allowing it to be stored in the display memory. If this attribute and the double height attribute are set for the same character, the normal blank function is disabled for that character position and the character is displayed as the bottom half of a double height character.

Graphics: This attribute determines whether the video memory data byte as accessed by the display memory controller is routed through the character generator or block graphics control logic. If routed through the block graphics logic (attribute active) the effect on the video display will be as described in the Block Graphics section. Note that because Block Graphics mode is selected as an attribute it may be mixed in with normal alphanumerics characters. Also all other attributes with the exception of double height operate on the block graphics characters.

2.7 CHARACTER GENERATOR

The internal character generator holds 128 characters in a 7 x 11 matrix. The standard character sets are addressed using 7-bit ASCII codes stored in the display memory. When operating with fonts smaller than the maximum of 7 x 11, zeroes are encoded into the unused bits. When putting out a character the video controller always starts character generation on the second scan line of a row, leaving the first scan line blank. Similarly, the first (left) column in a character cell is blanked with character generation starting on the second column. Therefore, the specified cell size must be one greater in height and width than the display characters (including descenders) otherwise they will be chopped off. If the character cells are larger than the internal 7 x 11 matrix, blank dots will be put out after exhausting the internal generator (See *Figure 20* for example.)

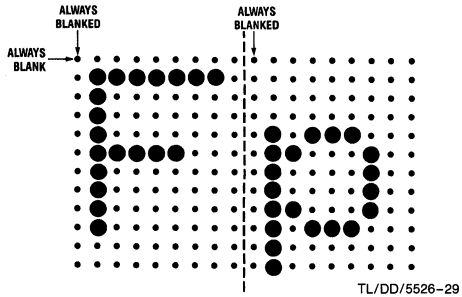


FIGURE 20. Character Cell Format

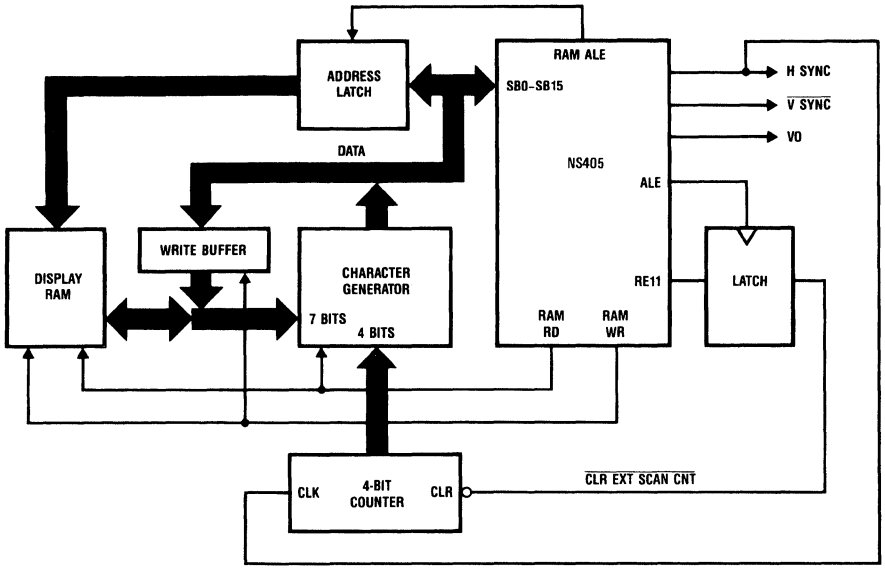
2.7.1 External Character Generation

The chip may be used with an external character generator by switching over to a pixel graphic display mode with modified address stepping as controlled by VCR bits 6, 7. In this mode an external character generator supplies pixel data to the chip as depicted in *Figure 21*. Character addressing comes from the display memory and scan line stepping from a 4-bit counter clocked by the Horizontal Sync. Scan line synchronization is achieved by using the Scan Count Clear signal coming out on RE11, pin 36. After the display of a row it pulses low to initialize the scan line counter for the start of a new row. In pixel mode both the character and any spacing between characters must be encoded into the external character generator. In addition, the chip will access and use at most 8 bits of pixel data for each character cell. However, if the cell width is specified to be 9 or 10, the ninth and tenth dots will repeat what was coded into the first. Therefore, assuming at least one dot spacing between characters, external fonts can at most be seven dots wide.

No limitations apply to the height of a character as long as the external generator can supply all of the scan lines as specified by the CSHR. As in regular pixel mode the LSB brought in is the first dot put out.

Since the eighth data bit is used for character generation it cannot effectively be used for internal attribute latch selection although one of the latches will be selected every data byte. Therefore, both internal attribute latches must be loaded with the same values. If external attribute operation is specified the full 8-bit high order attribute field is available for usage.

2.0 Functional Description (Continued)

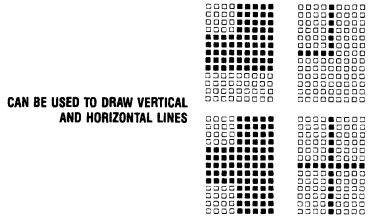


TL/DD/5526-30

FIGURE 21. External Character Set Implementation

2.8 BLOCK GRAPHICS

Block graphics is an alternative display mode to normal alphanumerics which is selected through attribute bit 7. Example *(Figure 22)*. It can operate on a character cell by character cell basis (see Attributes) and words by rerouting display memory bytes through the Block graphics logic instead of the internal character generator.



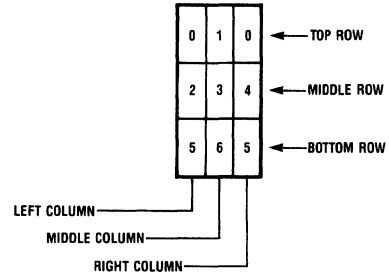
TL/DD/5526-31

FIGURE 22. Example Block Graphics Display Patterns

The Graphics Logic operates by partitioning the character cell space into nine possible areas as shown in *Figure 23* and then using the seven lower bits in the display data byte to turn these areas on or off. In this way one can draw contiguous lines or simple geometric figures while at the same time displaying alphanumeric characters in other cells.

The partitioning of the cell is controlled by two timing chain registers which specify two Horizontal and two Vertical cut off points to the graphics logic. Through these two registers one can make the sections as large or as small as desired, even eliminating sections entirely. Note that data bits 0 and 5 each control two sections as depicted in *Figure 23*.

2.8.1 Graphics Partitioning



TL/DD/5526-32

FIGURE 23. Block Graphics Cell Partitioning

The registers defining the graphics areas function as follows:

The Graphics Row Register — 8 bits (GRR) is divided into the following two (2) registers:

- Graphics Middle Row, (GMR):
  Defines the scan count at which the middle row begins (4 most significant bits of GRR).
- Graphics Bottom Row, (GBR):
  Defines the scan count at which the bottom row begins (4 least significant bits of GRR).

See *Figure 24.1a* for row example.

5

2.0 Functional Description (Continued)

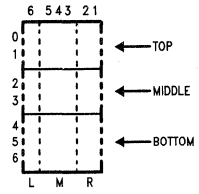
The Graphics Column Register — 8 bits (GCR) controls vertical partitioning through bit patterns as follows: (See *Figure 24.*)

7	6	5	4	3	2	1	0	
L	L	L	R	R	X	X	X	← 6 DOT FIELD WIDTH
L	L	L	R	R	R	X	X	← 7 DOT FIELD WIDTH
L	L	L	R	R	R	R	X	← 8 DOT FIELD WIDTH
L	L	L	L	R	R	R	R	← 9 DOT FIELD WIDTH

← GCR

TL/DD/5526–33

FIGURE 24. Block Graphics Column Partitioning



TL/DD/5526–44

GRR = 24
GCR = 60 (0110 0XXX)
cell size = 6 x 7

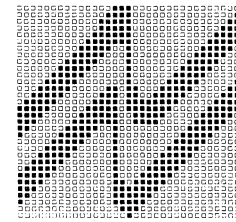
FIGURE 24.1a Block Graphics Example

For all bits in the Graphics Column Register, a one assigns that bit position to the middle column. A zero in an L bit position assigns that bit position to the left column. A zero in an R bit position assigns that bit position to the right column. There is always at least one middle dot although the left and right sections may be eliminated entirely. For 10 dot wide cells the 10th bit will repeat the 9th bit. An easy way to determine the column partitioning is to fill the GCR with all ones, thereby making it one large middle section. Then, starting from the outermost L and R bit positions, put zeros in until the left and right sections are the sizes needed.

2.9 PIXEL GRAPHICS

When bits 6 and 7 of the Video Control Register are both set to 1, the character generator and block graphics circuits are disabled. Video output directly reflects the contents of the display memory byte on a pixel (dot) per bit basis with data output LSB first. Example *(Figure 25).*

Nine bits at a time are accessed from each video memory location with as many bits being used as defined in the character cell width specification. If a cell width of 10 is specified



TL/DD/5526–34

FIGURE 25. Example Pixel Graphics

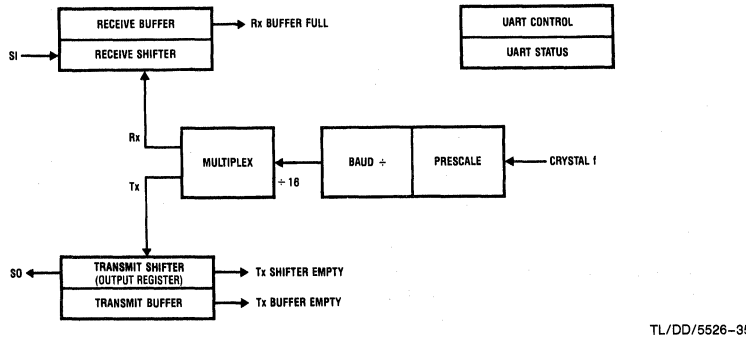
the 10th bit will merely repeat the 9th bit. Attributes are still operable in pixel mode, on a data byte basis, with internal and external operation possible. With internal attribute latch operation the same values must be loaded into both latches since the usual latch select bit is now being used for pixel control. Unless, however, only a 7 dot wide cell is used leaving the 8th bit free. With external attribute operation we are now limited to a 7-bit attribute field since pixel data can now occupy 9 of the 16 bus bits. Because of this the LSB attribute, Reverse Video is totally disabled from operation in Pixel Graphic mode. This also applies to internal attribute latch operation. Note, however, that reverse entire screen video is still operable. Address sequencing through the video memory is sequential with as many data bytes being read in as is necessary to satisfy the pixel requirements of the screen.

2.10 LIGHT PEN

Activation of the light pen interrupt causes the horizontal and vertical screen position of the currently displayed character to be latched into the Horizontal Light Pen Register HPEN (7 bits) and Vertical Light Pen Register VPEN (5 bits) respectively. Both HPEN and VPEN may be read into the CPU accumulator. The values latched remain in VPEN and HPEN until another light pen interrupt latches new values.

2.11 UART

The UART features full duplex operation with double buffered Receive and Transmit sections. Baud rate generation is fully programmable through a 2-stage divider chain. CPU control of the UART is extensive with polled or interrupt driven operation possible.



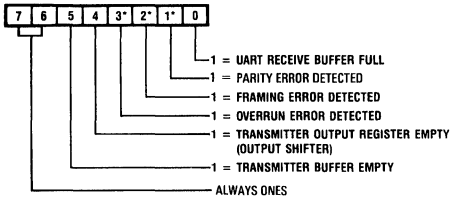
TL/DD/5526–35

FIGURE 26. TMP UART Block Diagram

2.0 Functional Description (Continued)

2.11.1 UART Control

UART Status Register (STAT): Contains error and status bits which reflect the internal state of the UART. Read into CPU accumulator. Bits 0, 5 are the same as those found in the internal interrupt register.



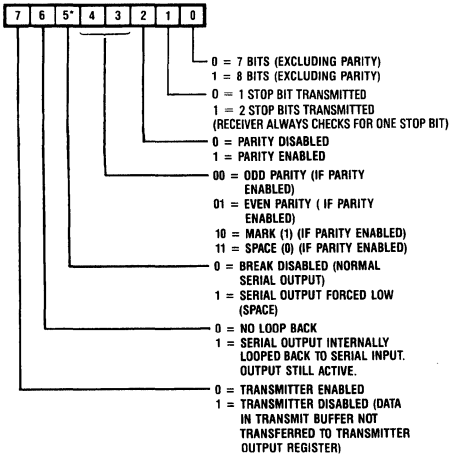
TL/DD/5526–36

UART Status Register bits 1, 2, 3 are only cleared on a chip reset or a read of the UART Receive Buffer. If another word were to come in before the Receive Buffer could be read the errors associated with the new word would add to those already present. The receipt of a new word can cause the three bits to go from a 0 to a 1, but not from a 1 to a 0.

FIGURE 27. UART Status Register

Note: The Transmit Output Register Empty flag is set to one whenever the transmitter is idle. The flag is reset to zero when a data character is transferred from the Transmit Buffer to the Output Register. This transfer does not occur until the next rising edge of the internal UART Transmit Clock. The Transmitter Output Register Empty flag occurs at the beginning of the last stop bit.

UART Control Register (UCR): Contains control bits which configure the format of transmitted data and tests made upon received data. Written to from CPU accumulator.



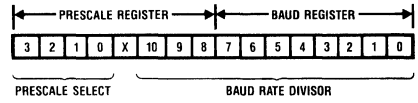
TL/DD/5526–37

*Bit 5 set to 0 by RESET.

FIGURE 28. UART Control Register

2.11.2 Baud Clock Generation

The basic BAUD clock is derived from the crystal frequency through a two-stage divider chain consisting of a 3.5–11 prescale and an 11-bit binary counter. *(Figure 29).* The divide factors are specified through 2 write only registers shown in *Figure 30.* Note that the 11-bit Baud Rate Divisor spills over into the Prescale Select Register. The correspondences between the 4-bit Prescale Select and Prescale factors is shown in Table I. There are many ways to calculate the two divisor factors but one particularly effective method would be to try to achieve a 1.8432 MHz frequency coming out of the first stage then use the BAUD Rate Divisor factors shown in Table II.



TL/DD/5526–39

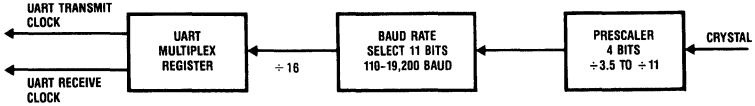
FIGURE 30. UART BAUD Clock Divider Registers

TABLE I. Prescale Factors

Prescale Select	Prescale Factor
0000	3.5
0001	4
0010	4.5
0011	5
0100	5.5
0101	6
0110	6.5
0111	7
1000	7.5
1001	8
1010	8.5
1011	9
1100	9.5
1101	10
1110	10.5
1111	11

TABLE II. Baud Rate Divisors (1.8432 MHz Input)

Baud Rate	Baud Rate Divisor (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



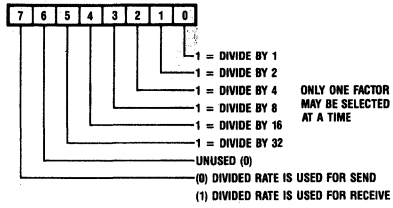
TL/DD/5526–38

FIGURE 29. UART BAUD Clock Generation

2.0 Functional Description (Continued)

The frequency coming out of the BAUD Rate Divisor is then passed through the UART Multiplex Register. Through the UART Multiplex Register one can specify that the Transmitter or Receiver clock be the same or a power of two multiple of the other.

UART Multiplex Register (UMX): Contains the bits which determine the divisor which is used to count down from the primary baud rate when different rates are used for send and receive (eight bits).



TL/DD/5526–40

FIGURE 31. UART Multiplex Register

The actual baud rate may be found from:

BR = Fc/(16*N*P*D)

Where:

BR is the Baud Rate

Fc is the external crystal frequency

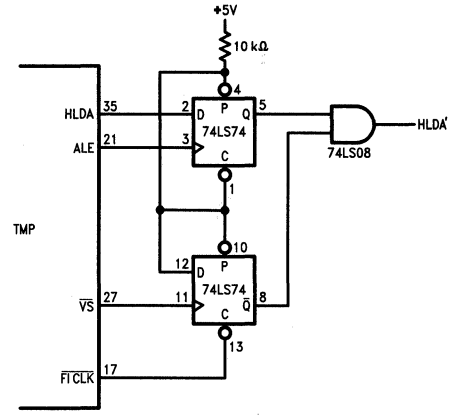
N is one plus the value of the Baud Rate Divisor contained in the Baud Rate Select Register and the Prescale Select Register.

P is the Prescale Divide Factor Selected by the value in the Prescale Select Register.

D is the Multiplex Register Divide Factor

3.0 Slave Processing

The TMP may be used as a slave video controller by having a host system perform Direct Memory Accesses into the display RAM. To assist in implementing such a system the chip features two DMA control pins—HOLD (Hold Request) and HLDA (Hold Acknowledge). These two signals come out on shared ROM Expand Bus pins RE8 and RE12. To request a DMA access a host would activate HOLD (active high and await the acknowledging HLDA from the TMP before proceeding with the DMA. The TMP only allows DMA operations during the vertical blanking period and will activate HLDA in response to a HOLD shortly after vertical blanking starts. In DMA mode all 16 TMP System Bus drivers are tri-stated while the bus control signals RAM ALE, RAM RD, RAM WR go to their inactive (high) states. A HOLD request must arrive two CPU cycles before vertical blanking starts; otherwise it will miss that retrace cycle and will have to wait until the next one, one frame later. Once DMA mode is entered, it is maintained for the duration of vertical blanking regardless of the state of HOLD. Near the end of vertical blanking the DMA mode will terminate in preparation for the display of the next frame, but the HLDA will NOT turn off. Specifically, this will occur one scan time before the end of vertical blanking. It is up to the designer to be sure that the host is off the BUS before this happens or suffer bus contention with the video controller. He can do this by either predetermining the length of time the host has to remain on the bus, or by using the end of vertical sync (as shown in *Figure 32*) to signal the end of a safe DMA period. If during DMA the CPU attempts to do a display memory access it would be put into a wait state until DMA is concluded and normal memory accessing is resumed.



TL/DD/5526–45

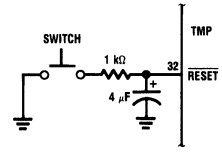
Vertical sync should be programmed to end as late as possible, but must end at least one scan time before the end of vertical blanking.

FIGURE 32

4.0 Reset

The TMP will reset if the $\overline{\text{RESET}}$ (32) pin is held at a logic low (< 0.8V) for at least five CPU cycle times. This pre-supposes that the V_{CC} is up, stable and within operational limits (+5V ±10%) and that the oscillator is running. For a power on reset, time must be allowed for the power supplies to stabilize (typically 50 ms) and the oscillator to start up. If power supply noise or ripple causes V_{CC} to exceed the +5V ±10% limits neither reset nor operation is guaranteed.

Internally, the $\overline{\text{RESET}}$ pin has a depletion load pullup that typically acts as a 30 μA current source from V_{CC} in the voltage range of interest. A typical reset circuit with a 0.5 second reset pulse is shown in *Figure 33*.



TL/DD/5526–41

FIGURE 33. Typical Reset Circuit

4.0 Reset (Continued)

During RESET a number of internal registers are initialized as follows:

4.1 CPU

CPU Clock divide = 1.5 (SCR bit 0 = 1)
Test Mode Control = 0 (SCR bit 7 = 0, Normal Operation)
Program Counter = 0
Stack Pointer = 0
Program Memory Bank = 0
RAM Register Bank = 0
Timer Stopped
Instruction Register cleared
F0 and F1 cleared

4.2 INTERRUPTS

Internal and External Interrupts disabled
Internal Interrupt Register set to 000011X0

4.3 UART

Receiver initialized to look for start bit
Status Register set to 11110000
Transmitter initialized to wait for OUT XMTR instruction
Control Register bit 5 = 0 (No BREAK)

4.4 VIDEO

Video generation shutdown (VCR bit 5 = 0)
FIFO Cleared Out
Timing Chain Character Counter = 0
Timing Chain Scan Counter = 0
Timing Chain Row Counter = 0
Timing Chain Blink Counter = 0
} IN TEST MODE ONLY

4.5 PIN STATES AT RESET

Pins 1–8 (SB0–7)	Tri-stated during reset and until either the CPU executes a MOVX instruction or bit 5 of the VCR is set.
Pins 9–16 (SB8–15)	If bit 4 of the SCR is set, SB8–15 will behave like SB0–7. If bit 4 of the SCR is cleared, SB8–15 will act as outputs (any of which may be either high or low). Note that bit 4 of the SCR may be one or zero at power-up.
Pin 17 (VID CLK/$\overline{\text{FI CLK}}$)	High during reset and until bit 5 of the VCR is set.
Pin 18 (RAM ALE)	High during reset and until the CPU executes a MOVX instruction or bit 5 of the VCR is set.
Pin 19 ($\overline{\text{RAM WR}}$)	High during reset and until the CPU executes a MOVX (of the output to display RAM variety) instruction.
Pin 20 ($\overline{\text{RAM RD}}$)	High during reset and until either the CPU executes a MOVX instruction or bit 5 of the VCR is set.
Pin 21 (ALE)	Pulses continuously.
Pin 22 (XTAL 2)	Crystal input or master clock input.
Pin 23 (XTAL 1)	Crystal input.
Pin 24 (Gnd.)	
Pin 25 ($\overline{\text{INTENS}}$/$\overline{\text{FO CLK}}$)	May be either high or low during reset.
Pin 26 (VO)	Low (because of asserted blanking signals) from reset until bit 5 of the VCR is set.
Pin 27 ($\overline{\text{VS}}$)	Tri-stated by reset, enabled when bit 5 of the VCR is set.
Pin 28 (HS)	Low from reset until bit 5 of the VCR is set.
Pin 29 (EA)	Input only.

4.0 Reset (Continued)

Pin 30 ($\overline{PSEN}$)	Active during reset if EA is high, always high if EA is low.
Pin 31 ($\overline{RD}$)	High during reset and until an IN PORT instruction is executed.
Pin 32 ($\overline{RESET}$)	Input only.
Pin 33 (SO)	High during reset and until an OUT XMTR instruction is executed.
Pin 34 (SI)	Input only.
Pin 35 (RE12/HLDA)	If EA is high or low and HOLD is low: low during reset. If EA is low and HOLD is high: may be either high or low during reset. If EA is high and HOLD is high: low at falling edge of ALE and during $\overline{PSEN}$, may be low or high at rising edge of ALE.
Pin 36 (RE11/$\overline{SC\ CLR}$)	If EA is low: output follows internal Scan Count Clear signal and is active during reset. If EA is high and reset asserted: low at falling edge of ALE and during $\overline{PSEN}$, sampled value of internal Scan Count Clear signal is output at rising edge of ALE.
Pin 37 (RE10/$\overline{INTR}$) Pin 38 (RE9/$\overline{LPEN}$) Pin 39 (RE8/HLDR)	If EA is low: inputs only. If EA is high and reset asserted: low at falling edge of ALE and during $\overline{PSEN}$. Always tri-stated at rising edge of ALE.
Pins 40–47 (RE0–7/1O0–7)	If EA is low: static port outputs (which may be high or low during reset). If EA is high and reset asserted: low at falling edge of ALE, tri-stated during $\overline{PSEN}$, and may be either high or low at the rising edge of ALE.
Pin 48 (V_{CC})	

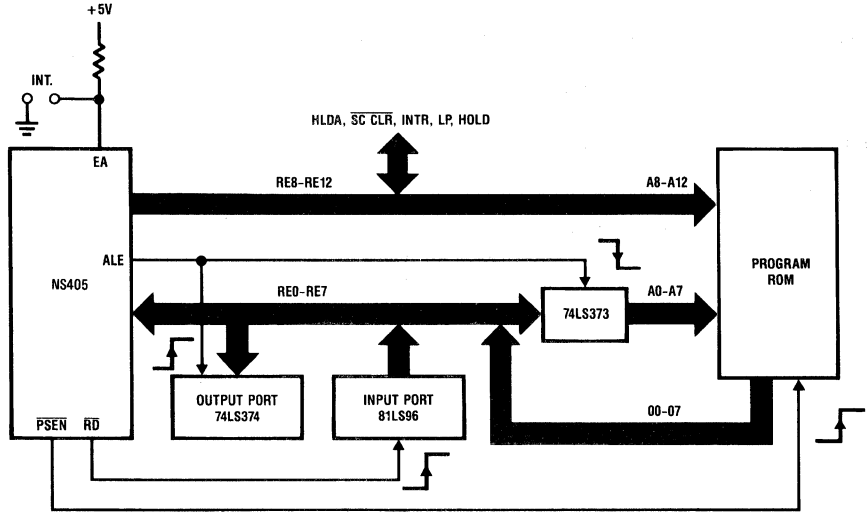
5.0 Extra Attributes

One may want to expand the external attribute field by adding more bits so that functions such as color (Red—Green—Blue drive) or grey scale may be implemented. Like the eight attributes which the chip handles internally these extra attributes would operate on a character cell basis. To add attribute bits one would have to duplicate the internal 4 level character/attribute FIFO externally using fast MSI chips. To assist in handling the external FIFO circuitry the TMP features two FIFO clocking signals on pins 17 and 25. The FIFO IN Clock ($\overline{FI\ CLK}$) is used to strobe attribute data into the external FIFO circuits in synchronism with the internal TMP FIFO. Its timing is identical to $\overline{RAM\ RD}$ but is only active when the video does a display RAM read to load its FIFO. The FIFO OUT Clock ($\overline{FO\ CLK}$) pulses for 1–3 bit times each time the video starts the display of a new character cell. The external FIFO would use the rising edge of this signal to clock out or latch the attribute output.

In order for the TMP CPU to access the additional attribute bits special bus gating arrangements would have to be worked out on the System Bus (Video Data Bus is at most 16 bits wide). Unless one were to run with internal attributes or only use a few of the external attributes in which case the unused bits could be used with the external FIFO. Whenever using the $\overline{FO\ CLK}$ the Intensity attribute is disabled since they both share the same pin.

6.0 TMP BUS Interfacing

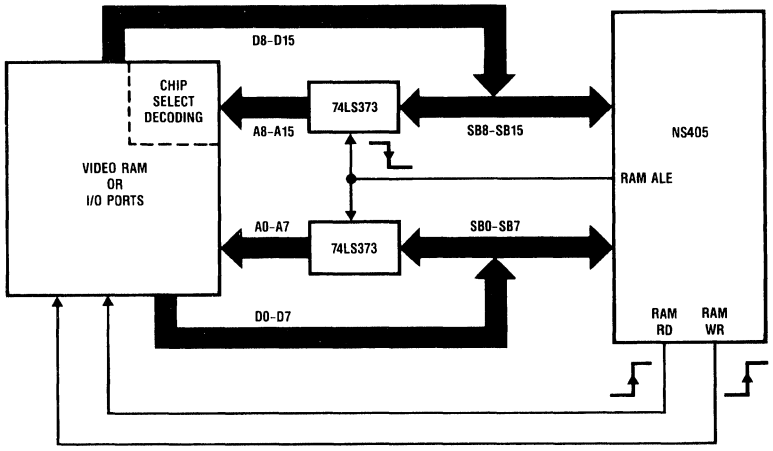
The two external buses on the TMP, ROM Expand and System are easily interfaced to as shown in *Figures 34* and *35*. Important bus information output from the chip is latched using the rising or falling edges of the various control signals. I/O port information is read in through a TRI-STATE® buffer chip such as an 81LS96.



TL/DD/5526–42

FIGURE 34. TMP ROM Expand BUS

6.0 TMP BUS Interfacing (Continued)



TL/DD/5526-43

FIGURE 35. TMP System Bus

TMP Registers (Excluding Timing Chain Registers)

TMP Registers		Associated Intructions			
		CPU SECTION			
A	= Accumulator — 8 bits	ADD A,Rr		MOV A,Rr	XCH A,Rr
#data	= data immediate	ADD A,#data		MOV A,@Rr	XCH A,@Rr
Rr	= Register	ADD A,@Rr		MOV A,#data	XCHD A,@Rr
@Rr	= Register pointed to by R0 or R1	ADDC A, Rr		MOV Rr,A	XRL A,Rr
		ADDC A,#data		MOV Rr,#data	XRL A,@Rr
		ADDC A,@Rr		MOV @Rr,A	XRL A,#data
		ANL A,Rr		MOV @Rr,#data	JBn addr
		ANL A,#data		MOVP A,@A	JNZ addr
		ANL A,@Rr		MOVP3 A,@A	JZ addr
		CLR A		RL A	DJNZ Rr,addr
		CPL A		RLC A	
		DAA		RR A	
		DEC A		RRC A	
		DEC Rr		ORL A,Rr	
		INC A		ORL A,@Rr	
		INC Rr		ORL A,#data	
		INC @Rr		SWAP A	
*HACC	= High Accumulator — 8 bits	*MOV A,HACC		*MOV HACC,A	
C	= Carry Bit	CLR C	CPL C	JNC addr	JC addr
*LONG R0	= Register Pair, R0, RA	*DECL R0		*INCL R0	*MOVL A,R0
		*MOVL R0,A		*MOVX A,@R0	*MOVX @R0,A
*LONG R1	= Register Pair R1, RB	*DECL R1		*INCL R1	*MOVL A,R1
		*MOVL R1,A		*MOVX A,@R1	*MOVX @R1,A
T	= Timer — 8 bits	MOV A,T		MOV T,A	STOP T
		STRT T		*JNTF addr	JTF addr
F0	= Flag 0	CLR F0	CPL F0	JF0 addr	*JNF0 addr
F1	= Flag 1	CLR F1	CPL F1	JF1 addr	*JNF1 addr
INTR	= Interrupt Register — 8 bits	MOV A,INTR		JNXI addr	JXI addr
		*DIS II		DIS XI	*EN II
		EN XI			

TMP Registers (Excluding Timing Chain Registers) (Continued)

TMP Registers		Associated Instructions		
CPU SECTION (Continued)				
MASK	= Internal Interrupt MasK — 8 bits	*MOV MASK,A		
PSW	= Program Status Word — 8 bits	MOV A,PSW	MOV PSW,A	
PORT	= 8 bit I/O Port	ANL PORT,#data	IN PORT	
		ORL PORT,#data	OUT PORT	
Miscellaneous Instructions		CALL addr	JMP addr	JMPP @A
		NOP	RET	RETR
		SEL MB0	SEL MB1	*SEL MB2
		*SEL MB3	SEL RB0	SEL RB1
VIDEO MANAGEMENT				
		Associated Instructions		
SCR	= System Control Register — 8 bits	*MOV SCR,A		
VCR	= Video Control Register — 8 bits	*MOV VCR,A		
HOME	= Home Address Register — 16 bits	*MOV A,HOME	*MOV HOME,A	
CURS	= Cursor Address Register — 16 bits	*DEC CURS	*INC CURS	*MOVX A,@CURS
		*MOV CURS,A	*MOV A,CURS	*MOVX @CURS,A
BEGD	= Beginning of Display RAM Register — 16 bits	*MOV BEGD,A		
ENDD	= End of Display RAM Register — 16 bits	*MOV ENDD,A		
SROW	= Status Row Register — 16 bits	*MOV SROW,A		
AL0	= Attribute Latch 0 — 8 bits	*MOV AL0,A		
AL1	= Attribute Latch 1 — 8 bits	*MOV AL1,A		
HPEN	= Horizontal Light Pen Register — 7 bits	*MOV A,HPEN		
VPEN	= Vertical Light Pen Register — 5 bits	*MOV A,VPEN		
VINT	= Vertical Interrupt Register — 5 bits	*MOV VINT,A		
UART CONTROL				
PSR	= Prescale Register (UART) — 8 bits	*MOV PSR,A		
BAUD	= Baud Rate Select Register — 8 bits	*MOV BAUD,A		
UCR	= UART Control Register — 8 bits	*MOV UCR,A		
UMX	= UART Multiplex Register — 8 bits	*MOV UMX,A		
STAT	= Status Latch (UART) — 6 bits	*MOV A,STAT		
RCVR	= UART Receive Buffer — 8 bits	*IN RCVR		
XMTR	= UART Transmit Buffer — 8 bits	*OUT XMTR		
TCP	= Timing Chain Pointer	*MOV TCP,A		
@TCP	= Register Pointed to by TCP	*MOV @TCP,A		

*New instruction added to 8040 subset.

Symbol Definitions

Symbol	Definition
AC	Auxiliary Carry Flag
addr	Program Memory Address
b	Bit Designator (b = 0 − 7)
BS	RAM Bank Switch
data	Number or Expression (8 bits)
DBF	Program Memory Bank Select Bits (2)
EXI	External Interrupt Pin
F0, F1	Internal Flags
P	I/O Port (8 bits)

Symbol	Definition
PC	Program Counter
SP	Stack Pointer
TF	Timer Flag
#	Prefix for Immediate Data
@	Prefix for Indirect Address
( )	Contents of Register
(( ))	Contents of Memory Location pointed to by designated register
←	Replaced by

Instruction Set

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
ADD A, Rr	0 1 1 0 1 r r r	(A) ← (A) + (Rr) for r= 0 − 7	Add contents of designated register to the Accumulator (8-bit operation)	1	1	*	*	*		
ADD A, #data	0 0 0 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) + data	Add immediate the specified data to the Accumulator (8-bit operation)	2	2	*	*	*		
ADD A, @ Rr	0 1 1 0 0 0 0 r	(A) ← (A) + ((Rr)) for r = 0 − 1	Add indirect the contents of data memory pointed to by Rr to the Accumulator (8-bit operation)	1	1	*	*	*		
ADDC A, Rr	0 1 1 1 1 r r r	(A) ← (A) + (C) + (Rr) for r = 0 − 7	Add with carry the contents of the designated register to the Accumulator (8-bit operation)	1	1	*	*	*		
ADDC A, # data	0 0 0 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) + (C) + data	Add immediate with carry the specified data to the Accumulator (8-bit operation)	2	2	*	*	*		
ADDC A, @ Rr	0 1 1 1 0 0 0 r	(A) ← (A) + (C) + ((Rr)) for r = 0 − 1	Add indirect with carry the contents of data memory pointed to by Rr to the Accumulator (8-bit operation)	1	1	*	*	*		
ANL A, Rr	0 1 0 1 1 r r r	(A) ← (A) AND (Rr) for r = 0 − 7	Logical AND contents of designated register with Accumulator (8-bit operation)	1	1					
ANL A, # data	0 1 0 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) AND data	Logical AND specified Immediate Data with Accumulator (8-bit operation)	2	2					
ANL A, @ Rr	0 1 0 1 0 0 0 r	(A) ← (A) AND ((Rr)) for r = 0 − 1	Logical AND indirect the contents of data memory pointed to by Rr with Accumulator (8-bit operation)	1	1					
ANL PORT, # data	0 1 1 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(P) ← (P) AND data	Logical AND immediate specified data with output port (8-bit operation)	2	2					
CALL addr	a10 a9 a8 1 0 1 0 0 a7 a6 a5 a4 a3 a2 a1 a0	((SP)) ← (PC0−12) ((SP)) ← (PSW3−7) (SP) ← (SP) + 1 (PC8−10) ← addr 8−10 (PC0−7) ← addr 0−7 (PC11−12 ← DBF 0, 1	Call designated subroutine	2	2					

5

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
CLR A	0 0 1 0 0 1 1 1	(A) ← 0	Clear the Accumulator	1	1					
CLR C	1 0 0 1 0 1 1 1	(C) ← 0	Clear carry bit	1	1	*				
CLR F0	1 0 0 0 0 1 0 1	(FO) ← 0	Clear Flag 0	1	1				*	
CLR F1	1 0 1 0 0 1 0 1	(F1) ← 0	Clear Flag 1	1	1					*
CPL A	0 0 1 1 0 1 1 1	(A) ← NOT (A)	Complement the contents of the Accumulator (8-bit operation)	1	1					
CPL C	1 0 1 0 0 1 1 1	(C) ← NOT (C)	Complement carry bit	1	1	*				
CPL F0	1 0 0 1 0 1 0 1	(F0) ← NOT (F0)	Complement Flag 0	1	1				*	
CPL F1	1 0 1 1 0 1 0 1	(F1) ← NOT (F1)	Complement Flag 1	1	1					*
DA A	0 1 0 1 0 1 1 1		Decimal Adjust the contents of the Accumulator (8-bit operation)	1	1	*	*			
DEC A	0 0 0 0 0 1 1 1	(HACC, A) ← (HACC, A) − 1	Decrement by 1 the contents of HACC/ ACC	1	1	*		*		
DEC CURS	0 0 0 0 1 0 1 0	(CURS) ← (CURS) − 1	Decrement by 1 the contents of the Cursor Address Register	1	1					
DEC Rr	1 1 0 0 1 r r r	(Rr) ← (Rr) − 1	Decrement by 1 the contents of the designated register (8-bit operation)	1	1	*				
DECL Rr	0 0 0 0 1 0 0 r	(Rr) ← (Rr) − 1 for r = 0 − 1	Decrement by 1 the contents of the designated 16-bit register pair	1	1					
DIS II	0 0 1 1 0 1 0 1		Disable internal interrupts	1	1					
DIS XI	0 0 0 1 0 1 0 1		Disable external interrupts	1	1					

DJNZ Rr, addr	1 1 1 0 1 r r r
	a7 a6 a5 a4 a3 a2 a1 a0
		(Rr) ← (Rr) − 1 for r = 0 − 7
		If (Rr) ≠ 0 do (PC0–7) ← addr
		If (Rr) = 0 do (PC) ← PC + 2
			Decrement the specified register and Jump if not zero to designated address within page (8-bit decrement)	2	2					

EN II	0 0 1 0 0 1 0 1		Enable internal interrupts.	1	1					
EN XI	0 0 0 0 0 1 0 1		Enable external interrupt.	1	1					
INC A	0 0 0 1 0 1 1 1	(HACC, A) ← (HACC, A) + 1	Increment by 1 the contents of HACC/A.	1	1	*		*		
INC CURS	0 0 1 1 1 0 1 0	(CURS) ← (CURS) + 1	Increment by 1 the contents of the Cursor Address Register.	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
INC Rr	0 0 0 1 1 r r r	(Rr) ← (Rr) + 1 for r = 0 − 7	Increment by 1 the contents of the designated register (8-bit increment)	1	1	*				
INC @ Rr	0 0 0 1 0 0 0 r	((Rr)) ← ((Rr)) + 1 for r = 0 − 1	Increment in direct the contents of data memory pointed to by Rr (8-bit increment)	1	1	*				
INCL Rr	0 0 1 1 1 0 0 r	(Rr) ← (Rr) + 1 for r = 0 − 1	Increment by 1 the contents of the designated 16-bit register pair	1	1					
IN PORT	1 1 1 0 0 0 0 1	(A) ← (P)	Input data from port into Accumulator (8-bit transfer)	2	1					
IN RCVR	1 1 1 0 0 0 0 0	(A) ← (RCVR)	Input contents of UART Receive buffer into Accumulator (8-bit transfer). Also, clears Receive Buffer Full interrupt.	1	1					
JBb addr	b2 b1 b0 1 0 0 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if (b) = 1 (PC) ← (PC) + 2 if (b) = 0 for b = 0 − 7	Jump to specified address within page if Accumulator bit is set	2	2					
JC addr	1 1 1 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if C = 1 (PC) ← (PC) + 2 if C = 0	Jump to specified address within page if Carry flag is set	2	2					
JF0 addr	1 0 0 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if F0 = 1 (PC) ← (PC) + 2 if F0 = 0	Jump to specified address within page if Flag F0 is set	2	2					
JF1 addr	0 1 1 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if F1 = 1 (PC) ← (PC) + 2 if F1 = 0	Jump to specified address within page if Flag F1 is set	2	2					
JMP addr	a10 a9 a8 0 0 1 0 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC8–10) ← addr 8–10 (PC0–7) ← addr 0–7 (PC11–12) ← DBF 0, 1	Direct Jump to specified address within 2k Bank	2	2					
JMPP @ A	1 0 1 0 0 0 1 1	(PC0–7) ← ((A))	Jump indirect within page to the address specified in the memory location pointed to by the Accumulator	2	1					
JNC addr	1 1 1 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if C = 0 (PC) ← (PC) + 2 if C = 1	Jump within page to specified address if Carry flag is 0	2	2					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
JNF0 addr	1 0 0 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if F0 = 0 (PC) ← (PC) + 2 if F0 = 1	Jump within page to specified address if F0 is 0	2	2					
JNF1 addr	0 1 1 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if F1 = 0 (PC) ← (PC) + 2 if F1 = 1	Jump within page to specified address if F1 is 0	2	2					
JNTF addr	0 0 0 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if TF = 0 (PC) ← (PC) + 2 if TF = 1, (TF) ← 0	Jump within page to specified address if Timer flag is reset. If not, continue and reset TF	2	2					
JNXI addr	1 0 1 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if EXI = LOW (PC) ← (PC) + 2 if EXI = HIGH	Jump within page to specified address if External Interrupt pin is LOW	2	2					
JNZ addr	1 1 0 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if A ≠ 0 (PC) ← (PC) + 2 if A = 0	Jump within page to specified address if Accumulator is not 0	2	2					
JTF addr	0 0 0 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if TF = 1, (TF) ← 0 (PC) ← (PC) + 2 if TF = 0	Jump within page to specified address if Timer flag is set. If jump taken Timer flag reset	2	2					
JXI addr	1 0 1 1 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if EXI = HIGH (PC) ← (PC) + 2 if EXI = LOW	Jump within page to specified address if External Interrupt pin is HIGH	2	2					
JZ addr	1 1 0 0 0 1 1 0 a7 a6 a5 a4 a3 a2 a1 a0	(PC0–7) ← addr if A = 0 (PC) ← (PC) + 2 if A ≠ 0	Jump within page to specified address if Accumulator is 0	2	2					
MOV A, CURS	1 0 0 1 1 0 1 1	(HACC/A) ← (CURS)	Copy the contents of the Cursor Address Register into the HACC/A (16-bit transfer)	1	1			*		
MOV A, HACC	1 1 1 0 0 0 1 0	(A) ← (HACC)	Copy contents of the High Accumulator into the Low Accumulator (8-bit transfer)	1	1					
MOV A, HOME	1 0 0 1 1 0 1 0	(HACC/A) ← (HOME)	Copy the contents of the Home Address register into the HACC/A (16-bit transfer)	1	1			*		
MOV A, HPEN	0 0 1 1 1 1 1 1	(A0–6) ← (HPEN) (A7) ← O	Copy the contents of the Horizontal Light Pen Register into the Accumulator (7-bit transfer, A7 cleared)	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOV A, INTR	1 0 0 0 1 1 0 0	(A) ← (INTR)	Copy the contents of the Interrupt Register into the Accumulator (8-bit transfer)	1	1					
MOV A, PSW	1 1 0 0 0 1 1 1	(A) ← (PSW)	Copy contents of the Program Status word into the Accumulator (8-bit transfer)	1	1					
MOV A, Rr	1 1 1 1 1 r r r	(A) ← (Rr) for r = 0 − 7	Copy the contents of the designated Register into the Accumulator (8-bit transfer)							
MOV A, STAT	1 0 0 1 1 1 0 0	(A0–5) ← (STAT) (A6–7) ← 11	Copy the contents of the UART Status Latch into the Accumulator (6-bit transfer, A6 and A7 set)	1	1					
MOV A, T	0 1 0 0 0 0 1 0	(A) ← (T)	Copy the contents of the Timer into the Accumulator (8-bit transfer)	1	1					
MOV A, VPEN	0 0 1 1 1 1 1 0	(A0–4) ← (VPEN) (A5–7) ← O	Copy contents of the Vertical Light Pen Register into the Accumulator (5-bit transfer, A5–A7 cleared)	1	1					
MOV A, @ Rr	1 1 1 1 0 0 0 r	(A) ← ((Rr)) for r = 0 − 1	Copy indirect the contents of data memory pointed to by Rr into the Accumulator (8-bit transfer)	1	1					
MOV A, # data	0 0 1 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← data	Load immediate the specified data into the Accumulator (8-bit load)	2	2					
MOV AL0, A	0 0 1 1 1 1 0 0	(AL0) ← (A)	Copy the contents of the Accumulator into Attribute Latch 0 (8-bit transfer)	1	1					
MOV AL1, A	0 0 1 1 1 1 0 1	(AL1) ← (A)	Copy the contents of the Accumulator into Attribute Latch 1 (8-bit transfer)	1	1					
MOV BAUD, A	0 0 0 0 0 0 1 0	(BAUD) ← (A)	Copy the contents of the Accumulator into the UART Baud Rate Select Register (8-bit transfer)	1	1					
MOV BEGD, A	0 0 0 0 1 1 0 1	(BEGD) ← (HACC/A)	Copy the contents of HACC/A into the Beginning of Display RAM Register (16-bit transfer)	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOV CURS, A	1 0 0 0 1 0 1 1	(CURS) ← (HACC/A)	Copy the contents of HACC/A into the Cursor Address Register (16-bit transfer)	1	1					
MOV ENDD, A	0 0 0 0 1 1 0 0	(ENDD) ← (HACC/A)	Copy the contents of HACC/A into the End of Display RAM Register (16-bit transfer)	1	1					
MOV HACC, A	1 1 0 0 0 0 1 0	(HACC) ← (A)	Copy the contents of the Low Accumulator into the High Accumulator (8-bit transfer)	1	1			*		
MOV HOME, A	1 0 0 0 1 0 1 0	(HOME) ← (HACC/A)	Copy the contents of HACC/A into the Home Address Register (16-bit transfer)	1	1					
MOV MASK, A	1 0 0 0 0 0 1 0	(MASK) ← (A)	Copy the contents of the Accumulator into the Interrupt Mask Register (8-bit transfer)	1	1					
MOV PSR, A	0 0 1 0 0 0 1 0	(PSR) ← (A)	Copy the contents of the Accumulator into the UART Prescale Register (8-bit transfer)	1	1					
MOV PSW, A	1 1 0 1 0 1 1 1	(PSW) ← (A)	Copy contents of the Accumulator into the Program Status Word (8-bit transfer)	1	1	*	*			
MOV Rr, A	1 0 1 0 1 r r r	(Rr) ← (A) for r = 0 − 7	Copy contents of the Accumulator into the designated register (8-bit transfer)	1	1					
MOV SCR, A	0 1 0 1 0 1 0 1	(SCR) ← (A)	Copy contents of the Accumulator into the System Control Register (8-bit transfer)	1	1					
MOV SROW, A	0 0 0 0 1 1 1 0	(SROW) ← (HACC/A)	Copy the contents of HACC/A into the Status Row Register (16-bit transfer)	1	1					
MOV T, A	0 1 1 0 0 0 1 0	(T) ← (A)	Copy the contents of the Accumulator into the Timer (8-bit transfer)	1	1					
MOV TCP, A	1 0 0 0 0 1 1 1	(TCP) ← (A)	Copy the contents of the Accumulator into the Timing Chain Pointer	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOV UCR, A	0 0 0 0 0 0 0 1	(UCR) ← (A)	Copy the contents of the Accumulator into the UART Control Register (8-bit transfer)	1	1					
MOV VCR, A	0 1 0 0 0 1 0 1	(VCR) ← (A)	Copy the contents of the Accumulator into the Video Control Register (8-bit transfer)	1	1					
MOV VINT, A	1 0 1 0 0 0 1 0	(VINT) ← (A)	Copy the contents of the Accumulator into the Vertical Interrupt Register	1	1					
MOV Rr, # data	1 0 1 1 1 r r r d7 d6 d5 d4 d3 d2 d1 d0	(Rr) ← data for r = 0 − 7	Load immediate the specified data into the designated register (8-bit load)	2	2					
MOV @ Rr, A	1 0 1 0 0 0 0 r	((Rr)) ← (A) for r = 0 − 1	Copy indirect the contents of the Accumulator into the data memory location pointed to by Rr (8-bit transfer)	1	1					
MOV @ Rr, # data	1 0 1 1 0 0 0 r d7 d6 d5 d4 d3 d2 d1 d0	((Rr)) ← data for r = 0 − 1	Load indirect the specified immediate data into the data memory location pointed to by Rr (8-bit load)	2	2					
MOV @ TCP, A	1 0 1 1 0 1 1 1	((TCP)) ← (A) (TCP) ← (TCP) + 1	Copy indirect the contents of the Accumulator into the Timing Chain Register pointed to by TCP. Contents of TCP incremented by 1	1	1					
MOV UMX, A	0 0 1 1 0 0 1 1	(UMX) ← (A)	Copy the contents of the Accumulator into the UART Multiplex Register (8-bit transfer)	1	1					
MOVL A, R0	1 0 0 1 1 0 0 0	(HACC/A) ← (RA, R0)	Copy the contents of RA, R0 into HACC/A (16-bit transfer)	1	1			*		
MOVL A, R1	1 0 0 1 1 0 0 1	(HACC/A) ← (RB, R1)	Copy the contents of RB, R1 into HACC/A (16-bit transfer)	1	1			*		
MOVL R0, A	1 0 0 0 1 0 0 0	(RA, R0) ← (HACC/A)	Copy the contents of HACC/A into RA, R0 (16-bit transfer)	1	1					
MOVL R1, A	1 0 0 0 1 0 0 1	(RB, R1) ← (HACC/A)	Copy the contents of HACC/A into RB, R1 (16-bit transfer)	1	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOVP A, @ A	1 0 1 1 0 0 1 1	(PC0–7) ← (A) (A) ← ((PC)) (PC0–7) ← (old PC0–7) + 1	Replace low 8 bits of PC with A. Load indirect within page the contents of the memory location pointed to by new PC into Accumulator. Restore PC with old value plus 1. Operates in all memory banks.	2	1					
MOVP3 A, @ A	1 1 1 1 0 0 1 1	(PC0–7) ← (A) (PC8–10) ← 011 (A) ← ((PC)) (PC) ← (old PC) + 1	Replace low 8 bits of PC with A. Next 3 bits replaced with 011. Load indirect within page 3 the contents of the memory location pointed to by new PC into the Accumulator. Restore PC with old value plus 1. Operates in all memory banks.	2	1					
MOVX A, @ CURS	1 0 0 1 1 1 0 1	(HACC/A) ← ((CURS))	Copy indirect the contents of display memory as pointed to by CURS into HACC/A (16-bit transfer)	Min. 2	1			*		
MOVX A, @ R0	1 0 0 1 0 0 0 0	(HACC/A) ← ((RA, R0))	Copy indirect the contents of display memory as pointed to by RA, R0 into HACC/A (16-bit transfer)	Min. 2	1			*		
MOVX A, @ R1	1 0 0 1 0 0 0 1	(HACC/A) ← ((RB, R1))	Copy indirect the contents of display memory as pointed to by RB, R1 into HACC/A (16-bit transfer)	Min. 2	1			*		
MOVX @ CURS, A	1 0 0 0 1 1 0 1	((CURS)) ← (HACC/A)	Copy indirect the contents of HACC/A into the display memory location as pointed to by CURS (16-bit transfer)	Min. 2	1					
MOVX @ R0, A	1 0 0 0 0 0 0 0	((RA, R0)) ← (HACC/A)	Copy indirect the contents of HACC/A into the display memory location as pointed to by RA, R0 (16-bit transfer)	Min. 2	1					

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
MOVX @ R1, A	1 0 0 0 0 0 0 1	((RB, R1)) ← (HACC/A)	Copy indirect the contents of HACC/A into the display memory location pointed to by RB, R1 (16-bit transfer)	Min. 2	1					
NOP	0 0 0 0 0 0 0 0		No Operation	1	1					
ORL A, Rr	0 1 0 0 1 r r r	(A) ← (A) OR (Rr) for r = 0 – 7	Logical OR contents of designated register with Accumulator (8-bit transfer)	1	1					
ORL A, @ Rr	0 1 0 0 0 0 0 r	(A) ← (A) OR ((Rr)) for r = 0 – 1	Logical OR indirect the contents of the data memory location pointed to by Rr with Accumulator (8-bit operation)	1	1					
ORL A, # data	0 1 0 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) OR data	Logical OR the specified immediate data with the Accumulator (8-bit operation)	2	2					
ORL PORT, # data	0 1 1 0 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(P) ← (P) OR data	Logical OR immediate specified data with output port	2	2					
OUT PORT	1 1 0 0 0 0 0 1	(P) ← (A)	Output the contents of the Accumulator to the I/O Port (8-bit transfer)	2	1					
OUT XMTR	1 1 0 0 0 0 0 0	(XMTR) ← (A)	Copy the contents of the Accumulator into the UART Transmit Buffer (8-bit transfer). Also clears Transmit Buffer empty interrupt	1	1					
RET	1 0 0 0 0 0 1 1	(SP) ← (SP) – 1 (PC0–12) ← ((SP))	Return from subroutine without restoring Program Status Word bits 5–7	2	1					
RETR	1 0 0 1 0 0 1 1	(SP) ← (SP) – 1 (PC0–12) ← ((SP)) (PSW 3–7) ← ((SP))	Return from Subroutine restoring Program Status Word (use for all returns from interrupts)	2	1	*	*			
RL A	1 1 1 0 0 1 1 1	(An + 1) ← (An) for n = 0 – 6 (A0) ← (A7)	Rotate Accumulator left by 1 bit without carry	1	1					
RLC A	1 1 1 1 0 1 1 1	(An + 1) ← (An) for n = 0 – 6 (A0) ← (C) (C) ← (A7)	Rotate Accumulator left by 1 bit through carry	1	1	*				

Instruction Set (Continued)

Mnemonic	Machine Code	Function	Description	Cycles	Bytes	Flags				
						C	AC	HACC	F0	F1
RR A	0 1 1 1 0 1 1 1	(An) ← (An) + 1 for n = 0 − 6	Rotate Accumulator right by 1 bit without carry	1	1					
RRC A	0 1 1 0 0 1 1 1	(An) ← (An) + 1 for n = 0 −6 (A7) ← (C) (C) ← (A0)	Rotate Accumulator right by 1 bit through carry	1	1	*				
SEL MB0	1 1 0 0 0 1 0 1	(DBF) ← 00	Select Bank 0 (0–2047) of Program Memory	1	1					
SEL MB1	1 1 0 1 0 1 0 1	(DBF) ← 01	Select Bank 1 (2048–4095) of Program Memory	1	1					
SEL MB2	1 1 1 0 0 1 0 1	(DBF) ← 10	Select Bank 2 (4096–6143) of Program Memory	1	1					
SEL MB3	1 1 1 1 0 1 0 1	(DBF) ← 11	Select Bank 3 (6144–8191) of Program Memory	1	1					
SEL RBn	1 1 n 0 0 0 1 1	(BS) ← n for n = 0 − 1	Select Data RAM Bank (0–7) or 1 (24–31)	1	1					
STOP T	0 1 1 0 0 1 0 1		Stop Timer	1	1					
STRT T	0 1 1 1 0 1 0 1		Start Timer	1	1					
SWAP A	0 1 0 0 0 1 1 1	(A4–A7) <–> (A0–A3)	SWAP 4 bit nibbles in Accumulator	1	1					
XCH A, Rr	0 0 1 0 1 r r r	(A) <–> (Rr) for r = 0 − 7	Exchange the Accumulator and contents of designated register (8-bit transfer)	1	1					
XCH A, @ Rr	0 0 1 0 0 0 0 r	(A) <–> ((Rr)) for r = 0 − 1	Exchange indirect the contents of the Accumulator and the data memory location pointed to by Rr (8-bit transfer)	1	1					
XCHD A, @ Rr	0 0 1 1 0 0 0 r	(A0–3) <–> ((Rr)) 0−3 for r = 0 − 1	Exchange indirect the low 4 bits of the Accumulator and the data memory location pointed to by Rr (4-bit transfer)	1	1					
XRL A, Rr	1 1 0 1 1 r r r	(A) ← (A) XOR (Rr) for r = 0 − 7	Logical XOR contents of designated register with Accumulator (8-bit transfer)	1	1					
XRL A, @ Rr	1 1 0 1 0 0 0 r	(A) ← (A) XOR ((Rr)) for r = 0 − 1	Logical XOR indirect the contents of the data memory location pointed to by Rr with the Accumulator	1	1					
XRL A, # data	1 1 0 1 0 0 1 1 d7 d6 d5 d4 d3 d2 d1 d0	(A) ← (A) XOR data	Logical XOR the immediate specified data with the Accumulator	2	2					

TMP Opcode Chart

MSN \ LSN	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	MOV UCR, A	MOV BAUD, A	ADD A, #data	JMP (page 0)	EN XI	JNTF	DEC A	DECL R0	DECL R1	DEC CURS		MOV ENDD, A	MOV BECD, A	MOV SROW, A	
1	INC @R0	INC @R1	JB0	ADDC A, #data	CALL (page 0)	DIS XI	JTF	INC A	INC R0	INC R1	INC R2	INC R3	INC R4	INC R5	INC R6	INC R7
2	XCH A, @R0	XCH A, @R1	MOV PSR, A	MOV A, #data	JMP (page 1)	EN II		CLR A	XCH A, R0	XCH A, R1	XCH A, R2	XCH A, R3	XCH A, R4	XCH A, R5	XCH A, R6	XCH A, R7
3	XCHD A, @R0	XCHD A, @R1	JB1	MOV UMX, A	CALL (page 1)	DIS II		CPL A	INCL R0	INCL R1	INC CURS		MOV AL0, A	MOV AL1, A	MOV A, VPEN	MOV A, HFEN
4	ORL A, @R0	ORL A, @R1	MOV A,T	ORL A, #data	JMP (page 2)	MOV VCR, A		SWAP A	ORL A, R0	ORL A, R1	ORL A, R2	ORL A, R3	ORL A, R4	ORL A, R5	ORL A, R6	ORL A, R7
5	ANL A, @R0	ANL A, @R1	JB2	ANL A, #data	CALL (page 2)	MOV SCR, A		DA A	ANL A, R0	ANL A, R1	ANL A, R2	ANL A, R3	ANL A, R4	ANL A, R5	ANL A, R6	ANL A, R7
6	ADD A, @R0	ADD A, @R1	MOV T,A	ORL PORT, #data	JMP (page 3)	STOP T	JNF1	RRC A	ADD A, R0	ADD A, R1	ADD A, R2	ADD A, R3	ADD A, R4	ADD A, R5	ADD A, R6	ADD A, R7
7	ADDC A, @R0	ADDC A, @R1	JB3	ANL PORT, #data	CALL (page 3)	STRT T	JF1	RR A	ADDC A, R0	ADDC A, R1	ADDC A, R2	ADDC A, R3	ADDC A, R4	ADDC A, R5	ADDC A, R6	ADDC A, R7
8	MOVX @R0, A	MOVX @R1, A	MOV MASK, A	RET	JMP (page 4)	CLR F0	JNF0	MOV TCP, A	MOVL R0, A	MOVL R1, A	MOV HOME, A	MOV CURS, A	MOV A, INTR	MOVX @CURS, A		
9	MOVX A, @R0	MOVX A, @R1	JB4	RETR	CALL (page 4)	CPL F0	JF0	CLR C	MOVL A, R2	MOVL A, R1	MOV A, HOME	MOV A, CURS	MOV A, STAT	MOVX A, @ CURS		
A	MOV @R0, A	MOV @R1, A	MOV VINT, A	JMPP @A	JMP (page 5)	CLR F1	JNXI	CPL C	MOV R0, A	MOV R1, A	MOV R2, A	MOV R3, A	MOV R4, A	MOV R5, A	MOV R6, A	MOV R7, A
B	MOV @R0, #data	MOV @R1, #data	JB5	MOVP A, @A	CALL (page 5)	CPL F1	JXI	MOV @TCP, A	MOV R0, #data	MOV R1, #data	MOV R2, #data	MOV R3, #data	MOV R4, #data	MOV R5, #data	MOV R6, #data	MOV R7, #data
C	OUT XMTR	OUT PORT	MOV HACC, A	SEL RB0	JMP (page 6)	SEL MB0	JZ	MOV A, PSW	DEC R0	DEC R1	DEC R2	DEC R3	DEC R4	DEC R5	DEC R6	DEC R7
D	XRL A, @R0	XRL A, @R1	JB6	XRL A, #data	CALL (page 6)	SEL MB1	JNZ	MOV PSW, A	XRL A, R0	XRL A, R1	XRL A, R2	XRL A, R3	XRL A, R4	XRL A, R5	XRL A, R6	XRL A, R7
E	IN RCVR	IN PORT	MOV A, HACC	SEL RB1	JMP (page 7)	SEL MB2	JNC	RL A	DJNZ R0	DJNZ R1	DJNZ R2	DJNZ R3	DJNZ R4	DJNZ R5	DJNZ R6	DJNZ R7
F	MOV A, @R0	MOV A, @R1	JB7	MOVP3 A, @A	CALL (page 7)	SEL MB3	JC	RLC A	MOV A, R0	MOV A, R1	MOV A, R2	MOV A, R3	MOV A, R4	MOV A, R5	MOV A, R6	MOV A, R7

5

Ordering Information

ORDER PART NUMBERS

ROMless	NS405-A12N NS405-B12N NS405-C12N	NS405-B18N

National Semiconductor Corporation

PRELIMINARY

DP8390/NS32490 Network Interface Controller

General Description

The DP8390/NS32490 Network Interface Controller (NIC) is a microCMOS VLSI device designed to ease interfacing with CSMA/CD type local area networks including Ethernet, Cheapernet and STARLAN®. The NIC implements all Media Access Control (MAC) layer functions for transmission and reception of packets in accordance with the IEEE 802.3 Standard. Unique dual DMA channels and an internal FIFO provide a simple yet efficient packet management design. To minimize system parts count and cost, all bus arbitration and memory support logic are integrated into the NIC.

The NIC is the heart of a three chip set that implements the complete IEEE 802.3 protocol and node electronics as shown below. The other two chips are the DP8391 Serial Network Interface (SNI) and the DP8392 Coaxial Transceiver Interface (CTI).

Features

- Compatible with IEEE 802.3/Ethernet II/Cheapernet/STARLAN
- Interfaces with 8-, 16- and 32-bit microprocessor systems
- Implements simple, versatile buffer management
- Forms integral part of DP8390, 91, 92 Ethernet/Cheapernet solution
- Requires single 5V supply
- Utilizes low power microCMOS process
- Includes
  - — Two 16-bit DMA channels
  - — 16-byte internal FIFO with programmable threshold
  - — Network statistics storage
- Supports physical, multicast, and broadcast address filtering
- Provides 3 levels of loopback
- Utilizes independent system and network clocks

Table of Contents

1.0 SYSTEM DIAGRAM

2.0 BLOCK DIAGRAM

3.0 FUNCTIONAL DESCRIPTION

4.0 TRANSMIT/RECEIVE PACKET ENCAPSULATION/ DECAPSULATION

5.0 PIN DESCRIPTIONS

6.0 DIRECT MEMORY ACCESS CONTROL (DMA)

7.0 PACKET RECEPTION

8.0 PACKET TRANSMISSION

9.0 REMOTE DMA

10.0 INTERNAL REGISTERS

11.0 INITIALIZATION PROCEDURES

12.0 LOOPBACK DIAGNOSTICS

13.0 BUS ARBITRATION AND TIMING

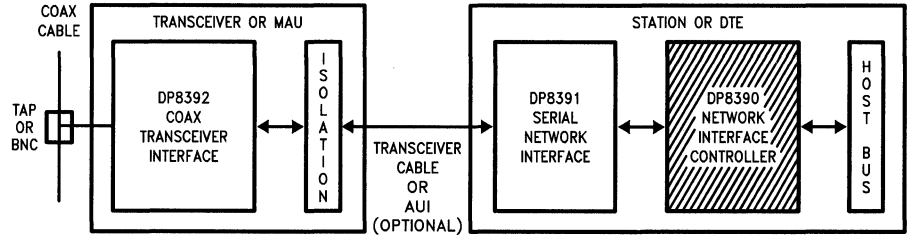
14.0 PRELIMINARY ELECTRICAL CHARACTERISTICS

15.0 SWITCHING CHARACTERISTICS

16.0 PHYSICAL DIMENSIONS

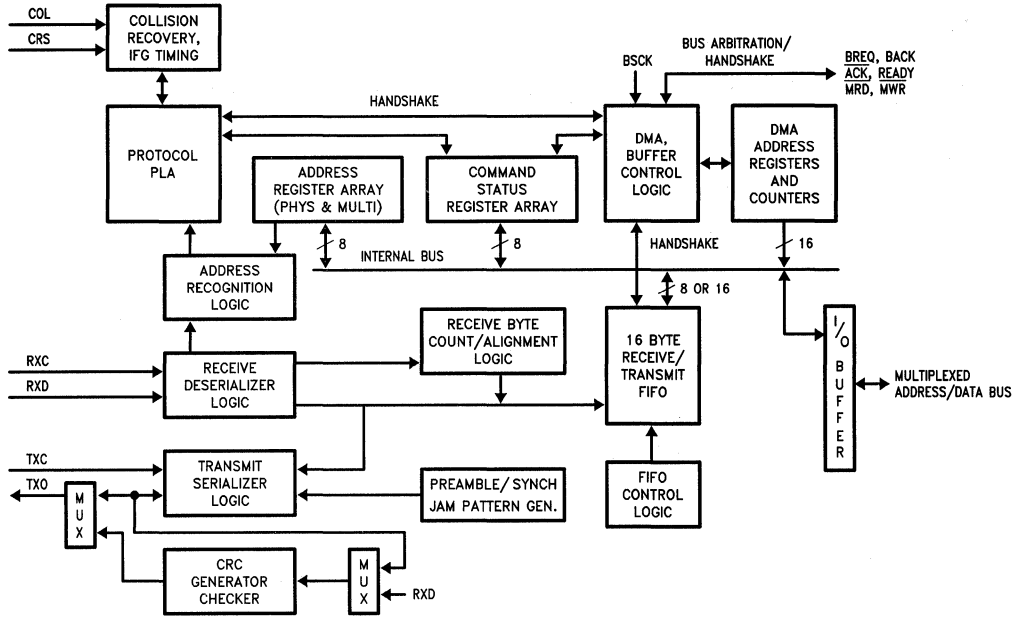
1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set



TL/F/8582–1

2.0 Block Diagram



TL/F/8582–2

FIGURE 1

3.0 Functional Description

(Refer to *Figure 1*)

RECEIVE DESERIALIZER

The Receive Deserializer is activated when the input signal Carrier Sense is asserted to allow incoming bits to be shifted into the shift register by the receive clock. The serial receive data is also routed to the CRC generator/checker. The Receive Deserializer includes a synch detector which detects the SFD (Start of Frame Delimiter) to establish where byte boundaries within the serial bit stream are located. After every eight receive clocks, the byte wide data is transferred to the 16-byte FIFO and the Receive Byte Count is incremented. The first six bytes after the SFD are checked for valid comparison by the Address Recognition Logic. If the Address Recognition Logic does not recognize the packet, the FIFO is cleared.

CRC GENERATOR/CHECKER

During transmission, the CRC logic generates a local CRC field for the transmitted bit sequence. The CRC encodes all fields after the synch byte. The CRC is shifted out MSB first following the last transmit byte. During reception the CRC logic generates a CRC field from the incoming packet. This local CRC is serially compared to the incoming CRC appended to the end of the packet by the transmitting node. If the local and received CRC match, a specific pattern will be generated and decoded to indicate no data errors. Transmission errors result in a different pattern and are detected, resulting in rejection of a packet.

TRANSMIT SERIALIZER

The Transmit Serializer reads parallel data from the FIFO and serializes it for transmission. The serializer is clocked by the transmit clock generated by the Serial Network Interface (DP8391). The serial data is also shifted into the CRC generator/checker. At the beginning of each transmission, the Preamble and Synch Generator append 62 bits of 1,0 preamble and a 1,1 synch pattern. After the last data byte of the packet has been serialized the 32-bit FCS field is shifted directly out of the CRC generator. In the event of a collision the Preamble and Synch generator is used to generate a 32-bit JAM pattern of all 1's

ADDRESS RECOGNITION LOGIC

The address recognition logic compares the Destination Address Field (first 6 bytes of the received packet) to the Physical address registers stored in the Address Register Array. If any one of the six bytes does not match the pre-programmed physical address, the Protocol Control Logic rejects the packet. All multicast destination addresses are filtered using a hashing technique. (See register description) If the multicast address indexes a bit that has been set in the filter bit array of the Multicast Address Register Array the packet is accepted, otherwise it is rejected by the Protocol Control Logic. Each destination address is also checked for all 1's which is the reserved broadcast address.

FIFO AND FIFO CONTROL LOGIC

The NIC features a 16-byte FIFO. During transmission the DMA writes data into the FIFO and the Transmit Serializer reads data from the FIFO and transmits it. During reception the Receive Deserializer writes data into the FIFO and the DMA reads data from the FIFO. The FIFO control logic is used to count the number of bytes in the FIFO so that after a preset level, the DMA can begin a bus access and write/read data to/from the FIFO before a FIFO underflow//overflow occurs.

3.0 Functional Description (Continued)

PROTOCOL PLA

The protocol PLA is responsible for implementing the Ethernet protocol, including collision recovery with random backoff. The Protocol PLA also formats packets during transmission and strips preamble and synch during reception.

DMA AND BUFFER CONTROL LOGIC

The DMA and Buffer Control Logic is used to control two 16-bit DMA channels. During reception, the Local DMA stores packets in a receive buffer ring, located in buffer memory. During transmission the Local DMA uses programmed pointer and length registers to transfer a packet from local buffer memory to the FIFO. A second DMA channel is used as a slave DMA to transfer data between the local buffer memory and the host system. The Local DMA and Remote DMA are internally arbitrated, with the Local DMA channel having highest priority. Both DMA channels use a common external bus clock to generate all required bus timing. External arbitration is performed with a standard bus request, bus acknowledge handshake protocol.

4.0 Transmit/Receive Packet Encapsulation/Decapsulation

A standard IEEE 802.3 packet consists of the following fields: preamble, Start of Frame Delimiter (SFD), destination address, source address, length, data, and Frame Check Sequence (FCS). The typical format is shown in *Figure 2.* The packets are Manchester encoded and decoded by the DP8391 SNI and transferred serially to the NIC using NRZ data with a clock. All fields are of fixed length except for the data field. The NIC generates and appends the preamble, SFD and FCS field during transmission. The Preamble and SFD fields are stripped during reception. (The CRC is passed through to buffer memory during reception.)

PREAMBLE AND START OF FRAME DELIMITER (SFD)

The Manchester encoded alternating 1,0 preamble field is used by the SNI (DP8391) to acquire bit synchronization with an incoming packet. When transmitted each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble will be lost as the packet travels through the network. The preamble field is stripped by the NIC. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern which consists of two consecutive 1's. The NIC does not treat the SFD pattern as a byte, it detects only the two bit pattern. This allows any preceding preamble within the SFD to be used for phase locking.

DESTINATION ADDRESS

The destination address indicates the destination of the packet on the network and is used to filter unwanted packets from reaching a node. There are three types of address formats supported by the NIC: physical, multicast, and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of "0". These addresses are compared to the internally stored physical address registers. Each bit in the destination address must match in order for the NIC to accept the packet. Multicast addresses begin with an MSB of "1". The DP8390 filters multicast addresses using a standard hashing algorithm that maps all multicast addresses into a six bit value. This six bit value indexes a 64 bit array that filters the value. If the address consists of all 1's it is a broadcast address, indicating that the packet is intended for all nodes. A promiscuous mode allows reception of all packets: the destination address is not required to match any filters. Physical, broadcast, multicast, and promiscuous address modes can be selected.

SOURCE ADDRESS

The source address is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is simply passed to buffer memory.

LENGTH FIELD

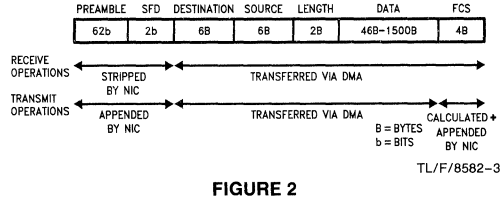
The 2-byte length field indicates the number of bytes that are contained in the data field of the packet. This field is not interpreted by the NIC.

DATA FIELD

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes will require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field. **The NIC does not strip or append pad bytes for short packets, or check for oversize packets.**

FCS FIELD

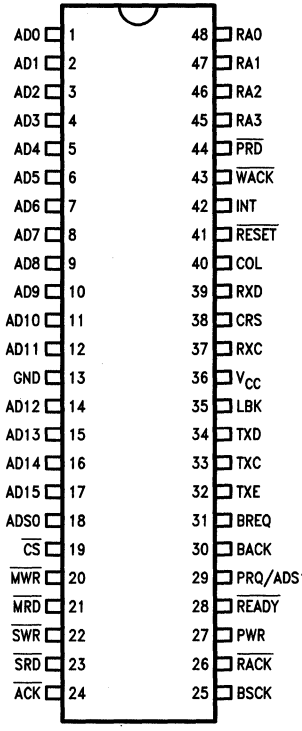
The Frame Check Sequence (FCS) is a 32-bit CRC field calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected. The AUTODIN II ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X^1 + 1$) polynomial is used for the CRC calculations.



TL/F/8582–3

FIGURE 2

Connection Diagram



TL/F/8582–4

Order Number DP8390D or DP8390N
See NS Package Number D48A or N48A

5.0 Pin Descriptions

BUS INTERFACE PINS

Symbol	DIP Pin No	Function	Description
AD0–AD15	1–12 14–17	I/O,Z	**MULTIPLEXED ADDRESS/DATA BUS:** • Register Access, with DMA inactive, $\overline{CS}$ low and $\overline{ACK}$ returned from NIC, pins AD0–AD7 are used to read/write register data. AD8–AD15 float during I/O transfers. $\overline{SRD}$, $\overline{SWR}$ pins are used to select direction of transfer. • Bus Master with BACK input asserted During t1 of memory cycle AD0–AD15 contain address During t2, t3, t4 AD0–AD15 contain data (word transfer mode). During t2, t3, t4 AD0–AD7 contain data, AD8–AD15 contain address (byte transfer mode). Direction of transfer is indicated by NIC on $\overline{MWR}$, $\overline{MRD}$ lines.
ADS0	18	I/O,Z	**ADDRESS STROBE 0** • Input with DMA inactive and $\overline{CS}$ low, latches RA0–RA3 inputs on falling edge. If high, data present on RA0–RA3 will flow through latch. • Output when Bus Master, latches address bits (A0–A15) to external memory during DMA transfers.

5.0 Pin Descriptions (Continued)

BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No	Function	Description
$\overline{CS}$	19	I	**CHIP SELECT:** Chip Select places controller in slave mode for μP access to internal registers. Must be valid through data portion of bus cycle. RA0–RA3 are used to select the internal register. $\overline{SWR}$ and $\overline{SRD}$ select direction of data transfer.
$\overline{MWR}$	20	O,Z	**MASTER WRITE STROBE:** Strobe for DMA transfers, active low during write cycles (t2, t3, tw) to buffer memory. Rising edge coincides with the presence of valid output data. TRI-STATE® until BACK asserted.
$\overline{MRD}$	21	O,Z	**MASTER READ STROBE:** Strobe for DMA transfers, active during read cycles (t2, t3, tw) to buffer memory. Input data must be valid on rising edge of $\overline{MRD}$. TRI-STATE until BACK asserted.
$\overline{SWR}$	22	I	**SLAVE WRITE STROBE:** Strobe from CPU to write an internal register selected by RA0–RA3.
$\overline{SRD}$	23	I	**SLAVE READ STROBE:** Strobe from CPU to read an internal register selected by RA0–RA3.
$\overline{ACK}$	24	O	**ACKNOWLEDGE:** Active low when NIC grants access to CPU. Used to insert WAIT states to CPU until NIC is synchronized for a register read or write operation.
RA0–RA3	45–48	I	**REGISTER ADDRESS:** These four pins are used to select a register to be read or written. The state of these inputs is ignored when the NIC is not in slave mode ($\overline{CS}$ high).
$\overline{PRD}$	44	O	**PORT READ:** Enables data from external latch onto local bus during a memory write cycle to local memory (remote write operation). This allows asynchronous transfer of data from the system memory to local memory.
$\overline{WACK}$	43	I	**WRITE ACKNOWLEDGE:** Issued from system to NIC to indicate that data has been written to the external latch. The NIC will begin a write cycle to place the data in local memory.
INT	42	O	**INTERRUPT:** Indicates that the NIC requires CPU attention after reception transmission or completion of DMA transfers. The interrupt is cleared by writing to the ISR. All interrupts are maskable.
$\overline{RESET}$	41	I	**RESET:** Reset is active low and places the NIC in a reset mode immediately, no packets are transmitted or received by the NIC until STA bit is set. Affects Command Register, Interrupt Mask Register, Data Configuration Register and Transmit Configuration Register. The NIC will execute reset within 10 BUSK cycles.
BREQ	31	O	**BUS REQUEST:** Bus Request is an active high signal used to request the bus for DMA transfers. This signal is automatically generated when the FIFO needs servicing.
BACK	30	I	**BUS ACKNOWLEDGE:** Bus Acknowledge is an active high signal indicating that the CPU has granted the bus to the DP8390. If immediate bus access is desired, BREQ should be tied to BACK. **Tying BACK to V_{CC} will result in a deadlock.**
PRQ, $\overline{ADS1}$	29	O,Z	**PORT REQUEST/ADDRESS STROBE 1** • 32 BIT MODE: If LAS is set in the Data Configuration Register, this line is programmed as ADS1. It is used to strobe addresses A16–A31 into external latches. (A16–A31 are the fixed addresses stored in RSAR0, RSAR1.) ADS1 will remain at TRI-STATE until BACK is received. • 16 BIT MODE: If LAS is not set in the Data Configuration Register, this line is programmed as PRQ and is used for Remote DMA Transfers. In this mode PRQ will be a standard logic output. **NOTE: This line will power up as TRI-STATE until the Data Configuration Register is programmed.**
$\overline{READY}$	28	I	**READY:** This pin is set high to insert wait states during a DMA transfer. The NIC will sample this signal at t3 during DMA transfers.

5.0 Pin Descriptions (Continued)

BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No	Function	Description
$\overline{PWR}$	27	O	**PORT WRITE:** Strobe used to latch data from the NIC into external latch for transfer to host memory during Remote Read transfers. The rising edge of $\overline{PWR}$ coincides with the presence of valid data on the local bus.
$\overline{RACK}$	26	I	**READ ACKNOWLEDGE:** Indicates that the system DMA or host CPU has read the data placed in the external latch by the NIC. The NIC will begin a read cycle to update the latch.
BSCK	25	I	This clock is used to establish the period of the DMA memory cycle. Four clock cycles (t1, t2, t3, t4) are used per DMA cycle. DMA transfers can be extended by one BSCK increments using the $\overline{READY}$ input.
NETWORK INTERFACE PINS			
COL	40	I	**COLLISION DETECT:** This line becomes active when a collision has been detected on the coaxial cable. During transmission this line is monitored after preamble and synch have been transmitted. At the end of each transmission this line is monitored for CD heartbeat.
RXD	39	I	**RECEIVE DATA:** Serial NRZ data received from the DP8391 SNI, clocked into the NIC on the rising edge of RXC.
CRS	38	I	**CARRIER SENSE:** This signal is provided by the DP8391 SNI and indicates that carrier is present on the coax. This signal is active high.
RXC	37	I	**RECEIVE CLOCK:** Re-synchronized clock from the DP8391 SNI used to clock data from the SNI into the DP8390 NIC.
LBK	35	O	**LOOPBACK:** This output is set high when the NIC is programmed to perform a loopback through the SNI. This output is connected directly to the loopback control pin (LBK) on the DP8391 SNI.
TXD	34	O	**TRANSMIT DATA:** Serial NRZ Data output to the DP8391 SNI. The data is valid on the rising edge of TXC.
TXC	33	I	**TRANSMIT CLOCK:** This clock is used to provide timing for internal operation and to shift bits out of the transmit serializer. TXC is nominally a 10 MHz clock provided by the SNI.
TXE	32	O	**TRANSMIT ENABLE:** This output becomes active when the first bit of the packet is valid on TxD and goes low after the last bit of the packet is clocked out of TxD. This signal connects directly to the DP8391 SNI. This signal is active high.
POWER			
V_{CC}	36		+5V DC is required. It is suggested that a decoupling capacitor be connected between these pins. It is essential to provide a path to ground for the GND pin with the lowest possible impedance.
GND	13		

6.0 Direct Memory Access Control (DMA)

The DMA capabilities of the NIC greatly simplify use of the DP8390 in typical configurations. The local DMA channel transfers data between the FIFO and memory. On transmission, the packet is DMA'd from memory to the FIFO in bursts. Should a collision occur (up to 15 times), the packet is retransmitted with no processor intervention. On reception, packets are DMAed from the FIFO to the receive buffer ring (as explained below).

A remote DMA channel is also provided on the NIC to accomplish transfers between a buffer memory and system memory. The two DMA channels can alternatively be combined to form a single 32-bit address with 8- or 16-bit data.

DUAL DMA CONFIGURATION

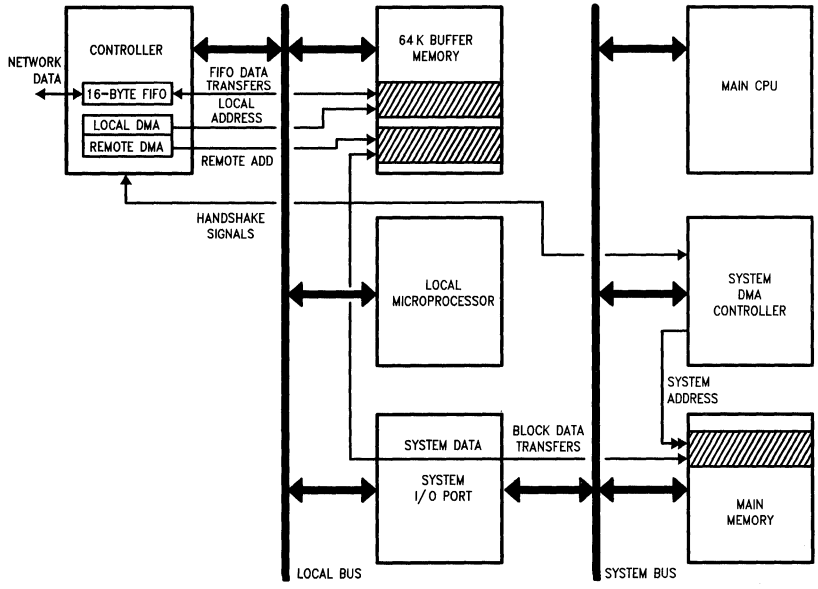
An example configuration using both the local and remote DMA channels is shown below. Network activity is isolated on a local bus, where the NIC's local DMA channel performs burst transfers between the buffer memory and the NIC's FIFO. The Remote DMA transfers data between the buffer memory and the host memory via a bidirectional I/O port. The Remote DMA provides local addressing capability and is used as a slave DMA by the host. Host side addressing must be provided by a host DMA or the CPU. The NIC allows Local and Remote DMA operations to be interleaved.

SINGLE CHANNEL DMA OPERATION

If desirable, the two DMA channels can be combined to provide a 32-bit DMA address. The upper 16 bits of the 32-bit address are static and are used to point to a 64K page of memory where packets are to be received and transmitted.

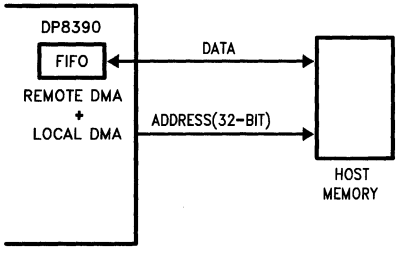
6.0 Direct Memory Access Control (DMA) (Continued)

Dual Bus System



TL/F/8582–55

32-Bit DMA Operation

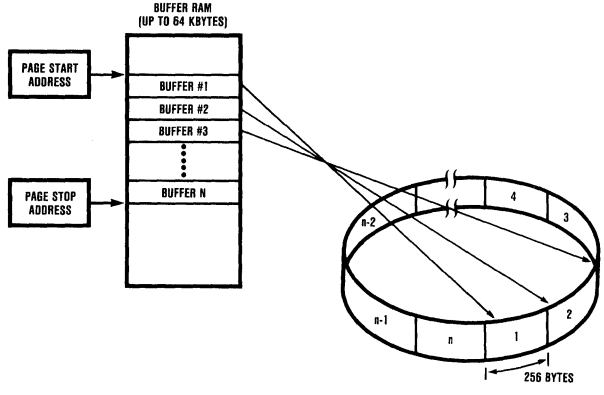


TL/F/8582–6

7.0 Packet Reception

The Local DMA receive channel uses a Buffer Ring Structure comprised of a series of contiguous fixed length 256 byte (128 word) buffers for storage of received packets. The location of the Receive Buffer Ring is programmed in two registers, a Page Start and a Page Stop register. Ethernet packets consist of a distribution of shorter link control packets and longer data packets, the 256 byte buffer length provides a good compromise between short packets and longer packets to most efficiently use memory. In addition these buffers provide memory resources for storage of back-to-back packets in loaded networks. The assignment of buffers

NIC Receive Buffer Ring



TL/F/8582–7

5

7.0 Packet Reception (Continued)

for storing packets is controlled by Buffer Management logic in the NIC. The Buffer Management logic provides three basic functions: linking receive buffers for long packets, recovery of buffers when a packet is rejected, and recirculation of buffer pages that have been read by the host.

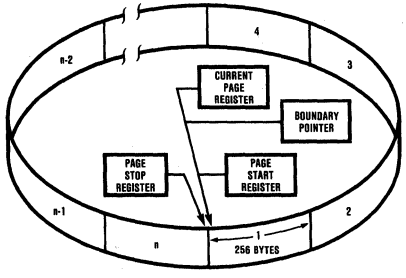
At initialization, a portion of the 64k address space is reserved for the receive buffer ring. Two eight bit registers, the Page Start Address Register (PSTART) and the Page Stop Address Register (PSTOP) define the physical boundaries of where the buffers reside. The NIC treats the list of buffers as a logical ring; whenever the DMA address reaches the Page Stop Address, the DMA is reset to the Page Start Address.

INITIALIZATION OF THE BUFFER RING

Two static registers and two working registers control the operation of the Buffer Ring. These are the Page Start Register, Page Stop Register (both described previously), the Current Page Register and the Boundary Pointer Register. The Current Page Register points to the first buffer used to store a packet and is used to restore the DMA for writing status to the Buffer Ring or for restoring the DMA address in the event of a Runt packet, a CRC, or Frame Alignment error. The Boundary Register points to the first packet in the Ring not yet read by the host. If the local DMA address ever reaches the Boundary, reception is aborted. The Boundary Pointer is also used to initialize the Remote DMA for removing a packet and is advanced when a packet is removed. A simple analogy to remember the function of these registers is that the Current Page Register acts as a Write Pointer and the Boundary Pointer acts as a Read Pointer.

NOTE: At initialization, the Page Start Register value should be loaded into both the Current Page Register and the Boundary Pointer Register.

Receive Buffer Ring At Initialization

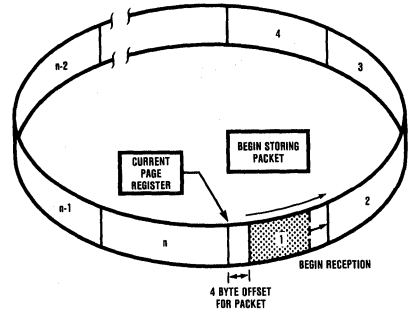


TL/F/8582–30

BEGINNING OF RECEPTION

When the first packet begins arriving the NIC begins storing the packet at the location pointed to by the Current Page Register. An offset of 4 bytes is saved in this first buffer to allow room for storing receive status corresponding to this packet.

Received Packet Enters Buffer Pages



TL/F/8582–31

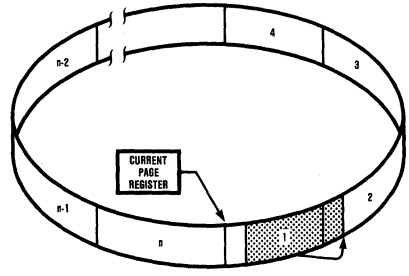
LINKING RECEIVE BUFFER PAGES

If the length of the packet exhausts the first 256 byte buffer, the DMA performs a forward link to the next buffer to store the remainder of the packet. For a maximal length packet the buffer logic will link six buffers to store the entire packet. Buffers cannot be skipped when linking, a packet will always be stored in contiguous buffers. Before the next buffer can be linked, the Buffer Management Logic performs two comparisons. The first comparison tests for equality between the DMA address of the next buffer and the contents of the Page Stop Register. If the buffer address equals the Page Stop register, the buffer management logic will restore the DMA to the first buffer in the Receive Buffer Ring value programmed in the Page Start Address register. The second comparison tests for equality between the DMA address of the next buffer address and the contents of the Boundary Pointer register. If the two values are equal the reception is aborted. The Boundary Pointer register can be used to protect against overwriting any area in the receive buffer ring that has not yet been read. When linking buffers, buffer management will never cross this pointer, effectively avoiding any overwrites. If the buffer address does not match either the Boundary Pointer or Page Stop address, the link to the next buffer is performed.

Linking Buffers

Before the DMA can enter the next contiguous 256 byte buffer, the address is checked for equality to PSTOP and to the Boundary Pointer. If neither are reached, the DMA is allowed to use the next buffer.

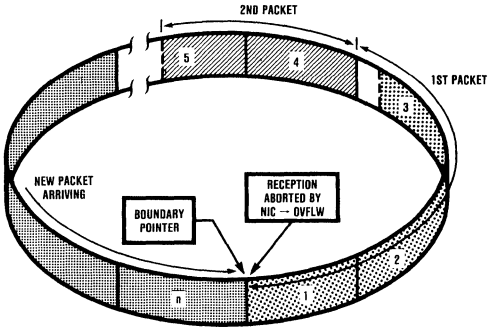
Linking Receive Buffer Pages



TL/F/8582–32

7.0 Packet Reception (Continued)

Received Packet Aborted If It Hits Boundary Pointer



TL/F/8582–33

Buffer Ring Overflow

If the Buffer Ring has been filled and the DMA reaches the Boundary Pointer Address, reception of the incoming packet will be aborted by the NIC. Thus, the packets previously received and still contained in the Ring will not be destroyed.

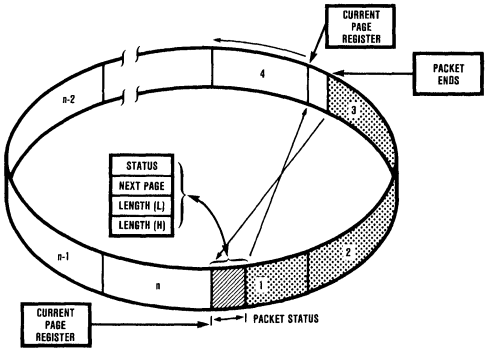
END OF PACKET OPERATIONS

At the end of the packet the NIC determines whether the received packet is to be accepted or rejected. It either branches to a routine to store the Buffer Header or to another routine that recovers the buffers used to store the packet.

SUCCESSFUL RECEPTION

If the packet is successfully received as shown, the DMA is restored to the first buffer used to store the packet (pointed to by the Current Page Register). The DMA then stores the Receive Status, a Pointer to where the next packet will be stored (Buffer 4) and the number of received bytes. Note that the remaining bytes in the last buffer are discarded and reception of the next packet begins on the next empty 256-byte buffer boundary. The Current Page Register is then initialized to the next available buffer in the Buffer Ring. (The location of the next buffer had been previously calculated and temporarily stored in an internal scratchpad register.)

Termination of Received Packet—Packet Accepted

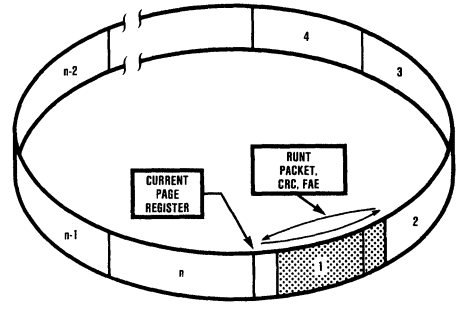


TL/F/8582–34

BUFFER RECOVERY FOR REJECTED PACKETS

If the packet is a runt packet or contains CRC or Frame Alignment errors, it is rejected. The buffer management logic resets the DMA back to the first buffer page used to store the packet (pointed to by CURR), recovering all buffers that had been used to store the rejected packet. This operation will not be performed if the NIC is programmed to accept either runt packets or packets with CRC or Frame Alignment errors. The received CRC is always stored in buffer memory after the last byte of received data for the packet.

Termination of Received Packet—Packet Rejected



TL/F/8582–35

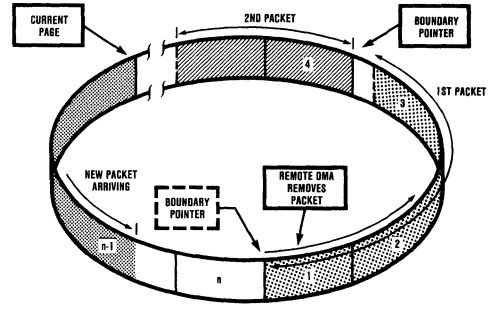
Error Recovery

If the packet is rejected as shown, the DMA is restored by the NIC by reprogramming the DMA starting address pointed to by the Current Page Register.

REMOVING PACKETS FROM THE RING

Packets are removed from the ring using the Remote DMA or an external device. When using the Remote DMA the Send Packet command can be used. This programs the Remote DMA to automatically remove the received packet pointed to by the Boundary Pointer. At the end of the transfer, the NIC moves the Boundary Pointer, freeing additional buffers for reception. The Boundary Pointer can also be moved manually by programming the Boundary Register. Care should be taken to keep the Boundary Pointer at least one buffer behind the Current Page Pointer. This is discussed in the User Guide.

1st Received Packet Removed By Remote DMA



TL/F/8582–36

5

7.0 Packet Reception (Continued)

STORAGE FORMAT FOR RECEIVED PACKETS

The following diagrams describe the format for how received packets are placed into memory by the local DMA channel. These modes are selected in the Data Configuration Register.

Storage Format

AD15	AD8	AD7	AD0
Next Packet Pointer		Receive Status	
Receive Byte Count 1		Receive Byte Count 0	
Byte 2		Byte 1	

BOS = 0, WTS = 1 in Data Configuration Register.

This format used with NSC32000 808X type processors.

AD15	AD8	AD7	AD0
Next Packet Pointer		Receive Status	
Receive Byte Count 0		Receive Byte Count 1	
Byte 1		Byte 2	

BOS = 1, WTS = 1 in Data Configuration Register.

This format used with 68000 type processors.

AD7	AD0
Receive Status	
Next Packet Pointer	
Receive Byte Count 0	
Receive Byte Count 1	
Byte 0	
Byte 1	

BOS = 0, WTS = 0 in Data Configuration Register.

This format used with general 8-bit CPUs.

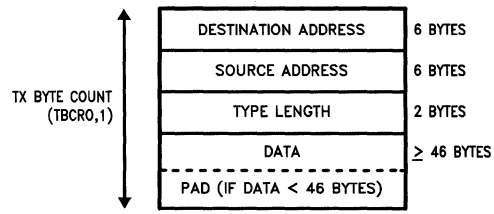
8.0 Packet Transmission

The Local DMA is also used during transmission of a packet. Three registers control the DMA transfer during transmission, a Transmit Page Start Address Register (TPSR) and the Transmit Byte Count Registers (TBCR0,1). When the NIC receives a command to transmit the packet pointed to by these registers, buffer memory data will be moved into the FIFO as required during transmission. The NIC will generate and append the preamble, synch and CRC fields.

TRANSMIT PACKET ASSEMBLY

The NIC requires a contiguous assembled packet with the format shown. The transmit byte count includes the Destination Address, Source Address, Length Field and Data. It does not include preamble and CRC. When transmitting data smaller than 46 bytes, the packet must be padded to a minimum size of 64 bytes. The programmer is responsible for adding and stripping pad bytes.

General Transmit Packet Format



TL/F/8582–9

TRANSMISSION

Prior to transmission, the TPSR (Transmit Page Start Register) and TBCR0, TBCR1 (Transmit Byte Count Registers) must be initialized. To initiate transmission of the packet the TXP bit in the Command Register is set. The Transmit Status Register (TSR) is cleared and the NIC begins to prefetch transmit data from memory (unless the NIC is currently receiving). If the interframe gap has timed out the NIC will begin transmission.

CONDITIONS REQUIRED TO BEGIN TRANSMISSION

In order to transmit a packet, the following three conditions must be met:

1. The Interframe Gap Timer has timed out the first 6.4 μs of the Interframe Gap (See appendix for Interframe Gap Flowchart)
2. At least one byte has entered the FIFO. (This indicates that the burst transfer has been started)
3. If the NIC had collided, the backoff timer has expired.

In typical systems the NIC has already prefetched the first burst of bytes before the 6.4 μs timer expires. The time during which NIC transmits preamble can also be used to load the FIFO.

Note: If carrier sense is asserted before a byte has been loaded into the FIFO, the NIC will become a receiver.

COLLISION RECOVERY

During transmission, the Buffer Management logic monitors the transmit circuitry to determine if a collision has occurred. If a collision is detected, the Buffer Management logic will reset the FIFO and restore the Transmit DMA pointers for retransmission of the packet. The COL bit will be set in the TSR and the NCR (Number of Collisions Register) will be incremented. If 15 retransmissions each result in a collision the transmission will be aborted and the ABT bit in the TSR will be set.

Note: NCR reads as zeroes if excessive collisions are encountered.

TRANSMIT PACKET ASSEMBLY FORMAT

The following diagrams describe the format for how packets must be assembled prior to transmission for different byte ordering schemes. The various formats are selected in the Data Configuration Register.

8.0 Packet Transmission (Continued)

D15 D8	D7 D0
DA1	DA0
DA3	DA2
DA5	DA4
SA1	DA0
SA3	DA2
SA5	DA4
T/L1	T/L0
DATA 1	DATA 0

BOS = 0, WTS = 1 in Data Configuration Register.

This format is used with NSC32000, 808X type processors.

D15 D8	D7 D0
DA0	DA1
DA2	DA3
DA4	DA5
SA0	SA1
SA2	SA3
SA4	SA5
T/L0	T/L1
DATA 0	DATA 1

BOS = 1, WTS = 1 in Data Configuration Register.

This format is used with 68000 type processors.

D7 D0
DA0
DA1
DA2
DA3
DA4
DA5
SA0
SA1
SA2
SA3

BOS = 0, WTS = 0 in Data Configuration Register.

This format is used with general 8-bit CPUs.

Note: All examples above will result in a transmission of a packet in order of DA0, DA1, DA2, DA3 . . . bits within each byte will be transmitted least significant bit first.

DA = Destination Address

SA = Source Address

T/L = Type/Length Field

9.0 Remote DMA

The Remote DMA channel is used to both assemble packets for transmission, and to remove received packets from the Receive Buffer Ring. It may also be used as a general purpose slave DMA channel for moving blocks of data or commands between host memory and local buffer memory. There are three modes of operation, Remote Write, Remote Read, or Send Packet.

Two register pairs are used to control the Remote DMA, a Remote Start Address (RSAR0, RSAR1) and a Remote Byte Count (RBCR0, RBCR1) register pair. The Start Address register pair point to the beginning of the block to be moved while the Byte Count register pair are used to indicate the number of bytes to be transferred. Full handshake logic is provided to move data between local buffer memory and a bidirectional I/O port.

REMOTE WRITE

A Remote Write transfer is used to move a block of data from the host into local buffer memory. The Remote DMA will read data from the I/O port and sequentially write it to local buffer memory beginning at the Remote Start Address. The DMA address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count register reaches a count of zero.

REMOTE READ

A Remote Read transfer is used to move a block of data from local buffer memory to the host. The Remote DMA will sequentially read data from the local buffer memory, beginning at the Remote Start Address, and write data to the I/O port. The DMA address will be incremented and the Byte Counter will be decremented after each transfer. The DMA is terminated when the Remote Byte Count register reaches zero.

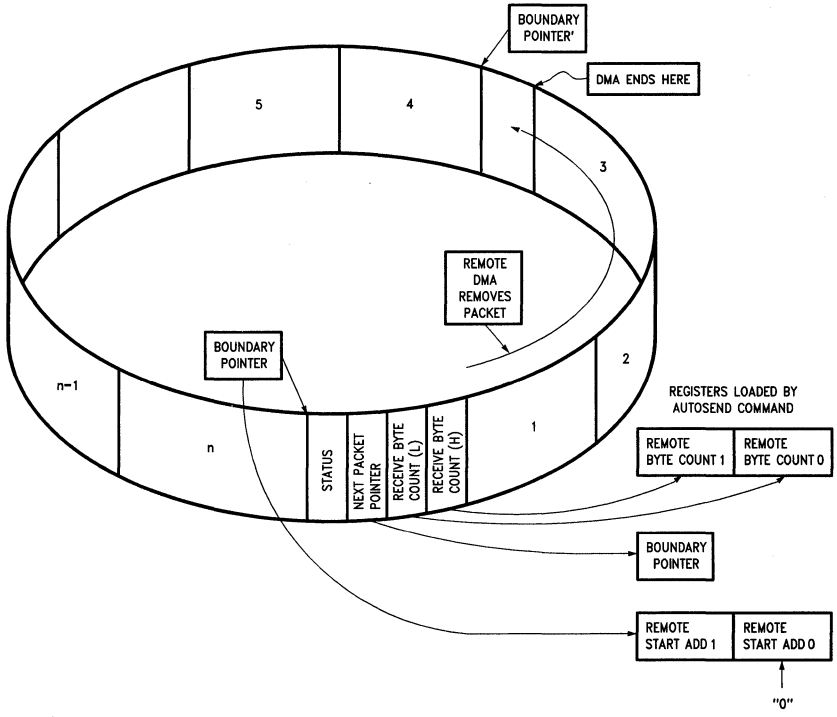
SEND PACKET COMMAND

The Remote DMA channel can be automatically initialized to transfer a single packet from the Receive Buffer Ring. The CPU begins this transfer by issuing a "Send Packet" Command. The DMA will be initialized to the value of the Boundary Pointer register and the Remote Byte Count register pair (RBCR0, RBCR1) will be initialized to the value of the Receive Byte Count fields found in the Buffer Header of each packet. After the data is transferred, the Boundary Pointer is advanced to allow the buffers to be used for new receive packets. The Remote Read will terminate when the Byte Count equals zero. The Remote DMA is then prepared to read the next packet from the Receive Buffer Ring. If the DMA pointer crosses the Page Stop register, it is reset to the Page Start Address. This allows the Remote DMA to remove packets that have wrapped around to the top of the Receive Buffer Ring.

Note: In order for the NIC to correctly execute the Autosend Command, the upper Remote Byte Count Register (RBCR1) must first be loaded with 01H.

9.0 Remote DMA (Continued)

Remote DMA Autoinitialization from Buffer Ring

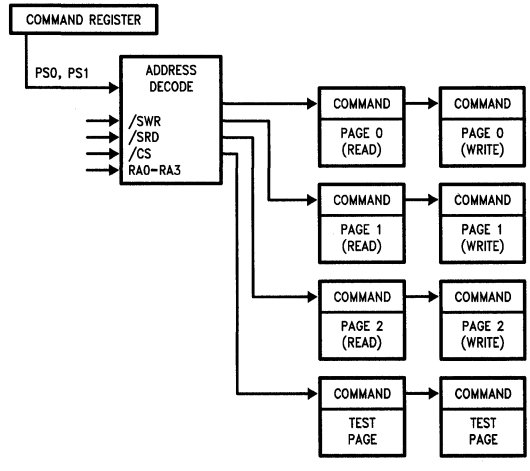


TL/F/8582–56

10.0 Internal Registers

All registers are 8-bit wide and mapped into two pages which are selected in the Command register (PS0, PS1). Pins RA0–RA3 are used to address registers within each page. Page 0 registers are those registers which are commonly accessed during NIC operation while page 1 registers are used primarily for initialization. The registers are partitioned to avoid having to perform two write/read cycles to access commonly used registers.

10.1 REGISTER ADDRESS MAPPING



TL/F/8582–11

10.0 Internal Registers (Continued)

10.2 REGISTER ADDRESS ASSIGNMENTS

Page 0 Address Assignments (PS1 = 0, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Current Local DMA Address 0 (CLDA0)	Page Start Register (PSTART)
02H	Current Local DMA Address 1 (CLDA1)	Page Stop Register (PSTOP)
03H	Boundary Pointer (BNRY)	Boundary Pointer (BNRY)
04H	Transmit Status Register (TSR)	Transmit Page Start Address (TPSR)
05H	Number of Collisions Register (NCR)	Transmit Byte Count Register 0 (TBCR0)
06H	FIFO (FIFO)	Transmit Byte Count Register 1 (TBCR1)
07H	Interrupt Status Register (ISR)	Interrupt Status Register (ISR)
08H	Current Remote DMA Address 0 (CRDA0)	Remote Start Address Register 0 (RSAR0)
09H	Current Remote DMA Address 1 (CRDA1)	Remote Start Address Register 1 (RSAR1)
0AH	Reserved	Remote Byte Count Register 0 (RBCR0)
0BH	Reserved	Remote Byte Count Register 1 (RBCR1)
0CH	Receive Status Register (RSR)	Receive Configuration Register (RCR)
0DH	Tally Counter 0 (Frame Alignment Errors) (CNTR0)	Transmit Configuration Register (TCR)
0EH	Tally Counter 1 (CRC Errors) (CNTR1)	Data Configuration Register (DCR)
0FH	Tally Counter 2 (Missed Packet Errors) (CNTR2)	Interrupt Mask Register (IMR)

Page 1 Address Assignments (PS1 = 0, PS0 = 1)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Physical Address Register 0 (PAR0)	Physical Address Register 0 (PAR0)
02H	Physical Address Register 1 (PAR1)	Physical Address Register 1 (PAR1)
03H	Physical Address Register 2 (PAR2)	Physical Address Register 2 (PAR2)
04H	Physical Address Register 3 (PAR3)	Physical Address Register 3 (PAR3)
05H	Physical Address Register 4 (PAR4)	Physical Address Register 4 (PAR4)
06H	Physical Address Register 5 (PAR5)	Physical Address Register 5 (PAR5)
07H	Current Page Register (CURR)	Current Page Register (CURR)
08H	Multicast Address Register 0 (MAR0)	Multicast Address Register 0 (MAR0)
09H	Multicast Address Register 1 (MAR1)	Multicast Address Register 1 (MAR1)
0AH	Multicast Address Register 2 (MAR2)	Multicast Address Register 2 (MAR2)
0BH	Multicast Address Register 3 (MAR3)	Multicast Address Register 3 (MAR3)
0CH	Multicast Address Register 4 (MAR4)	Multicast Address Register 4 (MAR4)
0DH	Multicast Address Register 5 (MAR5)	Multicast Address Register 5 (MAR5)
0EH	Multicast Address Register 6 (MAR6)	Multicast Address Register 6 (MAR6)
0FH	Multicast Address Register 7 (MAR7)	Multicast Address Register 7 (MAR7)

10.0 Internal Registers (Continued)

Page 2 Address Assignments (PS1 = 1, PS0 = 0)

RA0–RA3	RD	WR
00H	Command (CR)	Command (CR)
01H	Page Start Register (PSTART)	Current Local DMA Address 0 (CLDA0)
02H	Page Stop Register (PSTOP)	Current Local DMA Address 1 (CLDA1)
03H	Remote Next Packet Pointer	Remote Next Packet Pointer
04H	Transmit Page Start Address (TPSR)	Reserved
05H	Local Next Packet Pointer	Local Next Packet Pointer
06H	Address Counter (Upper)	Address Counter (Upper)
07H	Address Counter (Lower)	Address Counter (Lower)

RA0–RA3	RD	WR
08H	Reserved	Reserved
09H	Reserved	Reserved
0AH	Reserved	Reserved
0BH	Reserved	Reserved
0CH	Receive Configuration Register (RCR)	Reserved
0DH	Transmit Configuration Register (TCR)	Reserved
0EH	Data Configuration Register (DCR)	Reserved
0FH	Interrupt Mask Register (IMR)	Reserved

Note: Page 2 registers should only be accessed for diagnostic purposes. They should not be modified during normal operation.

Page 3 should never be modified.

10.3 Register Descriptions

COMMAND REGISTER (CR) 00H (READ/WRITE)

The Command Register is used to initiate transmissions, enable or disable Remote DMA operations and to select register pages. To issue a command the microprocessor sets the corresponding bit(s) (RD2, RD1, RD0, TXP). Further commands should not be issued to those bits until the initiated command is completed and the NIC has reset these bits. The remaining four bits, PS1, PS0, RD2, and STP, may be set at any time.

7	6	5	4	3	2	1	0
PS1	PS0	RD2	RD1	RD0	TXP	STA	STP

Bit	Symbol	Description
D0	STP	**STOP:** Software reset command, takes the controller offline, no packets will be received or transmitted. Any reception or transmission in progress will continue to completion before entering the reset state. To exit this state, the STP bit must be reset and the STA bit must be set high. To perform a software reset, this bit should be set high. The software reset has executed only when indicated by the RST bit in the ISR being set to a 1. **STP powers up high.**
D1	STA	**START:** This bit is used to activate the NIC after either power up, or when the NIC has been placed in a reset mode by software command or error. **STA powers up low.**
D2	TXP	**TRANSMIT PACKET:** This bit must be set to initiate transmission of a packet. TXP is internally reset either after the transmission is completed or aborted. This bit should be set only after the Transmit Byte Count and Transmit Page Start registers have been programmed.
D3, D4, D5	RD0, RD1, RD2	**REMOTE DMA COMMAND:** These three encoded bits control operation of the Remote DMA channel. RD2 can be set to abort any Remote DMA command in progress, RD2 is reset by NIC when a Remote DMA has been completed. The Remote Byte Count Registers should be cleared when a Remote DMA has been aborted. The Remote Start Addresses are not restored to the starting address if the Remote DMA is aborted. RD2 RD1 RD0 0 0 0 Not Allowed 0 0 1 Remote Read 0 1 0 Remote Write 0 1 1 Send Packet 1 X X Abort/Complete Remote DMA
D6, D7	PS0, PS1	**PAGE SELECT:** These two encoded bits select which register page is to be accessed with addresses RA0–3. PS1 PS0 0 0 Register Page 0 0 1 Register Page 1 1 0 Register Page 2 1 1 Reserved

10.3 Register Descriptions (Continued)

INTERRUPT STATUS REGISTER (ISR) 07H (READ/WRITE)

This register is accessed by the host processor to determine the cause of an interrupt. Any interrupt can be masked in the Interrupt Mask Register (IMR). Individual interrupt bits are cleared by writing a "1" into the corresponding bit of the ISR. The INT signal is active as long as any unmasked signal is set, and will not go low until all unmasked bits in this register have been cleared. The ISR must be cleared after power up by writing it with all 1's.

7	6	5	4	3	2	1	0
RST	RDC	CNT	OVW	TXE	RXE	PTX	PRX

Bit	Symbol	Description
D0	PRX	**PACKET RECEIVED:** Indicates packet received with no errors.
D1	PTX	**PACKET TRANSMITTED:** Indicates packet transmitted with no errors.
D2	RXE	**RECEIVE ERROR:** Indicates that a packet was received with one or more of the following errors: —CRC Error —Frame Alignment Error —FIFO Overrun —Missed Packet
D3	TXE	**TRANSMIT ERROR:** Set when packet transmitted with one or more of the following errors: —Excessive Collisions —FIFO Underrun
D4	OVW	**OVERWRITE WARNING:** Set when receive buffer ring storage resources have been exhausted. (Local DMA has reached Boundary Pointer).
D5	CNT	**COUNTER OVERFLOW:** Set when MSB of one or more of the Network Tally Counters has been set.
D6	RDC	**REMOTE DMA COMPLETE:** Set when Remote DMA operation has been completed.
D7	RST	**RESET STATUS:** Set when NIC enters reset state, reset when a Start Command is issued to the CR. Writing to this bit has no effect. **NOTE:** This bit does not generate an interrupt, it is merely a status indicator.

10.3 Register Descriptions (Continued)

INTERRUPT MASK REGISTER (IMR) 0FH (WRITE)

The Interrupt Mask Register is used to mask interrupts. Each interrupt mask bit corresponds to a bit in the Interrupt Status Register (ISR). If an interrupt mask bit is set an interrupt will be issued whenever the corresponding bit in the ISR is set. If any bit in the IMR is set low, an interrupt will not occur when the bit in the ISR is set. **The IMR powers up all zero.**

7	6	5	4	3	2	1	0
—	RDCE	CNTE	OVWE	TXEE	RXEE	PTXE	PRXE

Bit	Symbol	Description
D0	PRXE	**PACKET RECEIVED INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet received.
D1	PTXE	**PACKET TRANSMITTED INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet is transmitted.
D2	RXEE	**RECEIVE ERROR INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet received with error.
D3	TXEE	**TRANSMIT ERROR INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when packet transmission results in error.
D4	OVWE	**OVERWRITE WARNING INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when Buffer Management Logic lacks sufficient buffers to store incoming packet.
D5	CNTE	**COUNTER OVERFLOW INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when MSB of one or more of the Network Statistics counters has been set.
D6	RDCE	**DMA COMPLETE INTERRUPT ENABLE** 0: Interrupt Disabled 1: Enables Interrupt when Remote DMA transfer has been completed.
D7	reserved	reserved

10.3 Register Descriptions (Continued)

DATA CONFIGURATION REGISTER (DCR) 0EH (WRITE)

This Register is used to program the NIC for 8- or 16-bit memory interface, select byte ordering in 16-bit applications and establish FIFO threshholds. **The DCR must be initialized prior to loading the Remote Byte Count Registers. LAS is set on power up.**

7	6	5	4	3	2	1	0
—	FT1	FT0	ARM	BMS	LAS	BOS	WTS

Bit	Symbol	Description
D0	WTS	**WORD TRANSFER SELECT** 0: Selects byte-wide DMA transfers 1: Selects word-wide DMA transfers ; WTS establishes byte or word transfers for both Remote and Local DMA transfers
D1	BOS	**BYTE ORDER SELECT** 0: MS byte placed on AD15–AD8 and LS byte on AD7–AD0. (32000, 8086) 1: MS byte placed on AD7–AD0 and LS byte on AD15–AD8. (68000) ; ignored when WTS is low
D2	LAS	**LONG ADDRESS SELECT** 0: Dual 16-bit DMA mode. 1: Single 32-bit DMA mode. ; When LAS is high, the contents of the Remote DMA registers RSAR0,1 are issued as A16–A31 Power up high.
D3	BMS	**BURST MODE SELECT** 0: All Local DMA transfers continue until FIFO is emptied or filled 1: All Local DMA transfers continue until the number of bytes programmed in bits FT0, FT1 have been transferred.
D4	AR	**AUTOINITIALIZE REMOTE** 0: Send Command not executed, all packets removed from Buffer Ring under program control. 1: Send Command executed, Remote DMA autoinitialized to remove packets from Buffer ring.
D5, D6	FT0, FT1	**FIFO THRESHHOLD SELECT:** Encoded FIFO threshhold. Establishes point at which bus is requested when filling or emptying the FIFO. **Note:** FIFO threshold setting determines the DMA burst length. RECEIVE THRESHOLDS FT1 FT0 Word Wide Byte Wide 0 0 1 Word 2 Bytes 0 1 2 Words 4 Bytes 1 0 4 Words 8 Bytes 1 1 6 Words 12 Bytes **NOTE: TRANSMIT THRESHOLDS = 16 bytes less receive threshold.**

10.3 Register Descriptions (Continued)

TRANSMIT CONFIGURATION REGISTER (TCR) 0DH (WRITE)

The transmit configuration establishes the actions of the transmitter section of the NIC during transmission of a packet on the network. **LB1 and LB0 which select loopback mode power up as 0.**

7	6	5	4	3	2	1	0
—	—	—	OFST	ATD	LB1	LB0	CRC

Bit	Symbol	Description
D0	CRC	**INHIBIT CRC** 0: CRC appended by transmitter 1: CRC inhibited by transmitter ; In loopback mode CRC can be enabled or disabled to test the CRC logic.
D1, D2	LB0, LB1	**ENCODED LOOPBACK CONTROL:** These encoded configuration bits set the type of loopback that is to be performed. Note that loopback in mode 2 sets the LPBK pin high, this places the SNI in loopback mode. LB1 LB2 Mode 0 0 0 Normal Operation (LPBK = 0) Mode 1 0 1 Internal Loopback (LPBK = 0) Mode 2 1 0 External Loopback (LPBK = 1) Mode 3 1 1 External Loopback (LPBK = 0)
D3	ATD	**AUTO TRANSMIT DISABLE:** This bit allows another station to disable the NIC's transmitter by transmission of a particular multicast packet. The transmitter can be re-enabled by resetting this bit or by reception of a second particular multicast packet. 0: Normal Operation 1: Reception of multicast address hashing to bit 62 disables transmitter, reception of multicast address hashing to bit 63 enables transmitter.
D4	OFST	**COLLISION OFFSET ENABLE:** This bit modifies the backoff algorithm to allow prioritization of nodes. 0: Backoff Logic implements normal algorithm. 1: Forces Backoff algorithm modification to 0 to $2^{\min(3+n,10)}$ slot times for first three collisions, then follows standard backoff. (For first three collisions station has higher average backoff delay making a low priority mode.)
D5	reserved	reserved
D6	reserved	reserved
D7	reserved	reserved

10.3 Register Descriptions (Continued)

TRANSMIT STATUS REGISTER (TSR) 04H (READ)

This register records events that occur on the media during transmission of a packet. It is cleared when the next transmission is initiated by the host. All bits remain low unless the event that corresponds to a particular bit occurs during transmission. Each transmission should be followed by a read of this register. The contents of this register are not specified until after the first transmission.

7	6	5	4	3	2	1	0
OWC	CDH	FU	CRS	ABT	COL	DFR	PTX

Bit	Symbol	Description
D0	PTX	**PACKET TRANSMITTED:** Indicates transmission without error. (No excessive collisions or FIFO underrun) (ABT = "0", FU = "0").
D1	DFR	**NON DEFERRED TRANSMISSION:** Indicates that transmission proceeded without deferring. This bit is set only for the first transmission and may not accurately reflect deferral for retransmission during collisions.
D2	COL	**TRANSMIT COLLIDED:** Indicates that the transmission collided at least once with another station on the network. The number of collisions is recorded in the Number of Collisions Registers (NCR).
D3	ABT	**TRANSMIT ABORTED:** Indicates the NIC aborted transmission because of excessive collisions. (Total number of transmissions including original transmission attempt equals 16).
D4	CRS	**CARRIER SENSE LOST:** This bit is set when carrier is lost during transmission of the packet. Carrier Sense is monitored from the end of Preamble/Synch until TXEN is dropped. Transmission is not aborted on loss of carrier.
D5	FU	**FIFO UNDERRUN:** If the NIC cannot gain access of the bus before the FIFO empties, this bit is set. Transmission of the packet will be aborted.
D6	CDH	**CD HEARTBEAT:** Failure of the transceiver to transmit a collision signal after transmission of a packet will set this bit. The Collision Detect (CD) heartbeat signal must commence during the first 6.4 μs of the Interframe Gap following a transmission. In certain collisions, the CD Heartbeat bit will be set even though the transceiver is not performing the CD heartbeat test.
D7	OWC	**OUT OF WINDOW COLLISION:** Indicates that a collision occurred after a slot time (51.2 μs). Transmission will not be aborted.

10.3 Register Descriptions (Continued)

RECEIVE CONFIGURATION REGISTER (RCR) 0CH (WRITE)

This register determines operation of the NIC during reception of a packet and is used to program what types of packets to accept.

7	6	5	4	3	2	1	0
—	—	MON	PRO	AM	AB	AR	SEP

Bit	Symbol	Description
D0	SEP	**SAVE ERRORED PACKETS** 0: Packets with receive errors are rejected. 1: Packets with receive errors are accepted. Receive errors are CRC and Frame Alignment errors.
D1	AR	**ACCEPT RUNT PACKETS:** This bit allows the receiver to accept packets that are smaller than 64 bytes. The packet must be at least 8 bytes long to be accepted as a runt. 0: Packets with fewer than 64 bytes rejected. 1: Packets with fewer than 64 bytes accepted.
D2	AB	**ACCEPT BROADCAST:** Enables the receiver to accept a packet with an all 1's destination address. 0: Packets with broadcast destination address rejected. 1: Packets with broadcast destination address accepted.
D3	AM	**ACCEPT MULTICAST:** Enables the receiver to accept a packet with a multicast address, all multicast addresses must pass the hashing array. 0: Packets with multicast destination address not checked. 1: Packets with multicast destination address checked.
D4	PRO	**PROMISCUOUS PHYSICAL:** Enables the receiver to accept all packets with a physical address. 0: Physical address of node must match the station address programmed in PAR0–PAR5. 1: All packets with physical addresses accepted.
D5	MON	**MONITOR MODE:** Enables the receiver to check addresses and CRC on incoming packets without buffering to memory. The Missed Packet Tally counter will be incremented for each recognized packet. 0: Packets buffered to memory. 1: Packets checked for address match, good CRC and Frame Alignment but not buffered to memory.
D6	reserved	reserved
D7	reserved	reserved

Note: D2 and D3 are "OR'd" together, i.e., if D2 and D3 are set the NIC will accept broadcast and multicast addresses as well as its own physical address. To establish full promiscuous mode, bits D2, D3, and D4 should be set. In addition the multicast hashing array must be set to all 1's in order to accept all multicast addresses.

10.3 Register Descriptions (Continued)

RECEIVE STATUS REGISTER (RSR) 0CH (READ)

This register records status of the received packet, including information on errors and the type of address match, either physical or multicast. The contents of this register are written to buffer memory by the DMA after reception of a good packet. If packets with errors are to be saved the receive status is written to memory at the head of the erroneous packet if an erroneous packet is received. If packets with errors are to be rejected the RSR will not be written to memory. The contents will be cleared when the next packet arrives. CRC errors, Frame Alignment errors and missed packets are counted internally by the NIC which relinquishes the Host from reading the RSR in real time to record errors for Network Management Functions. The contents of this register are not specified until after the first reception.

7	6	5	4	3	2	1	0
DFR	DIS	PHY	MPA	FO	FAE	CRC	PRX

Bit	Symbol	Description
D0	PRX	**PACKET RECEIVED INTACT:** Indicates packet received without error. (Bits CRC, FAE, FO, and MPA are zero for the received packet.)
D1	CRC	**CRC ERROR:** Indicates packet received with CRC error. Increments Tally Counter (CNTR1). This bit will also be set for Frame Alignment errors.
D2	FAE	**FRAME ALIGNMENT ERROR:** Indicates that the incoming packet did not end on a byte boundary and the CRC did not match at last byte boundary. Increments Tally Counter (CNTR0).
D3	FO	**FIFO OVERRUN:** This bit is set when the FIFO is not serviced causing overflow during reception. Reception of the packet will be aborted.
D4	MPA	**MISSED PACKET:** Set when packet intended for node cannot be accepted by NIC because of a lack of receive buffers or if the controller is in monitor mode and did not buffer the packet to memory. Increments Tally Counter (CNTR2).
D5	PHY	**PHYSICAL/MULTICAST ADDRESS:** Indicates whether received packet had a physical or multicast address type. 0: Physical Address Match 1: Multicast/Broadcast Address Match
D6	DIS	**RECEIVER DISABLED:** Set when receiver disabled by entering Monitor mode. Reset when receiver is re-enabled when exiting Monitor mode.
D7	DFR	**DEFERRING:** Set when the Interframe Gap state machine is deferring, if the tranceiver has asserted the CD line as a result of the jabber, this bit will stay set indicating the jabber condition.

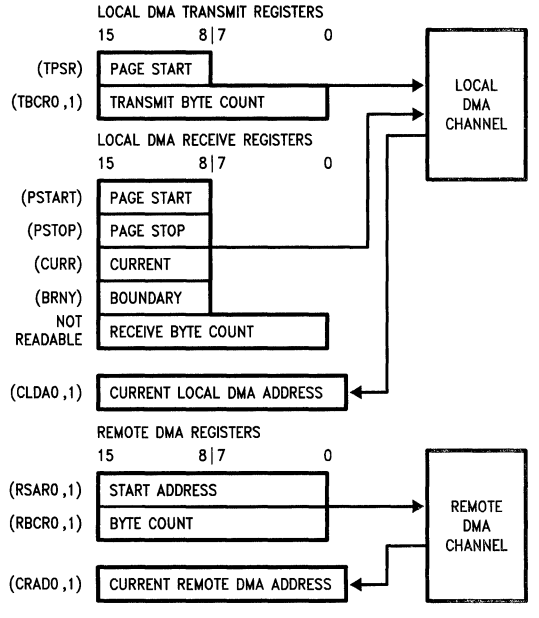
Note: Following coding applies to CRC and FAE bits

FAE	CRC	Type of Error
0	0	No Error (Good CRC and <6 Dribble Bits)
0	1	CRC Error
1	0	Illegal, will not occur
1	1	Frame Alignment Error and CRC Error

10.0 Internal Registers (Continued)

10.4 DMA REGISTERS

DMA Registers



TL/F/8582–12

The DMA Registers are partitioned into three groups; Transmit, Receive and Remote DMA Registers. The Transmit registers are used to initialize the Local DMA Channel for transmission of packets while the Receive registers are used to initialize the Local DMA Channel for packet Reception. The Page Stop, Page Start, Current and Boundary registers are used by the Buffer Management Logic to supervise the Receive Buffer Ring. The Remote DMA Registers are used to initialize the Remote DMA.

Note: In the figure above, registers are shown as 8 or 16 bits wide. Although some registers are 16-bit internal registers, all registers are accessed as 8-bit registers. Thus the 16-bit Transmit Byte Count Register is broken into two 8-bit registers, TBCR0 and TBCR1. Also TPSR, PSTART, PSTOP, CURR and BNRY only check or control the upper 8 bits of address information on the bus. Thus they are shifted to positions 15-8 in the diagram above.

10.5 TRANSMIT DMA REGISTERS

TRANSMIT PAGE START REGISTER (TPSR)

This register points to the assembled packet to be transmitted. Only the eight higher order addresses are specified since all transmit packets are assembled on 256 byte page boundaries. The bit assignment is shown below. The values placed in bits D7–D0 will be used to initialize the higher order address (A8–A15) of the Local DMA for transmission. The lower order bits (A7–A0) are initialized to zero.

Bit Assignment

	7	6	5	4	3	2	1	0
TPSR	A15	A14	A13	A12	A11	A10	A9	A8

(A7–A0 Initialized to zero)

TRANSMIT BYTE COUNT REGISTER 0,1 (TBCR0, TBCR1)

These two registers indicate the length of the packet to be transmitted in bytes. The count must include the number of bytes in the source, destination, length and data fields. The maximum number of transmit bytes allowed is 64k bytes. The NIC will not truncate transmissions longer than 1500 bytes. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
TBCR1	L15	L14	L13	L12	L11	L10	L9	L8

	7	6	5	4	3	2	1	0
TBCR0	L7	L6	L5	L4	L3	L2	L1	L0

10.6 LOCAL DMA RECEIVE REGISTERS

PAGE START STOP REGISTERS (PSTART, PSTOP)

The Page Start and Page Stop Registers program the starting and stopping address of the Receive Buffer Ring. Since the NIC uses fixed 256 byte buffers aligned on page boundaries only the upper eight bits of the start and stop address are specified.

PSTART,PSTOP bit assignment

	7	6	5	4	3	2	1	0
PSTART, PSTOP	A15	A14	A13	A12	A11	A10	A9	A8

BOUNDARY (BNRY) REGISTER

This register is used to prevent overflow of the Receive Buffer Ring. Buffer management compares the contents of this register to the next buffer address when linking buffers together. If the contents of this register match the next buffer address the Local DMA operation is aborted.

	7	6	5	4	3	2	1	0
BNRY	A15	A14	A13	A12	A11	A10	A9	A8

5

10.0 Internal Registers (Continued)

CURRENT PAGE REGISTER (CURR)

This register is used internally by the Buffer Management Logic as a backup register for reception. CURR contains the address of the first buffer to be used for a packet reception and is used to restore DMA pointers in the event of receive errors. This register is initialized to the same value as PSTART and should not be written to again unless the controller is Reset.

	7	6	5	4	3	2	1	0
CURR	A15	A14	A13	A12	A11	A10	A9	A8

CURRENT LOCAL DMA REGISTER 0,1 (CLDA0,1)

These two registers can be accessed to determine the current Local DMA Address.

	7	6	5	4	3	2	1	0
CLDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CLDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.7 REMOTE DMA REGISTERS

REMOTE START ADDRESS REGISTERS (RSAR0,1)

Remote DMA operations are programmed via the Remote Start Address (RSAR0,1) and Remote Byte Count (RBCR0,1) registers. The Remote Start Address is used to point to the start of the block of data to be transferred and the Remote Byte Count is used to indicate the length of the block (in bytes).

	7	6	5	4	3	2	1	0
RSAR1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
RSAR0	A7	A6	A5	A4	A3	A2	A1	A0

6.4.3.2 REMOTE BYTE COUNT REGISTERS (RBCR0,1)

	7	6	5	4	3	2	1	0
RBCR1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8

	7	6	5	4	3	2	1	0
RBCR0	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0

Note:

RSAR0 programs the start address bits A0–A7.
RSAR1 programs the start address bits A8–A15.
Address incremented by two for word transfers, and by one for byte transfers.
Byte Count decremented by two for word transfers and by one for byte transfers.
RBCR0 programs LSB byte count.
RBCR1 programs MSB byte count.

CURRENT REMOTE DMA ADDRESS (CRDA0, CRDA1)

The Current Remote DMA Registers contain the current address of the Remote DMA. The bit assignment is shown below:

	7	6	5	4	3	2	1	0
CRDA1	A15	A14	A13	A12	A11	A10	A9	A8

	7	6	5	4	3	2	1	0
CRDA0	A7	A6	A5	A4	A3	A2	A1	A0

10.8 PHYSICAL ADDRESS REGISTERS (PAR0–PAR5)

The physical address registers are used to compare the destination address of incoming packets for rejecting or accepting packets. Comparisons are performed on a byte-wide basis. The bit assignment shown below relates the sequence in PAR0–PAR5 to the bit sequence of the received packet.

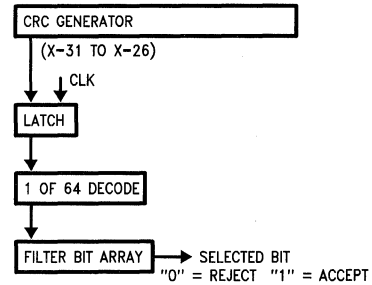
	D7	D6	D5	D4	D3	D2	D1	D0
PAR0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
PAR1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
PAR2	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
PAR3	DA31	DA30	DA29	DA28	DA27	DA26	DA25	DA24
PAR4	DA39	DA38	DA37	DA36	DA35	DA34	DA33	DA32
PAR5	DA47	DA46	DA45	DA44	DA43	DA42	DA41	DA40

	Destination Address								Source
P/S	DA0	DA1	DA2	DA3	.....	DA46	DA47	SA0	...

Note:
P/S = Preamble, Synch
DA0 = Physical/Multicast Bit

10.9 MULTICAST ADDRESS REGISTERS (MAR0–MAR7)

The multicast address registers provide filtering of multicast addresses hashed by the CRC logic. All destination addresses are fed through the CRC logic and as the last bit of the destination address enters the CRC, the 6 most significant bits of the CRC generator are latched. These 6 bits are then decoded by a 1 of 64 decode to index a unique filter bit (FB0–63) in the multicast address registers. If the filter bit selected is set, the multicast packet is accepted. The system designer would use a program to determine which filter bits to set in the multicast registers. All multicast filter bits that correspond to multicast address accepted by the node are then set to one. To accept all multicast packets all of the registers are set to all ones. **Note: Although the hashing algorithm does not guarantee perfect filtering of multicast address, it will perfectly filter up to 64 multicast addresses if these addresses are chosen to map into unique locations in the multicast filter.**



TL/F/8582–14

10.0 Internal Registers (Continued)

	D7	D6	D5	D4	D3	D2	D1	D0
MAR0	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
MAR1	FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8
MAR2	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
MAR3	FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24
MAR4	FB39	FB38	FB37	FB36	FB35	FB34	FB33	FB32
MAR5	FB47	FB46	FB45	FB44	FB43	FB42	FB41	FB40
MAR6	FB55	FB54	FB53	FB52	FB51	FB50	FB49	FB48
MAR7	FB63	FB62	FB61	FB60	FB59	FB58	FB57	FB56

If address Y is found to hash to the value 32 (20H), then FB32 in MAR4 should be initialized to "1". This will cause the NIC to accept any multicast packet with the address Y.

NETWORK TALLY COUNTERS

Three 8-bit counters are provided for monitoring the number of CRC errors, Frame Alignment Errors and Missed Packets. The maximum count reached by any counter is 192 (C0H). These registers will be cleared when read by the CPU. The count is recorded in binary in CT0–CT7 of each Tally Register.

Frame Alignment Error Tally (CNTR0)

This counter is incremented every time a packet is received with a Frame Alignment Error. The packet must have been recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR0	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

CRC Error Tally (CNTR1)

This counter is incremented every time a packet is received with a CRC error. The packet must first be recognized by the address recognition logic. The counter is cleared after it is read by the processor.

	7	6	5	4	3	2	1	0
CNTR1	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

Frames Lost Tally Register (CNTR2)

This counter is incremented if a packet cannot be received due to lack of buffer resources. In monitor mode, this counter will count the number of packets that pass the address recognition logic.

	7	6	5	4	3	2	1	0
CNTR2	CT7	CT6	CT5	CT4	CT3	CT2	CT1	CT0

FIFO

This is an eight bit register that allows the CPU to examine the contents of the FIFO after loopback. The FIFO will contain the last 8 data bytes transmitted in the loopback packet. Sequential reads from the FIFO will advance a pointer in the FIFO and allow reading of all 8 bytes.

	7	6	5	4	3	2	1	0
FIFO	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Note: The FIFO should only be read when the NIC has been programmed in loopback mode.

NUMBER OF COLLISIONS (NCR)

This register contains the number of collisions a node experiences when attempting to transmit a packet. If no collisions are experienced during a transmission attempt, the COL bit of the TSR will not be set and the contents of NCR will be zero. If there are excessive collisions, the ABT bit in the TSR will be set and the contents of NCR will be zero. The NCR is cleared after the TXP bit in the CR is set.

	7	6	5	4	3	2	1	0
NCR	—	—	—	—	NC3	NC2	NC1	NC0

11.0 Initialization Procedures

The NIC must be initialized prior to transmission or reception of packets from the network. Power on reset is applied to the NIC's reset pin. This clears/sets the following bits:

Register	Reset Bits	Set Bits
Command Register (CR)	TXP, STA	RD2, STP
Interrupt Status (ISR)		RST
Interrupt Mask (IMR)	All Bits	
Data Control (DCR)		LAS
Transmit Config. (TCR)	LB1, LB0	

The NIC remains in its reset state until a Start Command is issued. This guarantees that no packets are transmitted or received and that the NIC remains a bus slave until all appropriate internal registers have been programmed. After initialization the STP bit of the command register is reset and packets may be received and transmitted.

Initialization Sequence

The following initialization procedure is mandatory.

1) Program Command Register for Page 0
2) Initialize DCR with proper configuration
3) Clear Remote Byte Count Registers (RBCR0,1) (Write with 0's)
4) Initialize RCR to monitor mode
5) Initialize TCR
6) Initialization of Receive Buffer Ring: PAGE START = BNRY (initialize Current Page Register to same value.) Page Stop initialized to point to end of Buffer Ring.
7) Write all 1's to ISR to clear
8) Initialize IMR
9) Program Command Register for Page 1
   Initialize Physical Address and Multicast Filter
   i) The physical address registers (PAR0–PAR5) are initialized to the value of the node's assigned physical address.
   ii) The multicast filter (MAR0–MAR7) is programmed according to the specific multicast addresses that are to be recognized by the node.
   iii) Program Current Page Register = PAGE START
10) Program Command Register for Page 0
    Reset STP, STA to place NIC online.
11) Program RCR

5

11.0 Initialization Procedures

(Continued)

Before receiving packets, the user must specify the location of the Receive Buffer Ring . This is programmed in the Page Start and Page Stop Registers. In addition, the Boundary and Current Page Registers must be initialized to the value of the Page Start Register. These registers will be modified during reception of packets.

12.0 Loopback Diagnostics

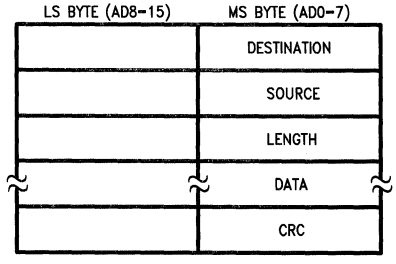
Three forms of local loopback are provided on the NIC. The user has the ability to loopback through the deserializer on the DP8390 NIC, through the DP8391 SNI, and to the coax to check the link through the tranceiver circuitry. **Because of the half duplex architecture of the NIC, loopback testing is a special mode of operation with the following restrictions:**

Restrictions During Loopback

The FIFO is split into two halves, one used for transmission the other for reception. Only 8 bit fields can be fetched from memory so two tests are required for 16 bit systems to verify integrity of the entire data path. In 8 bit systems only empty/fill is recommended (see DCR descriptions). During loopback the maximum latency from the assertion of BREQ to BACK is 5.6 μs. Systems that wish to use the loopback test yet do not meet this latency can limit the loopback packet to 7 bytes without experiencing underflow. Only the last 8 bytes of the loopback packet are retained in the FIFO. The last 8 bytes can be read through the FIFO register which will advance through the FIFO to allow reading the receive packet sequentially.

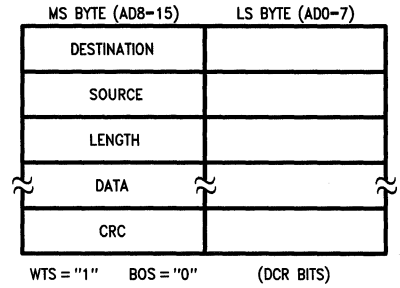
DESTINATION ADDRESS	= (6 bytes) Station Physical Address
SOURCE ADDRESS	|
LENGTH	2 bytes
DATA	= 46 to 1500 bytes
CRC	Appended by NIC if CRC = "0" in TCR

When in word-wide mode with Byte Order Select set, the loopback packet must be assembled in the even byte locations as shown below. (The loopback only operates with byte wide transfers.)



TL/F/8582–15

When in word-wide mode with Byte Order Select low, the following format must be used for the loopback packet.



TL/F/8582–16

Note: When using loopback in word mode 2n bytes must be programmed in TBCR0, 1. Where n = actual number of bytes assembled in even or odd location.

To initiate a loopback the user first assembles the loopback packet then selects the type of loopback using the Transmit Configuration register bits LB0, LB1. The transmit configuration register must also be set to enable or disable CRC generation during transmission. The user then issues a normal transmit command to send the packet. During loopback the receiver checks for an address match and if CRC bit in the TCR is set, the receiver will also check the CRC. The last 8 bytes of the loopback packet are buffered and can be read out of the FIFO using the FIFO read port.

Loopback Modes

MODE 1: Loopback Through the Controller (LB1 = 0, LB0 = 1).

If the loopback is through the NIC then the serializer is simply linked to the deserializer and the receive clock is derived from the transmit clock.

MODE 2: Loopback Through the SNI (LB1 = 1, LB0 = 0).

If the loopback is to be performed through the SNI, the NIC provides a control (LPBK) that forces the SNI to loopback all signals.

MODE 3: Loopback to Coax (LB1 = 1, LB0 = 1).

Packets can be transmitted to the coax in loopback mode to check all of the transmit and receive paths and the coax itself.

Note: In MODE 1, CRS and COL lines are masked. In MODE 2, COL is masked and in MODE 3 CRS and COL are not masked. It is not possible to go directly between the loopback modes, it is necessary to return to normal operation (00H) when changing modes.

Reading the Loopback Packet

The last eight bytes of a received packet can be examined by 8 consecutive reads of the FIFO register. The FIFO pointer is incremented after the rising edge of the CPU's read strobe by internally synchronizing and advancing the pointer. This may take up to four bus clock cycles, if the pointer has not been incremented by the time the CPU reads the FIFO register again, the NIC will insert wait states

12.0 Loopback Diagnostics (Continued)

Alignment of the Received Packet in the FIFO

Reception of the packet in the FIFO begins at location zero, after the FIFO pointer reaches the last location in the FIFO, the pointer wraps to the top of the FIFO overwriting the previously received data. This process continues until the last byte is received. The NIC then appends the received byte count in the next two locations of the FIFO. The contents of the Upper Byte Count are also copied to the next FIFO location. The number of bytes used in the loopback packet determines the alignment of the packet in the FIFO. The alignment for a 64 byte packet is shown below.

FIFO LOCATION	FIFO CONTENTS	
0	LOWER BYTE COUNT	→ First Byte Read
1	UPPER BYTE COUNT	→ Second Byte Read
2	UPPER BYTE COUNT	•
3	LAST BYTE	•
4	CRC1	•
5	CRC2	•
6	CRC3	•
7	CRC4	→ Last Byte Read

For the following alignment in the FIFO the packet length should be (N × 8) + 5 Bytes. Note that if the CRC bit in the TCR is set, CRC will not be appended by the transmitter. If the CRC is appended by the transmitter, the last four bytes, bytes N-3 to N, correspond to the CRC.

FIFO LOCATION	FIFO CONTENTS	
0	BYTE N-4	→ First Byte Read
1	BYTE N-3 (CRC1)	AR Second Byte Read
2	BYTE N-2 (CRC2)	•
3	BYTE N-1 (CRC3)	•
4	BYTE N (CRC4)	•
5	LOWER BYTE COUNT	•
6	UPPER BYTE COUNT	→ Last Byte Read
7	UPPER BYTE COUNT	

LOOPBACK TESTS

Testing CRC

If CRC = 0 in the TCR, the NIC computes and appends a 4 byte FCS field to the packet as in normal operation. The CRC will not be verified during reception in loopback mode. The CRC must be read from the FIFO and verified by comparison to a previously computed value.

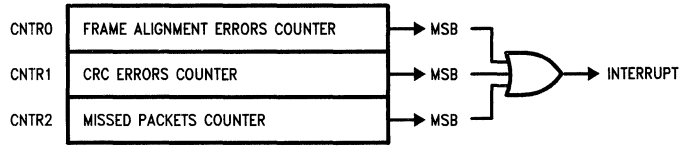
If CRC = 1, the NIC will not append a 4 byte FCS field. The user must supply a pre-calculated CRC value and append it to the transmitted packet.

NETWORK MANAGEMENT FUNCTIONS

Network management capabilities are required for maintenance and planning of a local area network. The NIC supports the minimum requirement for network management in hardware, the remaining requirements can be met with software counts. There are three events that software alone could not track during reception of packets: CRC errors, Frame Alignment errors, and missed packets.

Since errored packets can be rejected, the status associated with these packets is lost unless the CPU can access the Receive Status Register before the next packet arrives. In situations where another packet arrives very quickly, the CPU may have no opportunity to do this. The NIC counts the number of packets with CRC errors and Frame Alignment errors. 8-bit counters have been selected to reduce overhead. The counters will generate interrupts whenever their MSBs are set so that a software routine can accumulate the network statistics and reset the counters before overflow occurs. The counters are sticky so that when they reach a count of 192 (C0H) counting is halted. An additional counter is provided to count the number of packets DP8390 NIC misses due to buffer overflow or being offline.

The structure of the counters is shown below:



TL/F/8582–17

Additional information required for network management is available in the Receive and Transmit Status registers. Transmit status is available after each transmission for information regarding events during transmission.

Typically, the following statistics might be gathered in software:

Traffic:
- Frames Sent OK
- Frames Received OK
- Multicast Frames Received
- Packets Lost Due to Lack of Resources
- Retries/Packet

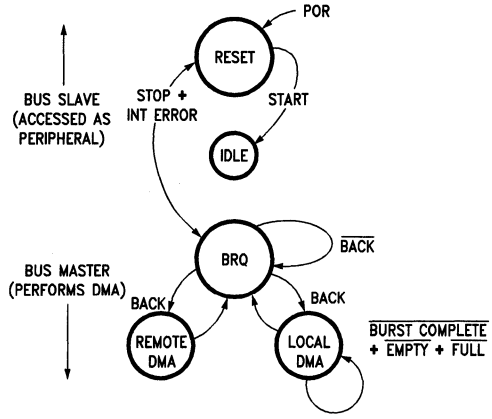
Errors:
- CRC Errors
- Alignment Errors
- Excessive Collisions
- Packet with Length Errors
- Heartbeat Failure

5

13.0 Bus Arbitration and Timing

The NIC operates in three possible modes:

- BUS MASTER (WHILE PERFORMING DMA)
- BUS SLAVE (WHILE BEING ACCESSED BY CPU)
- IDLE



TL/F/8582–18

The NIC powers up as a bus slave in the Reset State, the receiver and transmitter are both disabled in this state. The reset state can be reentered under three conditions, soft reset (Stop Command), hard reset (RESET input) or an error that shuts down the receiver or transmitter (FIFO underflow or oveflow, receive buffer ring overflow). After initialization of registers, the NIC is issued a Start command and the NIC enters Idle state. Until the DMA is required the NIC remains in an idle state. The idle state is exited by a request from the FIFO in the case of receive or transmit, or from the Remote/ DMA in the case of Remote DMA operation. After acquiring the bus in a BREQ/BACK handshake the Remote or Local DMA transfer is completed and the NIC reenters the idle state.

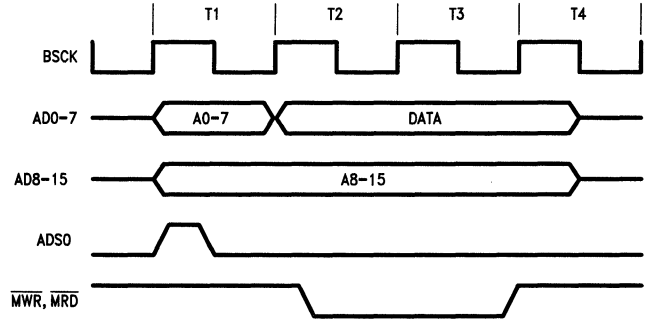
DMA TRANSFERS TIMING

The DMA can be programmed for the following types of transfers:

16 Bit Address, 8 bit Data Transfer
16 Bit Address, 16 bit Data Transfer
32 Bit Address, 8 bit Data Transfer
32 Bit Address, 16 bit Data Transfer

All DMA transfers use BSCK for timing. 16 Bit Address modes require 4 BSCK cycles as shown below:

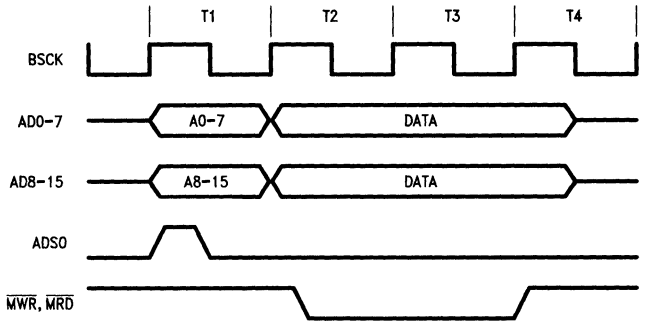
16 Bit Address, 8 Bit Data



TL/F/8582–19

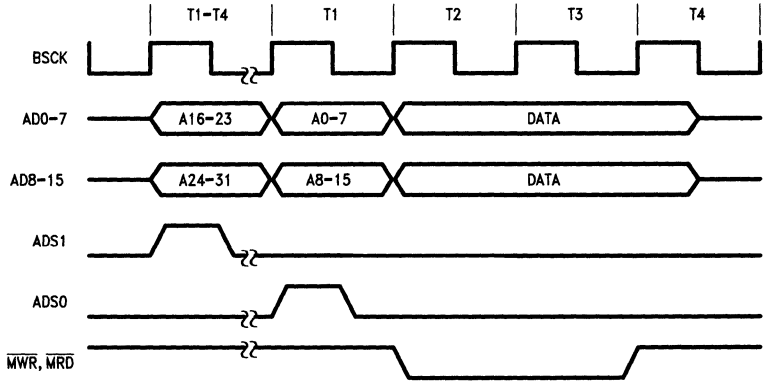
13.0 Bus Arbitration and Timing (Continued)

16 Bit Address, 16 Bit Data



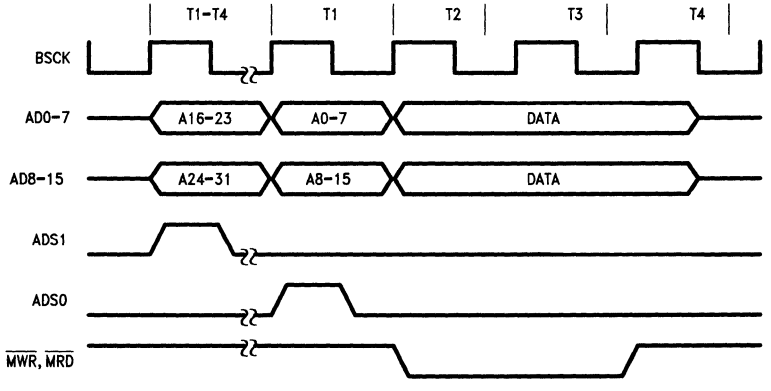
TL/F/8582-20

32 Bit Address, 8 Bit Data



TL/F/8582-21

32 Bit Address, 16 Bit Data



TL/F/8582-22

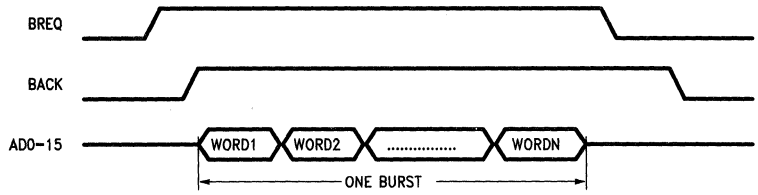
5

13.0 Bus Arbitration and Timing (Continued)

When in 32 bit mode four additional BSCK cycles are required per burst. The first bus cycle (T1′–T4′) of each burst is used to output the upper 16 bit addresses. This 16 bit address is programmed in RSAR0 and RSAR1 and points to a 64k page of system memory. All transmitted or received packets are constrained to reside within this 64k page.

FIFO BURST CONTROL

All Local DMA transfers are burst transfers, once the DMA requests the bus and the bus is acknowledged, the DMA will transfer an exact burst of bytes programmed in the Data Configuration Register (DCR) then relinquish the bus. If there are remaining bytes in the FIFO the next burst will not be initiated until the FIFO threshold is exceeded. If desired the DMA can empty/fill the FIFO when it acquires the bus. If BACK is removed during the transfer, the burst transfer will be aborted. **(DROPPING BACK DURING A DMA CYCLE IS NOT RECOMMENDED.)**

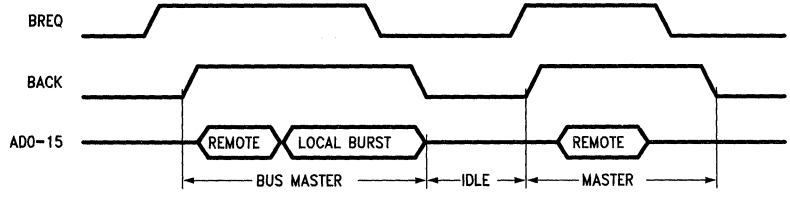


TL/F/8582–23

where N = 1, 2, 4, or 6 Words or N = 2, 4, 8, or 12 Bytes when in byte mode

INTERLEAVED LOCAL OPERATION

If a remote DMA transfer is initiated or in progress when a packet is being received or transmitted, the Remote DMA transfer will be interrupted for higher priority Local DMA transfers. When the Local DMA transfer is completed the Remote DMA will rearbitrate for the bus and continue its transfers. This is illustrated below:



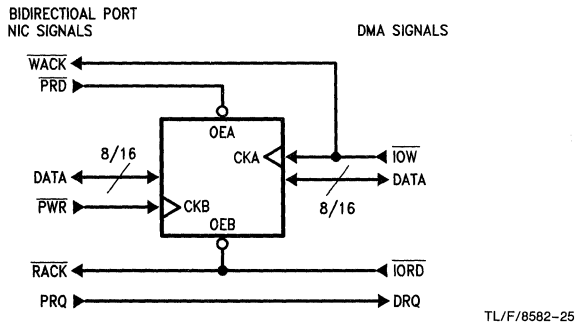
TL/F/8582–24

Note that if the FIFO requires service while a remote DMA is in progress, BREQ is not dropped and the Local DMA burst is appended to the Remote Transfer. When switching from a local transfer to a remote transfer, however, BREQ is dropped and raised again. This allows the CPU or other devices to fairly contend for the bus.

REMOTE DMA-BIDIRECTIONAL PORT CONTROL

The Remote DMA transfers data between the local buffer memory and a bidirectional port (memory to I/O transfer). This transfer is arbited on a byte by byte basis versus the burst transfer used for Local DMA transfers. This bidirectional port is also read/written by the host. All transfers through this port are asynchronous. At any one time transfers are limited to one direction, either from the port to local buffer memory (Remote Write) or from local buffer memory to the port (Remote Read).

Bus Handshake Signals for Remote DMA Transfers



TL/F/8582–25

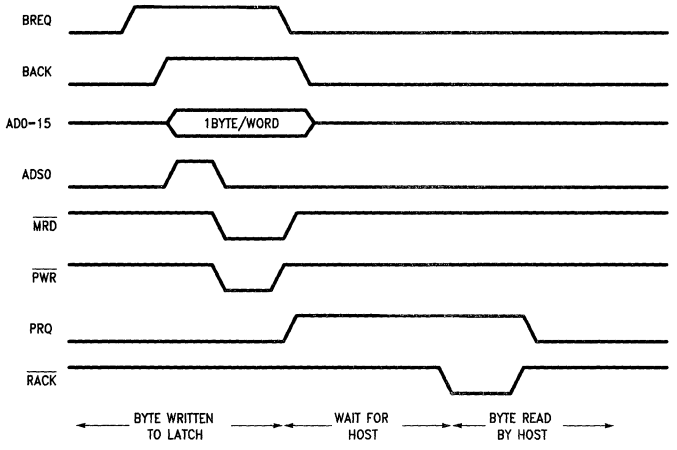
13.0 Bus Arbitration and Timing (Continued)

REMOTE READ TIMING

1) The DMA reads byte/word from local buffer memory and writes byte/word into latch, increments the DMA address and decrements the byte count (RBCR0,1).
2) A Request Line (PRQ) is asserted to inform the system that a byte is available.
3) The system reads the port, the read strobe ($\overline{RACK}$) is used as an acknowledge by the Remote DMA and it goes back to step 1.

   Steps 1–3 are repeated until the remote DMA is complete.

Note that in order for the Remote DMA to transfer a byte from memory to the latch, it must arbitrate access to the local bus via a BREQ, BACK handshake. After each byte or word is transferred to the latch, BREQ is dropped. If a Local DMA is in progress, the Remote DMA is held off until the local DMA is complete.



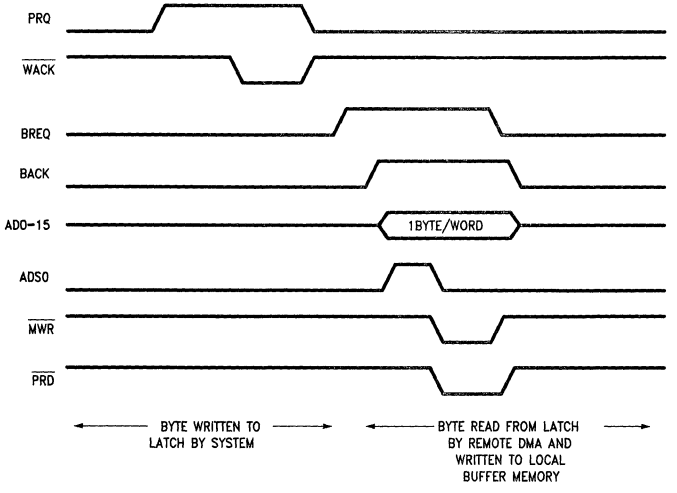
TL/F/8582–26

REMOTE WRITE TIMING

A Remote Write operation transfers data from the I/O port to the local buffer RAM. The NIC initiates a transfer by requesting a byte/word via the PRQ. The system transfers a byte/word to the latch via $\overline{IOW}$, this write strobe is detected by the NIC and PRQ is removed. By removing the PRQ, the Remote DMA holds off further transfers into the latch until the current byte/word has been transferred from the latch, PRQ is reasserted and the next transfer can begin.

1) NIC asserts PRQ. System writes byte/word into latch. NIC removes PRQ.
2) Remote DMA reads contents of port and writes byte/word to local buffer memory, increments address and decrements byte count (RBCR0,1).
3) Go back to step 1.

   Steps 1–3 are repeated until the remote DMA is complete.



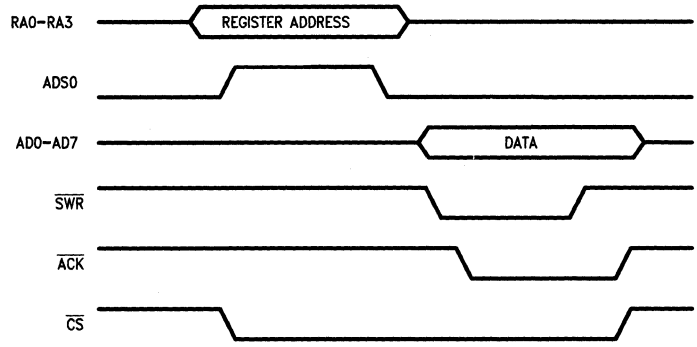
TL/F/8582–27

13.0 Bus Arbitration and Timing (Continued)

SLAVE MODE TIMING

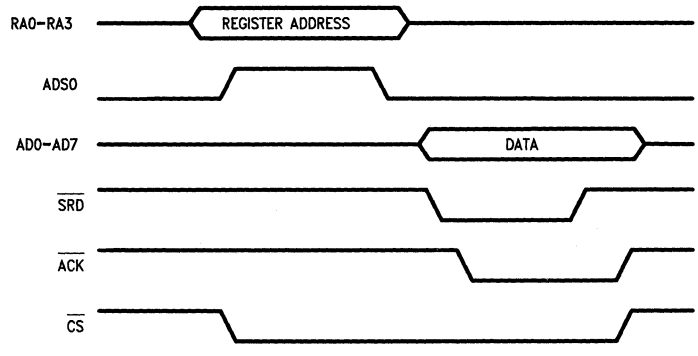
When $\overline{CS}$ is low, the DP8390 NIC becomes a bus slave. The CPU can then read or write any internal registers. All register access is byte wide. The timing for register access is shown below. The host CPU accesses internal registers with four address lines, RA0–RA3, $\overline{SRD}$ and $\overline{SWR}$ strobes. ADS0 is used to latch the address when interfacing to a multiplexed, address data bus. Since the DP8390 may be a local bus master when the host CPU attempts to read or write to the controller, an $\overline{ACK}$ line is used to hold off the CPU until the DP8390 leaves master mode. Some number of BUSCK cycles are also required to allow the NIC to synchronize to the read or write cycle.

Write to Register



TL/F/8582–28

Read from Register



TL/F/8582–29

14.0 Preliminary Electrical Characteristics

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage (V_{CC})	−0.5V to 7.0V
DC Input Voltage (V_{IN})	−0.5V to V_{CC} + 0.5V
DC Output Voltage (V_{OUT})	−0.5V to V_{CC} + 0.5V
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (PD)	500 mW
Lead Temp. (TL) (Soldering, 10 sec.)	260°C

ESD rating is to be determined.

Preliminary DC Specifications T_A = 0°C to 70°C V_{CC} = 5V ±10%, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{OH}	Minimum High Level Output Voltage (Note 1, 4)	I_{OH} = −20 μA I_{OH} = −2.0 mA	V_{CC} − 0.1 3.5			V V
V_{OL}	Minimum Low Level Output Voltage (Note 1, 4)	I_{OL} = 20 μA I_{OL} = 2.0 mA			0.1 0.4	V V
V_{IH}	Minimum High Level Input Voltage (Note 2, 5)		2.0			V
V_{IH2}	Minimum High Level Input Voltage for RACK, WACK (Note 2, 5)					V
V_{IL}	Minimum Low Level Input Voltage (Note 2, 5)				0.8	V
I_{IN}	Input Current	V_I = V_{CC} or GND	−1.0		+1.0	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND	−10		+10	μA
I_{CC}	Average Supply Current (Note 3)	TXCK = 10 MHz RXCK = 10 MHz BSCK = 20 MHz I_{OUT} = 0 μA V_{IN} = V_{CC} or GND			40	mA

*At this time a characterization of the NIC has not been completed and the above specifications are provided as a guideline.

Note 1: All outputs are CMOS compatible outputs with TTL driving capability except LBK, TXE and TXD.

Note 2: All inputs are TTL compatible inputs.

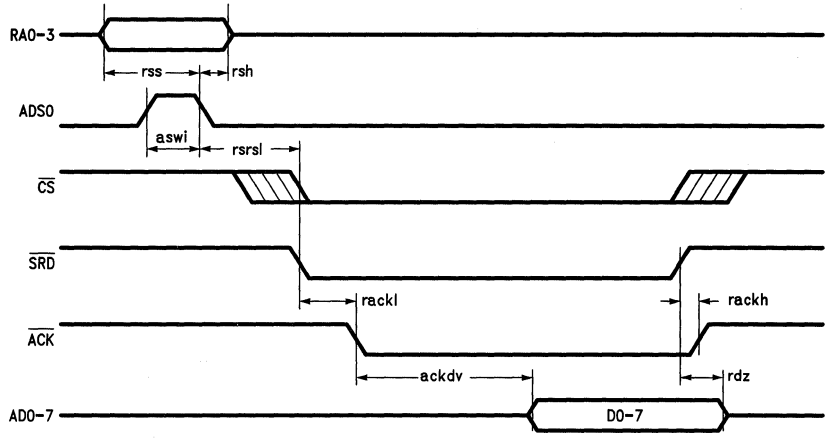
Note 3: This is measured with a 0.1 μF bypass capacitor between V_{CC} and GND.

Note 4: Refer to AC Test Load

Note 5: Limited functional test patterns are performed at these input levels. The majority of functional tests are performed at levels of 0 and 3 volts.

15.0 Switching Characteristics AC Specs DP8390 Note: All Timing is Preliminary

Register Read (Latched Using ADS0)



TL/F/8582–37

Symbol	Parameter	Min*	Max*	Units
rss	Register Select Setup to ADS0 Low			
rsh	Register Select Hold from ADS0 Low			
aswi	Address Strobe Width In			
ackdv	Acknowledge Low to Data Valid			
rdz	Read Strobe to Data TRI-STATE			
rackl	Read Strobe to $\overline{ACK}$ Low (Notes 1, 3)			
rackh	Read Strobe to $\overline{ACK}$ High			
rsrsl	Register Select to Slave Read Low, Latched RS0–3 (Note 2) BAC as set			

Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SRD}$ are low and the NIC has synchronized to the register access. The NIC will insert an internal number of Bus Clock cycles until it is synchronized. In Dual Bus systems additional cycles will be used for a local or remote DMA to complete. Wait states must be issued to the CPU until $\overline{ACK}$ is asserted low.

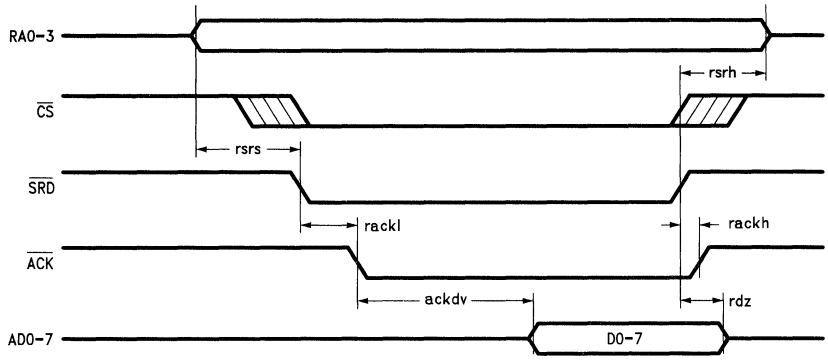
Note 2: $\overline{CS}$ should be asserted concurrently with $\overline{SRD}$ or before $\overline{SRD}$ is asserted. $\overline{CS}$ can be deasserted concurrently with $\overline{SRD}$ or after $\overline{SRD}$ is deasserted.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

Register Read (Non Latched, ADS0 = 1)



TL/F/8582-38

Symbol	Parameter	Min*	Max*	Units
rsrs	Register Select to Read Setup (Notes 1, 3)			
rsrh	Register Select Hold from Read			
ackdv	$\overline{ACK}$ Low to Valid Data			
rdz	Read Strobe to Data TRI-STATE (Note 2)			
rackl	Read Strobe to $\overline{ACK}$ Low			
rackh	Read Strobe to $\overline{ACK}$ High			

Note 1: rsrs includes flow through time of latch.

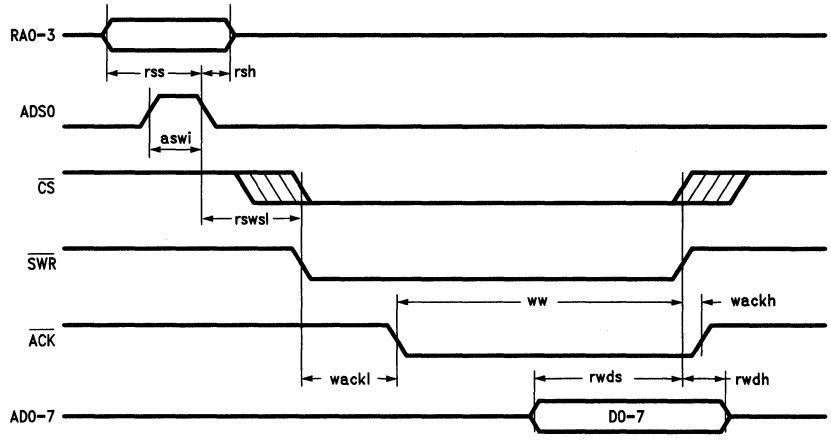
Note 2: These limits inlcude the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

Note 3: RA0–3, $\overline{CS}$ and $\overline{SRD}$ can be asserted concurrently since address decode begins when $\overline{ACK}$ is asserted.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

Register Write (Latched Using ADS0)



TL/F/8582–39

Symbol	Parameter	Min*	Max*	Units
rss	Register Select Setup to ADS0 Low			
rsh	Register Select Hold from ADS0 Low			
aswi	Address Strobe Width In			
rwds	Register Write Data Setup			
rwdh	Register Write Data Hold			
ww	Write Strobe Width from $\overline{ACK}$			
wackh	Write Strobe High to $\overline{ACK}$ High			
wackl	Write Low to $\overline{ACK}$ Low (Note 1)			
rswsl	Register Select to Write Strobe Low			

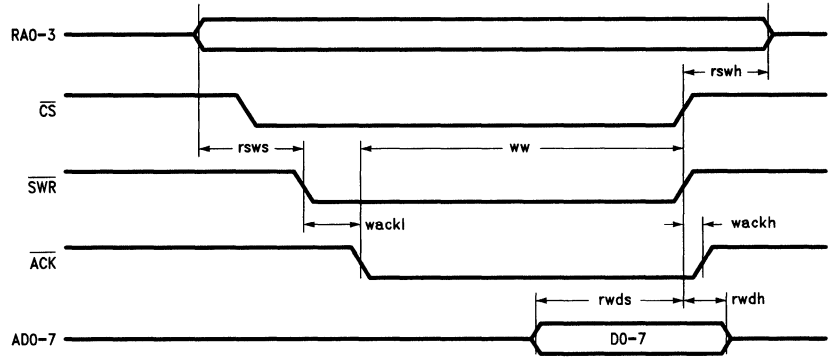
Note 1: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the NIC has synchronized to the register access. In Dual Bus Systems additional cycles will be used for a local DMA or Remote DMA to complete.

Note 2: $\overline{CS}$ should be asserted concurrently with $\overline{SWR}$ or before $\overline{SWR}$ is asserted. $\overline{CS}$ can be deasserted concurrently with $\overline{SWR}$ or after $\overline{SWR}$ is deasserted.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

Register Write (Non Latched, ADS0 = 1)



TL/F/8582-40

Symbol	Parameter	Min*	Max*	Units
rsws	Register Select to Write Setup (Note 1)			
rswh	Register Select Hold from Write			
rwds	Register Write Data Setup			
rwdh	Register Write Data Hold			
wackl	Write Low to $\overline{ACK}$ Low (Note 2)			
wackh	Write High to $\overline{ACK}$ High			
ww	Write Width from $\overline{ACK}$			

Note 1: Assumes ADS0 is high when RA0–3 changing.

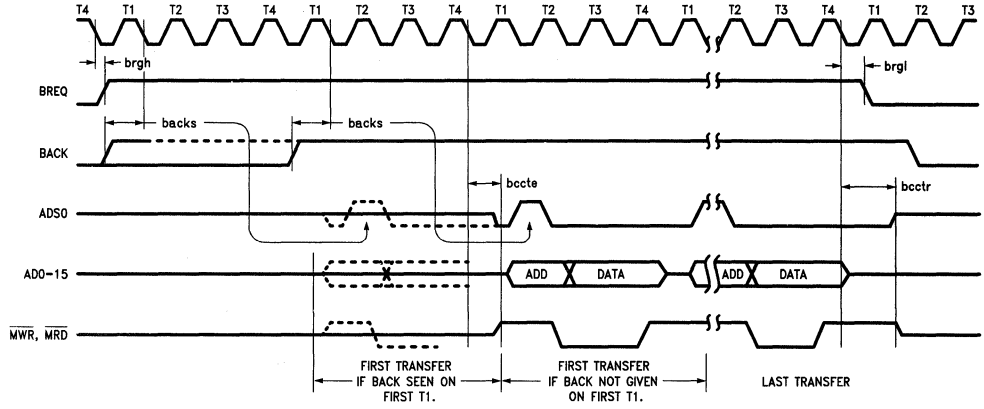
Note 2: $\overline{ACK}$ is not generated until $\overline{CS}$ and $\overline{SWR}$ are low and the NIC has synchronized to the register access. In Dual Bus systems additional cycles will be used for a local DMA or remote DMA to complete.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

5

15.0 Switching Characteristics (Continued)

DMA Control, Bus Arbitration



TL/F/8582–41

Symbol	Parameter	Min*	Max*	Units
brqh	Bus Clock to Bus Request High			
brql	Bus Request Low from Bus Clock			
backs	Acknowledge Setup to Bus Clock (Note 1)			
bccte	Bus Clock to Control Enable			
bcctr	Bus Clock to Control Release (Notes 2, 3)			

Note 1: BACK must be setup before T1 after BREQ is asserted. Missed setup will slip the beginning of the DMA by four bus clocks. The Bus Latency will influence the allowable FIFO threshold and transfer mode (empty fill vs exact burst XFER).

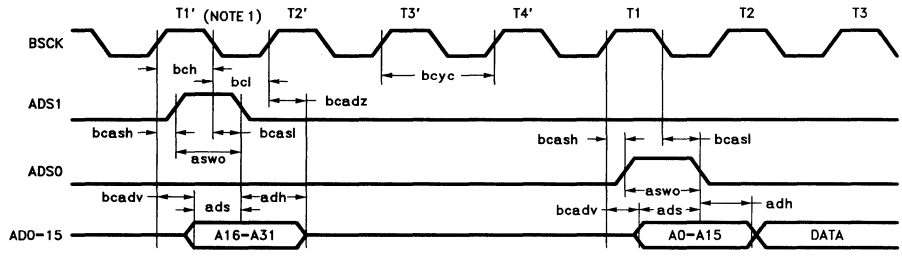
Note 2: During remote DMA transfers only, a single bus transfer is performed. During local DMA operations burst mode transfers are performed.

Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns enabling other devices to drive these lines with no contention.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

DMA Address Generation



TL/F/8582-42

Symbol	Parameter	Min*	Max*	Units
bcyc	Bus Clock Cycle Time (Note 2)			
bch	Bus Clock High Time			
bcl	Bus Clock Low Time			
bcash	Bus Clock to Address Strobe High			
bcasl	Bus Clock to Address Strobe Low			
aswo	Address Strobe Width Out			
bcadv	Bus Clock to Address Valid			
bcadz	Bus Clock to Address TRI-STATE (Note 3)			
ads	Address Setup to ADS0/1 Low			
adh	Address Hold from ADS0/1 Low			

Note 1: Cycles T1', T2', T3', T4' are only issued for the first transfer in a burst when 32-bit mode has been selected.

Note 2: The rate of bus clock must be high enough to support transfers to/from the FIFO at a rate greater than the serial network transfers from/to the FIFO.

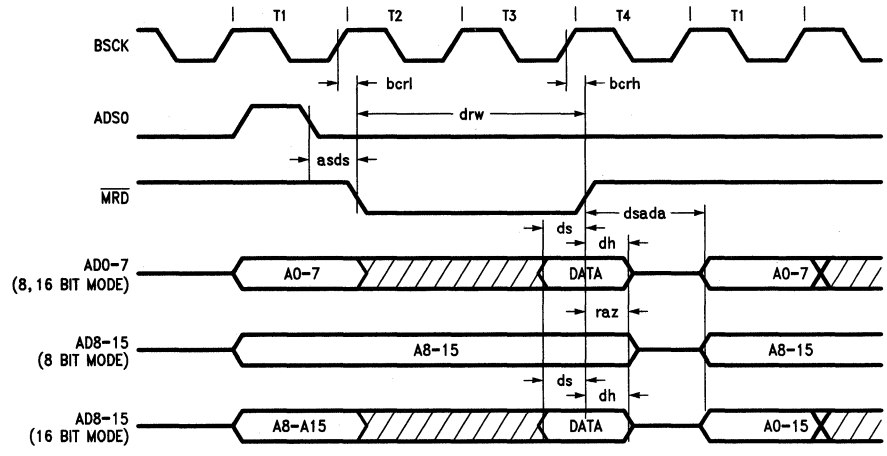
Note 3: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

5

15.0 Switching Characteristics (Continued)

DMA Memory Read



TL/F/8582–43

Symbol	Parameter	Min*	Max*	Units
bcrl	Bus Clock to Read Strobe Low			
bcrh	Bus Clock to Read Strobe High			
ds	Data Setup to Read Strobe High			
dh	Data Hold from Read Strobe High			
drw	DMA Read Strobe Width Out			
raz	Memory Read High to Address TRI-STATE (Notes 1, 2)			
asds	Address Strobe to Data Strobe			
dsada	Data Strobe to Address Valid			

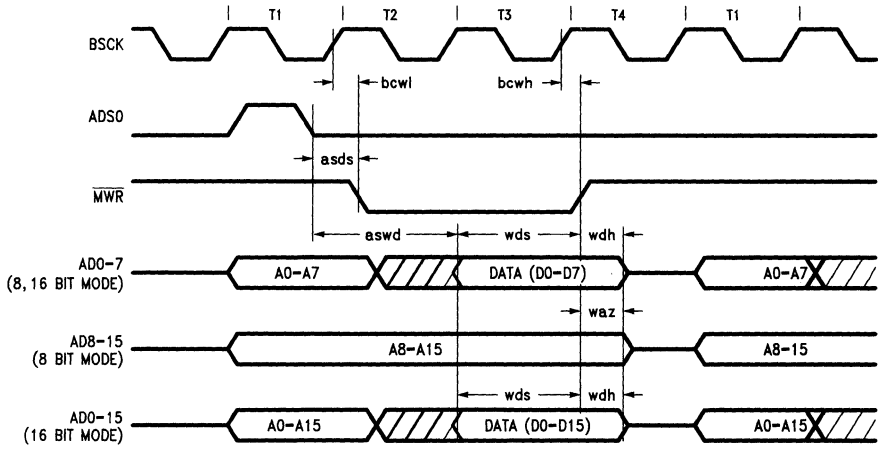
Note 1: During a burst A8–A15 are not TRI-STATE if byte wide transfers are selected. On the last transfer A8–A15 are TRI-STATE as shown above.

Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

DMA Memory Write



TL/F/8582–44

Symbol	Parameter	Min*	Max*	Units
bcwl	Bus Clock to Write Strobe Low			
bcwh	Bus Clock to Write Strobe High			
wds	Data Setup to $\overline{WR}$ High			
wdh	Data Hold from $\overline{WR}$ Low			
waz	Write Strobe to Address TRI-STATE (Notes 1, 2)			
asds	Address Strobe to Data Strobe			
aswd	Address Strobe to Write Data Valid			

Note 1: When using byte mode transfers A8–A15 are only TRI-STATE on the last transfer, waz timing is only valid for last transfer in a burst.

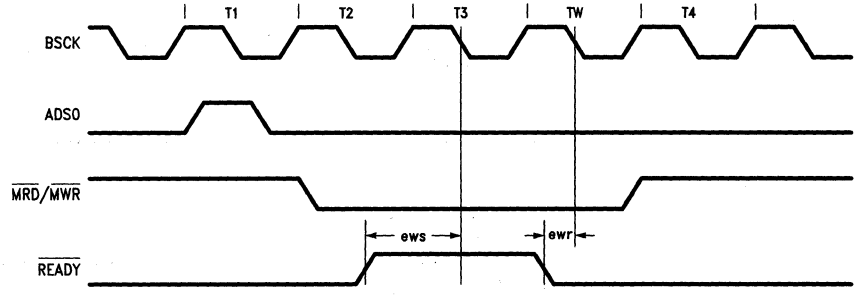
Note 2: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive these lines with no contention.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

5

15.0 Switching Characteristics (Continued)

Wait State Insertion



TL/F/8582–45

Symbol	Parameter	Min*	Max*	Units
ews	External Wait Setup to T3 ↓ Clock (Note 1)			
ewr	External Wait Release Time (Note 1)			

Note 1: The addition of wait states affects the count of deserialized bytes and is limited to a number of bus clock cycles depending on the bus clock and network rates. The allowable wait states are found in the table below. (Assumes 10 Mbit/sec data rate.)

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

BUSCK (MHz)	# of Wait States	
	Byte Transfer	Word Transfer
8	0	1
10	0	1
12	1	2
14	1	2
16	1	3
18	2	3
20	2	4

Table assumes 10 MHz network clock.

The number of allowable wait states in byte mode can be calculated using:

$$\#W_{(\text{byte mode})} = \left(\frac{8\ tnw}{4.5\ tbsck} - 1\right)$$

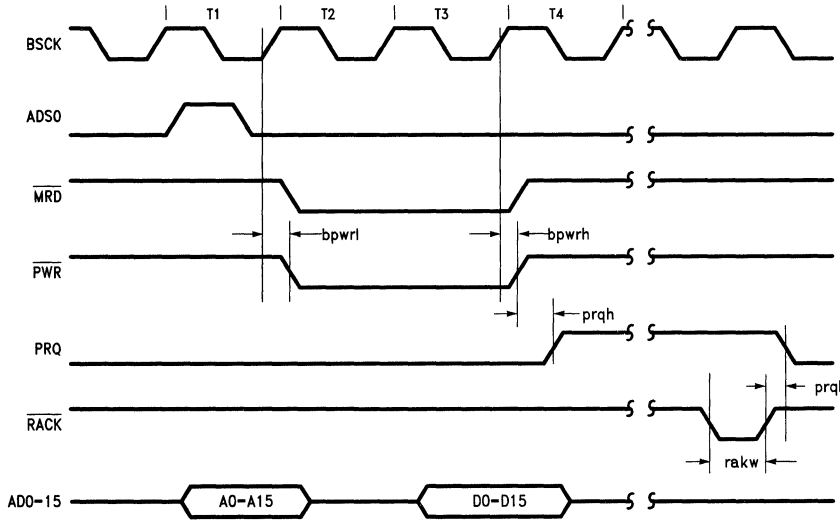
#W = Number of Wait States
tnw = Network Clock Period
tbsck = BSCK Period

The number of allowable wait states in word mode can be calculated using:

$$\#W_{(\text{word mode})} = \left(\frac{5\ tnw}{2\ tbsck} - 1\right)$$

15.0 Switching Characteristics (Continued)

Remote DMA (Read, Send Command)



TL/F/8582–46

Symbol	Parameter	Min*	Max*	Units
bpwrl	Bus Clock to Port Write Low			
bpwrh	Bus Clock to Port Write High			
prqh	Port Write High to Port Request High (Note 1)			
prql	Port Request Low from Read Acknowledge High			
rakw	Remote Acknowledge Read Strobe Pulse Width			

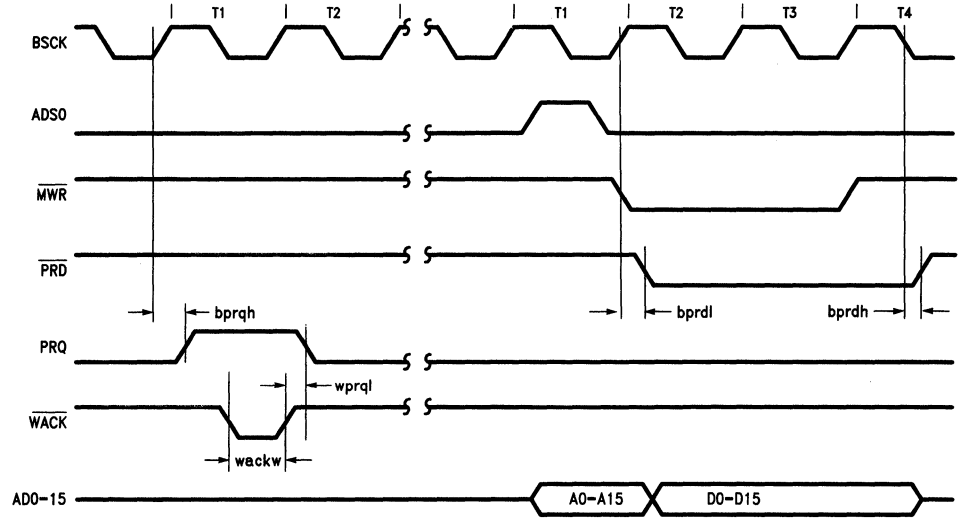
Note 1: Start of next transfer is dependent on where $\overline{RACK}$ is generated relative to BSCK and whether a local DMA is pending.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

5

15.0 Switching Characteristics (Continued)

Remote DMA (Write Cycle)



TL/F/8582-47

Symbol	Parameter	Min*	Max*	Units
bprqh	Bus Clock to Port Request High (Note 1)			
wprql	$\overline{WACK}$ to Port Request Low			
wackw	$\overline{WACK}$ Pulse Width			
bprdl	Bus Clock to Port Read Low (Note 2)			
bprdh	Bus Clock to Port Read High			

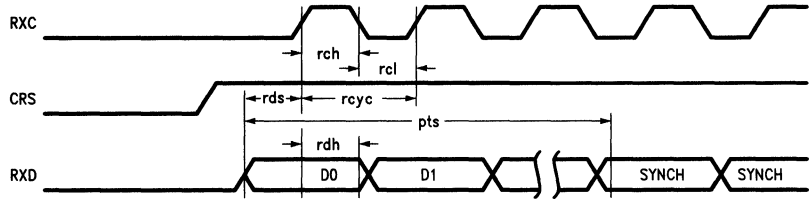
Note 1: The first port request is issued in response to the remote write command. It is subsequently issued on T1 clock cycles following completion of remote DMA cycles.

Note 2: The start of the remote DMA write following $\overline{WACK}$ is dependent on where $\overline{WACK}$ is issued relative to BUSCK and whether a local DMA is pending.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

Serial Timing—Receive (Beginning of Frame)



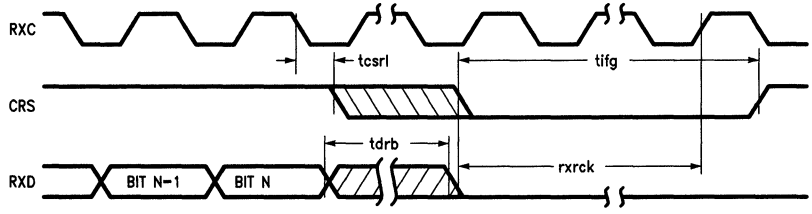
TL/F/8582–48

Symbol	Parameter	Min*	Max*	Units
rch	Receive Clock High Time			
rcl	Receive Clock Low Time			
rcyc	Receive Clock Cycle Time			
rds	Receive Data Setup Time to Receive Clock High (Note 1)			
rdh	Receive Data Hold Time from Receive Clock High			
pts	First Preamble Bit to Synch (Note 2)			

Note 1: All bits entering NIC must be properly decoded, if the PLL is still locking, the clock to the NIC should be disabled or CRS delayed. Any two sequential 1 data bits will be interpreted as Synch.

Note 2: This is a minimum requirement which allows reception of a packet.

Serial Timing—Receive (End of Frame)



TL/F/8582–49

Symbol	Parameter	Min*	Max*	Units
rxrck	Minimum Number of Receive Clocks after CRS Low (Note 1)			
tdrb	Maximum of Allowed Dribble Bits/Clocks (Note 2)			
tifg	Receive Recovery Time (Note 4)			
tcrsl	Receive Clock to Carrier Sense Low (Notes 3, 5)			

Note 1: The NIC requires a minimum number of receive clocks following the deassertion of carrier sense (CRS). These additional clocks are provided by the DP8391 SNI. If other decoder/PLLs are being used additional clocks should be provided. Short clocks or glitches are not allowed.

Note 2: Up to 5 bits of dribble bits can be tolerated without resulting in a receive error.

Note 3: Guarantees to only load bit N, additional bits up to tdrb can be tolerated.

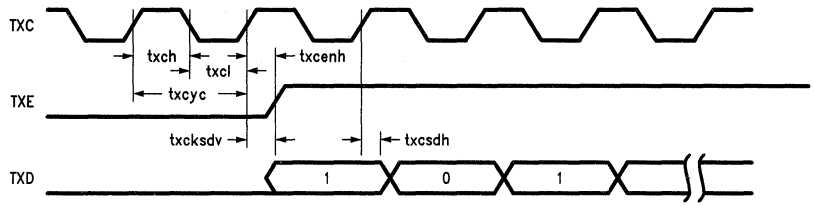
Note 4: This is the time required for the receive state machine to complete end of receive processing. This includes a minimum of 8 RXCK cycles for CRS to be low. This is not a measured parameter but is a design requirement.

Note 5: CRS must remain deasserted for a minimum of 2 RXC cycles to be recognized as end of carrier.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

Serial Timing—Transmit (Beginning of Frame)

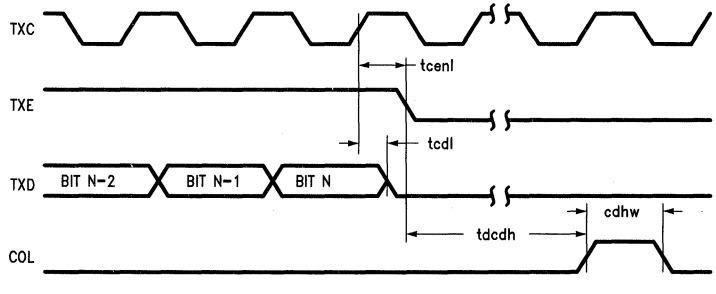


TL/F/8582–50

Symbol	Parameter	Min*	Max*	Units
txch	Transmit Clock High Time			
txcl	Transmit Clock Low Time			
txcyc	Transmit Clock Cycle Time			
txcenh	Transmit Clock to Transmit Enable High (Note 1)			
txcksdv	Transmit Clock to Serial Data Valid			
txcksdh	Serial Data Hold Time from Transmit Clock High			

Note 1: The NIC issues TXEN coincident with the first bit of preamble. The first bit of preamble is always a 1.

Serial Timing—Transmit (End of Frame, CD Heartbeat)



TL/F/8582–51

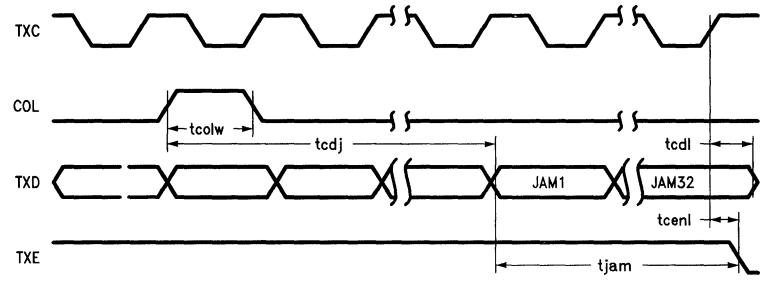
Symbol	Parameter	Min*	Max*	Units
tcdl	Transmit Clock to Data Low			
tcenl	Transmit Clock to TXEN Low			
tdcdh	TXEN Low to Start of Collision Detect Heartbeat (Note 1)			
cdhw	Collision Detect Width			

Note 1: If COL is not seen during the first 64 TX clock cycles following deassertion of TXEN, the CDH bit in the TSR is set.

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

15.0 Switching Characteristics (Continued)

Serial Timing—Transmit (Collision)



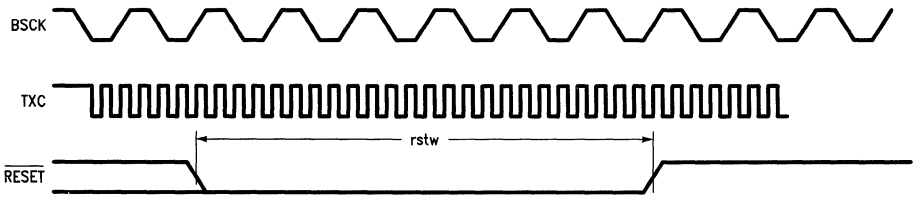
TL/F/8582–52

Symbol	Parameter	Min*	Max*	Units
tcolw	Collision Detect Width			
tcdj	Delay from Collision to First Bit of Jam (Note 1)			
tjam	Jam Period (Note 2)			

Note 1: The NIC must synchronize to collision detect. If the NIC is in the middle of serializing a byte of data the remainder of the byte will be serialized. Thus the jam pattern will start anywhere from 1 to 8 TXC cycles after COL is asserted.

Note 2: The NIC always issues 32 bits of jam. The jam is all 1's data.

Reset Timing



TL/F/8582–53

Symbol	Parameter	Min*	Max*	Units
rstw	Reset Pulse Width (Note 1)	8		BSCK Cycles or TXC Cycles (Note 2)

Note 1: The RESET pulse requires that BSCK and TXC be stable. On power up, $\overline{\text{RESET}}$ should not be raised until BSCK and TXC have become stable. Several registers are affected by $\overline{\text{RESET}}$. Consult the register descriptions for details.

Note 2: The slower of BSCK or TXC clocks will determine the minimum time for the $\overline{\text{RESET}}$ signal to be low.

If BSCK < TXC then $\overline{\text{RESET}}$ = 8 × BSCK

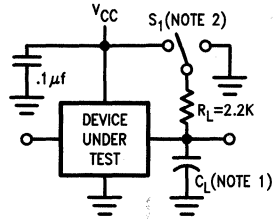
If TXC < BSCK then $\overline{\text{RESET}}$ = 8 × TXC

***Data not available at press time for min and max values. Please call your local National Sales engineer for more complete information.**

5

AC Timing Test Conditions

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	5 ns
Input and Output Reference Levels	1.3V
TRI-STATE Reference Levels	Float (ΔV) ±0.5V
Output Load (See Figure below)	



TL/F/8582–54

Note 1: C_L = 50 pF, includes scope and jig capacitance.

Note 2: S1 = Open for timing tests for push pull outputs.

S1 = V_{CC} for V_{OL} test.

S1 = GND for V_{OH} test.

S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.

S1 = GND for High Impedance to active high and active high to High Impedance measurements.

Capacitance T_A = 25°C, f = 1 MHz

Parameter	Description	Typ	Max	Unit
C_{IN}	Input Capacitance	7	15	pF
C_{OUT}	Output Capacitance	7	15	pF

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

DERATING FACTOR

Output timings are measured with a purely capacitave load for 50 pF. The following correction factor can be used for other loads:

$C_L \geq$ 50 pf: +0.3 ns/pF (for all outputs except TXE, TXD, and LBK)

National Semiconductor Corporation

DP8391/NS32491 Serial Network Interface

General Description

The DP8391 Serial Network Interface (SNI) provides the Manchester data encoding and decoding functions for IEEE 802.3 Ethernet/Cheapernet type local area networks. The SNI interfaces the DP8390 Network Interface Controller (NIC) to the Ethernet transceiver cable. When transmitting, the SNI converts non-return-to-zero (NRZ) data from the controller and clock pulses into Manchester encoding and sends the converted data differentially to the transceiver. The opposite process occurs on the receive path, where a digital phase-locked loop decodes 10 Mbit/s signals with as much as ±20 ns of jitter.

The DP8391 SNI is a functionally complete Manchester encoder/decoder including ECL like balanced driver and receivers, on board crystal oscillator, collision signal translator, and a diagnostic loopback circuit.

The SNI is part of a three chip set that implements the complete IEEE compatible network node electronics as shown below. The other two chips are the DP8392 Coax Transceiver Interface (CTI) and the DP8390 Network Interface Controller (NIC).

Incorporated into the CTI are the transceiver, collision and jabber functions. The Media Access Protocol and the buffer management tasks are performed by the NIC. There is an isolation requirement on signal and power lines between the CTI and the SNI. This is usually accomplished by using a set of miniature pulse transformers that come in a 16-pin plastic DIP for signal lines. Power isolation, however, is done by using a DC to DC converter.

Features

- Compatible with Ethernet II, IEEE 802.3 10base5 and 10base2 (Cheapernet)
- 10 Mb/s Manchester encoding/decoding with receive clock recovery
- Patented digital phase locked loop (DPLL) decoder requires no precision external components
- Decodes Manchester data with up to ±20 ns of jitter
- Loopback capability for diagnostics
- Externally selectable half or full step modes of operation at transmit output
- Squelch circuits at the receive and collision inputs reject noise
- High voltage protection at transceiver interface (16V)
- TTL/MOS compatible controller interface
- Connects directly to the transceiver (AUI) cable

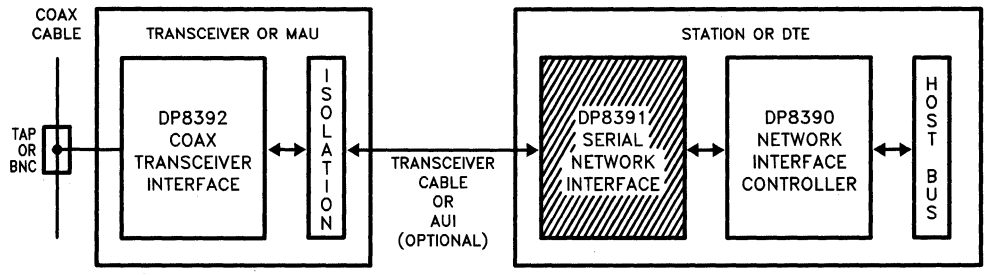
Table of Contents

1.0 System Diagram
2.0 Block Diagram
3.0 Functional Description
3.1 Oscillator
3.2 Encoder
3.3 Decoder
3.4 Collision Translator
3.5 Loopback
4.0 Connection Digram
5.0 Pin Description
6.0 Absolute Maximum Ratings
7.0 Electrical Characteristics
8.0 Switching Characteristics
9.0 Timing and Load Diagrams
10.0 Physical Dimensions

5

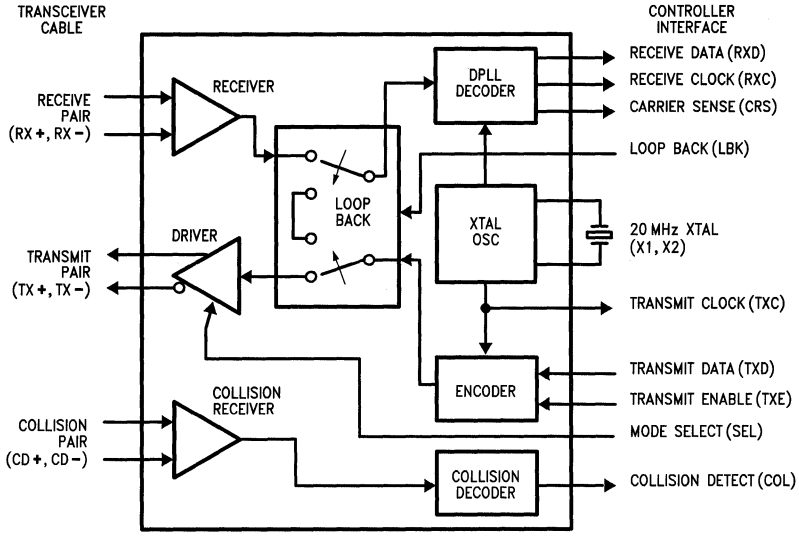
1.0 System Diagram

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set



TL/F/6758–1

2.0 Block Diagram



TL/F/6758–2

FIGURE 1

3.0 Functional Description

The SNI consists of five main logical blocks:

a) the oscillator—generates the 10 MHz transmit clock signal for system timing.

b) the Manchester encoder and differential output driver—accepts NRZ data from the controller, performs Manchester encoding, and transmits it differentially to the transceiver.

c) the Manchester decoder—receives Manchester data from the transceiver, converts it to NRZ data and clock pulses, and sends them to the controller.

d) the collision translator—indicates to the controller the presence of a valid 10 MHz signal at its input.

e) the loopback circuitry—when asserted, switches encoded data instead of receive input signals to the digital phase-locked loop.

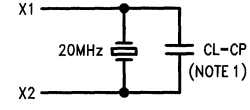
3.1 OSCILLATOR

The oscillator is controlled by a 20 MHz parallel resonant crystal connected between X1 and X2 or by an external clock on X1. The 20 MHz output of the oscillator is divided by 2 to generate the 10 MHz transmit clock for the controller. The oscillator also provides internal clock signals to the encoding and decoding circuits.

Crystal Specification

Resonant frequency	20 MHz
Tolerance	±0.001% at 25°C
Stability	±0.005% 0–70°C
Type	AT-Cut
Circuit	Parallel Resonance

The 20 MHz crystal connection to the SNI requires special care. The IEEE 802.3 standard requires a 0.01% absolute accuracy on the transmitted signal frequency. Stray capacitance can shift the crystal's frequency out of range, causing the transmitted frequency to exceed its 0.01% tolerance. The frequency marked on the crystal is usually measured with a fixed shunt capacitance (C_L) that is specified in the crystal's data sheet. This capacitance for 20 MHz crystals is typically 20 pF. The capacitance between the X1 and X2 pins of the SNI, of the PC board traces and the plated through holes plus any stray capacitance such as the socket capacitance, if one is used, should be estimated or measured. Once the total sum of these capacitances is determined, the value of additional external shunt capacitance required can be calculated. This capacitor can be a fixed 5% tolerance component. The frequency accuracy should be measured during the design phase at the transmit clock pin (TXC) for a given pc layout. *Figure 2* shows the crystal connection.



TL/F/6758–3

CL = Load capacitance specified by the crystal's manufacturer

CP = Total parasitic capacitance including:
a) SNI input capacitance between X1 and X2 (typically 5 pF)
b) PC board traces, plated through holes, socket capacitances

Note 1: When using a Viking (San Jose) VXB49N5 crystal, the external capacitor is not required, as the C_L of the crystal matches the input capacitance of the DP8391.

FIGURE 2. Crystal Connection

3.2 MANCHESTER ENCODER AND DIFFERENTIAL DRIVER

The encoder combines clock and data information for the transceiver. Data encoding and transmission begins with the transmit enable input (TXE) going high. As long as TXE re-

3.0 Functional Description (Continued)

mains high, transmit data (TXD) is encoded out to the transmit-driver pair (TX±). The transmit enable and transmit data inputs must meet the setup and hold time requirements with respect to the rising edge of transmit clock. Transmission ends with the transmit enable input going low. The last transition is always positive at the transmit output pair. It will occur at the center of the bit cell if the last bit is one, or at the boundary of the bit cell if the last bit is zero.

The differential line driver provides ECL like signals to the transceiver with typically 5 ns rise and fall times. It can drive up to 50 meters of twisted pair AUI Ethernet transceiver cable. These outputs are source followers which need external 270Ω pulldown resistors to ground. Two different modes, full-step or half-step, can be selected with SEL input. With SEL low, transmit + is positive with respect to transmit − in the idle state. With SEL high, transmit + and transmit − are equal in the idle state, providing zero differential voltage to operate with transformer coupled loads. *Figures 4, 5* and *6* illustrate the transmit timing.

3.3 MANCHESTER DECODER

The decoder consists of a differential input circuitry and a digital phase-locked loop to separate Manchester encoded data stream into clock signals and NRZ data. The differential input should be externally terminated if the standard 78Ω transceiver drop cable is used. Two 39Ω resistors connected in series and one optional common mode bypass capacitor would accomplish this. A squelch circuit at the input rejects signals with pulse widths less than 8 ns (negative going), or with levels less than −175 mV. Signals more negative than −300 mV and with a duration greater than 30 ns are always decoded. This prevents noise at the input from falsely triggering the decoder in the absence of a valid signal. Once the input exceeds the squelch requirements, carrier sense (CRS) is asserted. Receive data (RXD) and receive clock (RXC) become available typically within 6 bit times. At this point the digital phase-locked loop has locked to the incoming signal. The DP8391 decodes a data frame with up to ±20 ns of jitter correctly.

The decoder detects the end of a frame when the normal mid-bit transition on the differential input ceases. Within one and a half bit times after the last bit, carrier sense is de-asserted. Receive clock stays active for five more bit times before it goes low and remains low until the next frame. *Figures 7, 8* and *9* illustrate the receive timing.

3.4 COLLISION TRANSLATOR

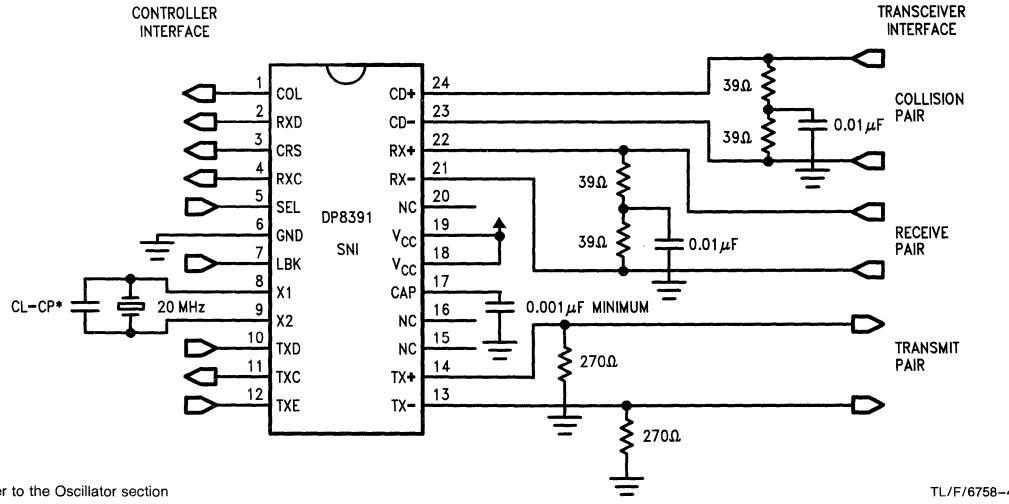
The Ethernet transceiver detects collisions on the coax cable and generates a 10 MHz signal on the transceiver cable. The SNI's collision translator asserts the collision detect output (COL) to the DP8390 controller when a 10 MHz signal is present at the collision inputs. The controller uses this signal to back off transmission and recycle itself. The collision detect output is de-asserted within 350 ns after the 10 MHz input signal disappears.

The collision differential inputs (+ and −) should be terminated in exactly the same way as the receive inputs. The collision input also has a squelch circuit that rejects signals with pulse widths less than 8 ns (negative going), or with levels less than −175 mV. *Figure 10* illustrates the collision timing.

3.5 LOOPBACK FUNCTIONS

Logic high at loopback input (LBK) causes the SNI to route serial data from the transmit data input, through its encoder, returning it through the phase-locked-loop decoder to receive data output. In loopback mode, the transmit driver is in idle state and the receive input circuitry is disabled.

4.0 Connection Diagram



*Refer to the Oscillator section

TL/F/6758–4

Top View

FIGURE 3

Order Number DP8391N
See NS Package Number N24C

5.0 Pin Descriptions

Pin No.	Name	I/O	Description
1	COL	O	**Collision Detect Output.** A TTL/MOS level active high output. A 10 MHz (+25%−15%) signal at the collision input will produce a logic high at COL output. When no signal is present at the collision input, COL output will go low.
2	RXD	O	**Receive Data Output**. A TTL/MOS level signal. This is the NRZ data output from the digital phase-locked loop. This signal should be sampled by the controller at the rising edge of receive clock.
3	CRS	O	**Carrier Sense**. A TTL/MOS level active high signal. It is asserted when valid data from the transceiver is present at the receive input. It is de-asserted one and a half bit times after the last bit at receive input.
4	RXC	O	**Receive Clock**. A TTL/MOS level recovered clock. When the phase-locked loop locks to a valid incoming signal a 10 MHz clock signal is activated on this output. This output remains low during idle (5 bit times after activity ceases at receive input).
5	SEL	I	**Mode Select.** A TTL level input. When high, transmit + and transmit − outputs are at the same voltage in idle state providing a "zero" differential. When low, transmit + is positive with respect to transmit − in idle state.
6	GND		**Negative Supply Pin.**
7	LBK	I	**Loopback**. A TTL level active high on this input enables the loopback mode.
8	X1	I	**Crystal or External Frequency Source Input (TTL).**
9	X2	O	**Crystal Feedback Output**. This output is used in the crystal connection only. It must be left open when driving X1 with an external frequency source.
10	TXD	I	**Transmit Data**. A TTL level input. This signal is sampled by the SNI at the rising edge of transmit clock when transmit enable input is high. The SNI combines transmit data and transmit clock signals into a Manchester encoded bit stream and sends it differentially to the transceiver.
11	TXC	O	**Transmit Clock**. A TTL/MOS level 10 MHz clock signal derived from the 20 MHz oscillator. This clock signal is always active.
12	TXE	I	**Transmit Enable**. A TTL level active high data encoder enable input. This signal is also sampled by the SNI at the rising edge of transmit clock.
13 14	TX− TX+	O	**Transmit Output**. Differential line driver which sends the encoded data to the transceiver. These outputs are source followers and require 270Ω pulldown resistors to GND.
15 16	NC		**No Connection.**
17	CAP	O	**Bypass Capacitor**. A ceramic capacitor (greater than 0.001 μF) must be connected from this pin to GND.
18 19	VCC		**Positive Supply Pins**. A 0.1 μF ceramic decoupling capacitor must be connected across VCC and GND as close to the device as possible.
20	NC		**No Connection.**
21 22	RX− RX+	I	**Receive Input**. Differential receive input pair from the transceiver.
23 24	CD− CD+	I	**Collision Input**. Differential collision input pair from the transceiver.

6.0 Absolute Maximum Ratings

Supply Voltage (V_{CC})	6V
Input Voltage (TTL)	0 to 5.5V
Input Voltage (differential)	−5.5 to +16V
Output Voltage (differential)	0 to 16V
Output Current (differential)	−40 mA
Storage Temperature	−65° to 150°C
Lead Temperature (soldering, 10 sec)	300°C
Package Power Rating at 25°C (PC Board Mounted) Derate Linearly at the rate of 23.8 mW/°C	2.95W*

*For actual power dissipation of the device please refer to Section 7.0.

ESD rating is to be determined.

Recommended Operating Conditions

Supply Voltage (V_{CC})	5V ± 5%
Ambient Temperature	0° to 70°C

Note: *Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.*

7.0 Electrical Characteristics V_{CC} = 5V ±5%, T_A = 0° to 70°C (Notes 1 & 2)

Symbol	Parameter	Test Conditions	Min	Max	Units
V_{IH}	Input High Voltage (TTL and X1)		2.0		V
V_{IL}	Input Low Voltage (TTL and X1)			0.8	V
I_{IH}	Input High Current (TTL) Input High Current (RX± CD±)	V_{IN} = V_{CC} V_{IN} = V_{CC}		50 500	μA μA
I_{IL}	Input Low Current (TTL) Input Low Current (RX± CD±)	V_{IN} = 0.5V V_{IN} = 0.5V		−300 −700	μA μA
V_{CL}	Input Clamp Voltage (TTL)	I_{IN} = −12 mA		−1.2	V
V_{OH}	Ouptut High Voltage (TTL/MOS)	I_{OH} = −100 μA	3.5		V
V_{OL}	Output Low Voltage (TTL/MOS)	I_{OL} = 8 mA		0.5	V
I_{OS}	Output Short Circuit Current (TTL/MOS)		−40	−200	mA
V_{OD}	Differential Output Voltage (TX±)	78Ω termination, and 270Ω from each to GND	±500	±1200	mV
V_{OB}	Diff. Output Voltage Imbalance (TX±)	same as above		±40	mV
V_{DS}	Diff. Squelch Threshold (RX± CD±)		−175	−300	mV
V_{CM}	Diff. Input Common Mode Voltage (RX± CD±)		5.25	5.25	V
I_{CC}	Power Supply Current	10Mbit/s		270	mA

8.0 Switching Characteristics V_{CC} = 5V ±5%, T_A = 0° to 70°C (Note 2)

Symbol	Parameter	Figure	Min	Typ	Max	Units
OSCILLATOR SPECIFICATION						
t_{XTH}	X1 to Transmit Clock High	12	8		20	ns
t_{XTL}	X1 to Transmit Clock Low	12	8		20	ns
TRANSMIT SPECIFICATION						
t_{TCd}	Transmit Clock Duty Cycle at 50% (10 MHz)	12	42	50	58	%
t_{TCr}	Transmit Clock Rise Time (20% to 80%)	12			8	ns
t_{TCf}	Transmit Clock Fall Time (80% to 20%)	12			8	ns
t_{TDs}	Transmit Data Setup Time to Transmit Clock Rising Edge	4 & 12	20			ns
t_{TDh}	Transmit Data Hold Time from Transmit Clock Rising Edge	4 & 12	0			ns
t_{TEs}	Transmit Enable Setup Time to Trans. Clock Rising Edge	4 & 12	20			ns
t_{TEh}	Transmit Enable Hold Time from Trans. Clock Rising Edge	5 & 12	0			ns
t_{TOd}	Transmit Output Delay from Transmit Clock Rising Edge	4 & 12			40	ns
t_{TOr}	Transmit Output Rise Time (20% to 80%)	12			7	ns
t_{TOf}	Transmit Output Fall Time (80% to 20%)	12			7	ns
t_{TOj}	Transmit Output Jitter	12		±0.25		ns
t_{TOh}	Transmit Output High Before Idle in Half Step Mode	5 & 12	200			ns
t_{TOi}	Transmit Output Idle Time in Half Step Mode	5 & 12			800	ns

Note 1: All currents into device pins are positive, all currents out of device pins are negative. All voltages are referenced to ground unless otherwise specified.

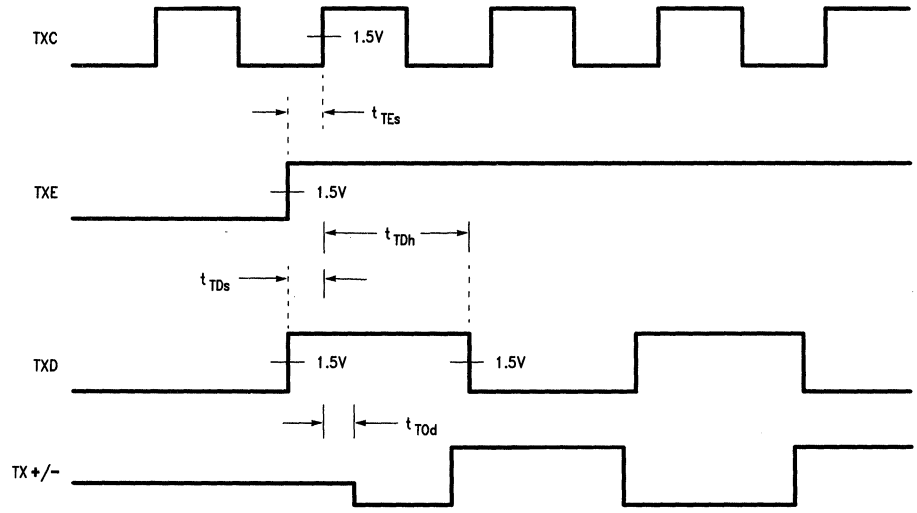
Note 2: All typicals are given for V_{CC} = 5V and T_A = 25°C.

8.0 Switching Characteristics V_{CC} = 5V ±5%, T_A = 0° to 70°C (Note 2) (Continued)

Symbol	Parameter	Figure	Min	Typ	Max	Units
RECEIVE SPECIFICATION						
t_{RCd}	Receive Clock Duty Cycle at 50% (10 MHz)	12	40	50	60	%
t_{RCr}	Receive Clock Rise Time (20% to 80%)	12			8	ns
t_{RCf}	Receive Clock Fall Time (80% to 20%)	12			8	ns
t_{RDr}	Receive Data Rise Time (20% to 80%)	12			8	ns
t_{RDf}	Receive Data Fall Time (80% to 20%)	12			8	ns
t_{RDs}	Receive Data Stable from Receive Clock Rising Edge	7 & 12	±40			ns
t_{CSon}	Carrier Sense Turn On Delay	7 & 12			50	ns
t_{CSoff}	Carrier Sense Turn Off Delay	8, 9 & 12			160	ns
t_{DAT}	Decoder Acquisition Time	7			700	ns
t_{Drej}	Differential Inputs Rejection Pulse Width (Squelch)	7	8		30	ns
t_{Rd}	Receive Throughput Delay	8 & 12			150	ns
COLLISION SPECIFICATION						
t_{COLon}	Collision Turn On Delay	10 & 12			50	ns
t_{COLoff}	Collision Turn Off Delay	10 & 12			350	ns
LOOPBACK SPECIFICATION						
t_{LBs}	Loopback Setup Time	11	20			ns
t_{LBh}	Loopback Hold Time	11	0			ns

Note 2: All typicals are given for V_{CC} = 5V and T_A = 25°C.

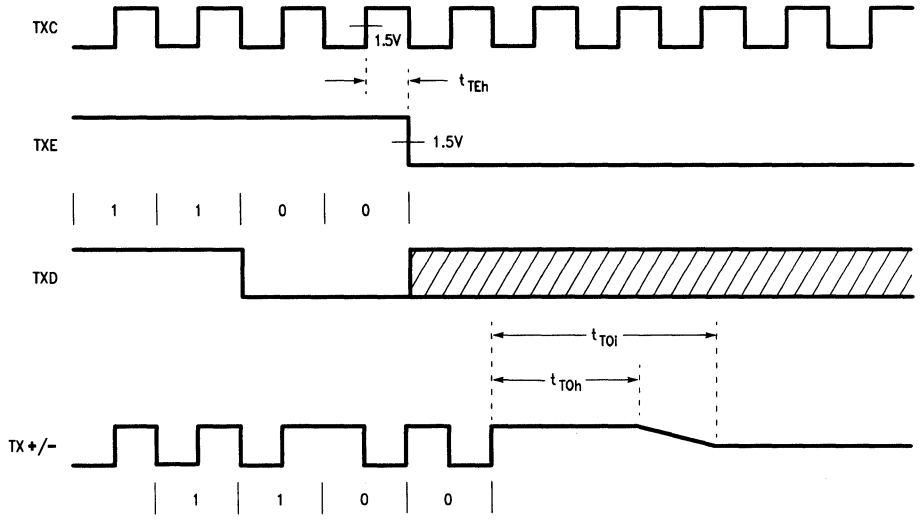
9.0 Timing and Load Diagrams



TL/F/6758–5

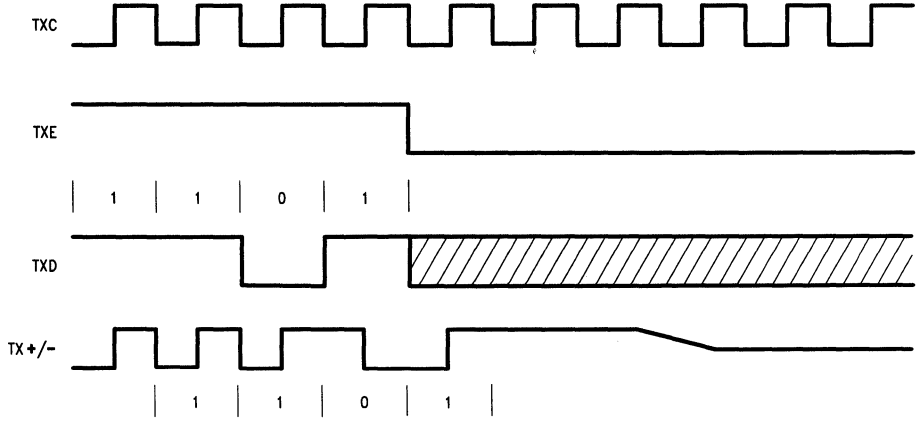
FIGURE 4. Transmit Timing - Start of Transmission

9.0 Timing and Load Diagrams (Continued)



TL/F/6758–6

FIGURE 5. Transmit Timing - End of Transmission (last bit = 0)



TL/F/6758–7

FIGURE 6. Transmit Timing - End of Transmission (last bit = 1)

5

9.0 Timing and Load Diagrams (Continued)

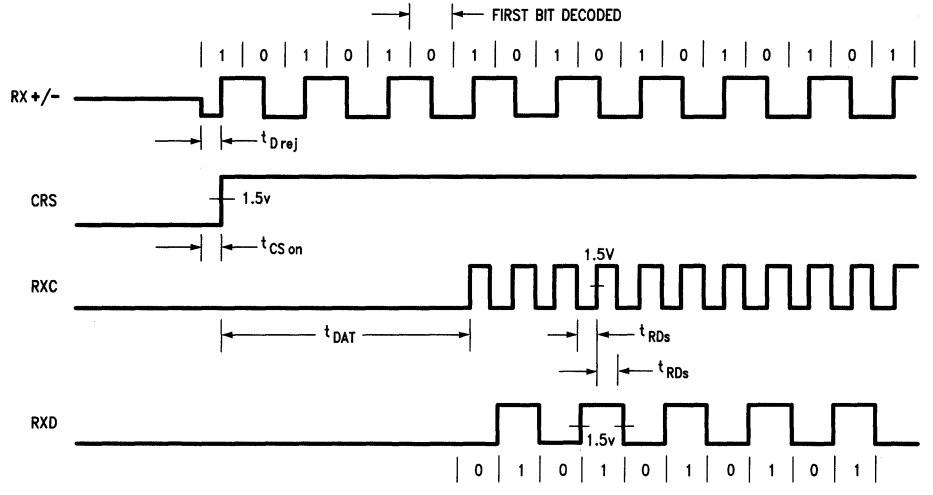


FIGURE 7. Receive Timing - Start of Packet

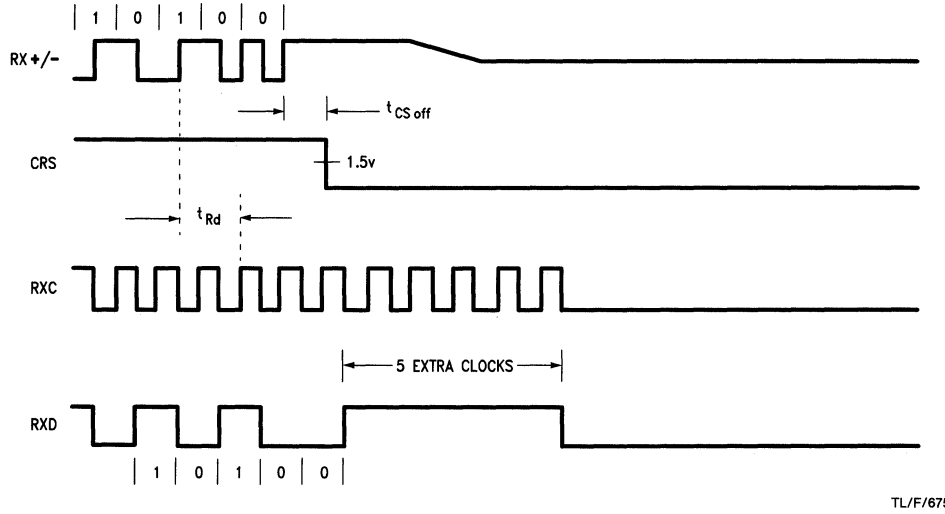
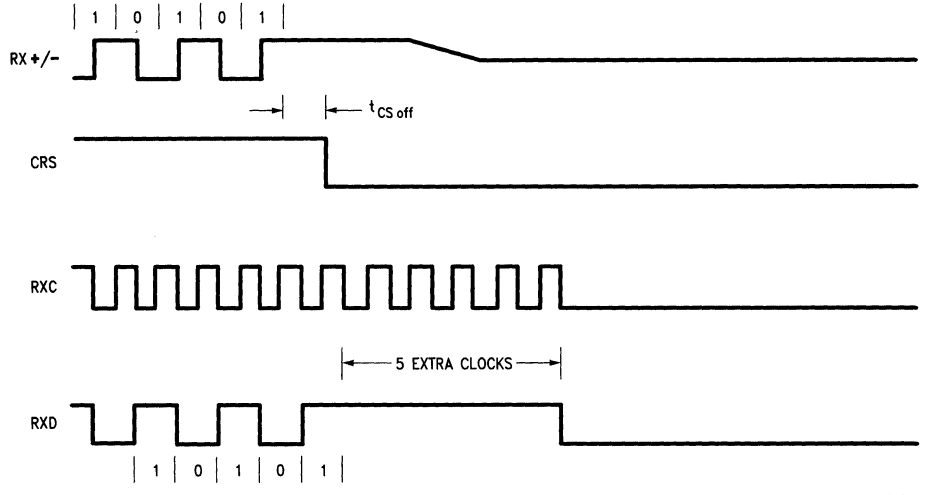


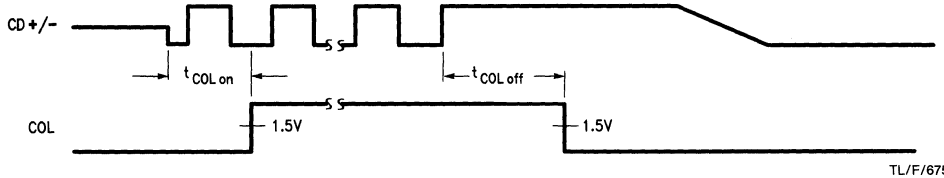
FIGURE 8. Receive Timing - End of Packet (last bit = 0)

9.0 Timing and Load Diagrams (Continued)



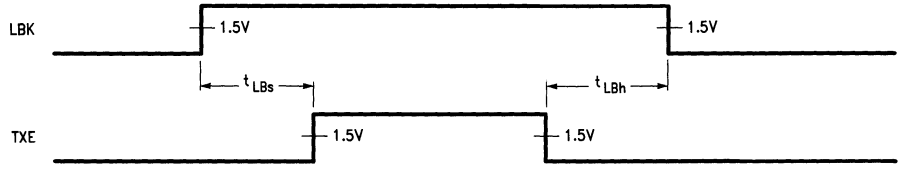
TL/F/6758–10

FIGURE 9. Receive Timing - End of Packet (last bit = 1)



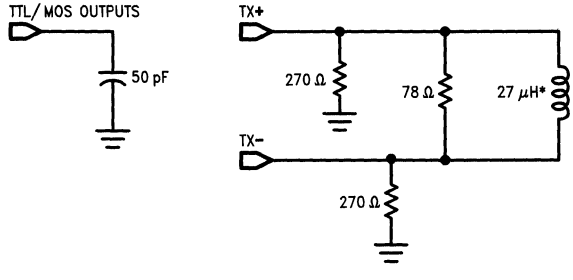
TL/F/6758–11

FIGURE 10. Collision Timing



TL/F/6758–12

FIGURE 11. Loopback Timing



TL/F/6758–13

*27 μH transformer is used for testing purposes, 100 μH transformers (Valor, LT1101, or Pulse Engineering 64103) are recommended for application use.

FIGURE 12. Test Loads

National Semiconductor Corporation

PRELIMINARY

DP8392/NS32492 Coaxial Transceiver Interface

General Description

The DP8392 Coaxial Transceiver Interface (CTI) is used as a coaxial cable line driver/receiver for Ethernet/Cheapernet type local area networks. The CTI is connected between the coaxial cable and the Data Terminal Equipment (DTE). In Ethernet applications the transceiver is usually mounted within a dedicated enclosure and is connected to the DTE via a transceiver cable. In Cheapernet applications, the CTI is typically located within the DTE and connects to the DTE through isolation transformers only. The CTI consists of a Receiver, Transmitter, Collision Detector, and a Jabber Timer. The Transmitter connects directly to a 50 ohm coaxial cable where it is used to drive the coax when transmitting. During transmission, a jabber timer is initiated to disable the CTI transmitter in the event of a longer than legal length data packet. Collision Detection circuitry monitors the signals on the coax to determine the presence of colliding packets and signals the DTE in the event of a collision.

The CTI is part of a three chip set that implements the complete IEEE 802.3 compatible network node electronics as shown below. The other two chips are the DP8391 Serial Network Interface (SNI) and the DP8390 Network Interface Controller (NIC).

The SNI provides the Manchester encoding and decoding functions; whereas the NIC handles the Media Access Protocol and the buffer management tasks. Isolation between the CTI and the SNI is an IEEE 802.3 requirement that can be easily satisfied on signal lines using a set of pulse transformers that come in a standard DIP. However, the power isolation for the CTI is done by DC-to-DC conversion through a power transformer.

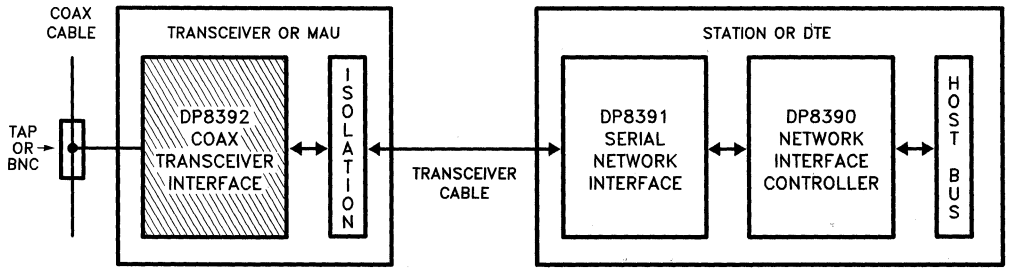
Features

- Compatible with Ethernet II, IEEE 802.3 10Base5 and 10Base2 (Cheapernet)
- Integrates all transceiver electronics except signal & power isolation
- Innovative design minimizes external component count
- Jabber timer function integrated on chip
- Externally selectable CD Heartbeat allows operation with IEEE 802.3 compatible repeaters
- Precision circuitry implements receive mode collision detection
- Squelch circuitry at all inputs rejects noise
- Designed for rigorous reliability requirements of IEEE 802.3
- Standard 16-pin DIP uses a special leadframe that significantly reduces the operating die temperature

Table of Contents

1.0 System Diagram
2.0 Block Diagram
3.0 Functional Description
3.1 Receiver and Squelch
3.2 Transmitter and Squelch
3.3 Collision and Heartbeat
3.4 Jabber Timer
4.0 Connection Diagram
5.0 Pin Description
6.0 Absolute Maximum Ratings
7.0 Electrical Characteristics
8.0 Switching Characteristics
9.0 Timing and Load Diagram
10.0 Physical Dimensions

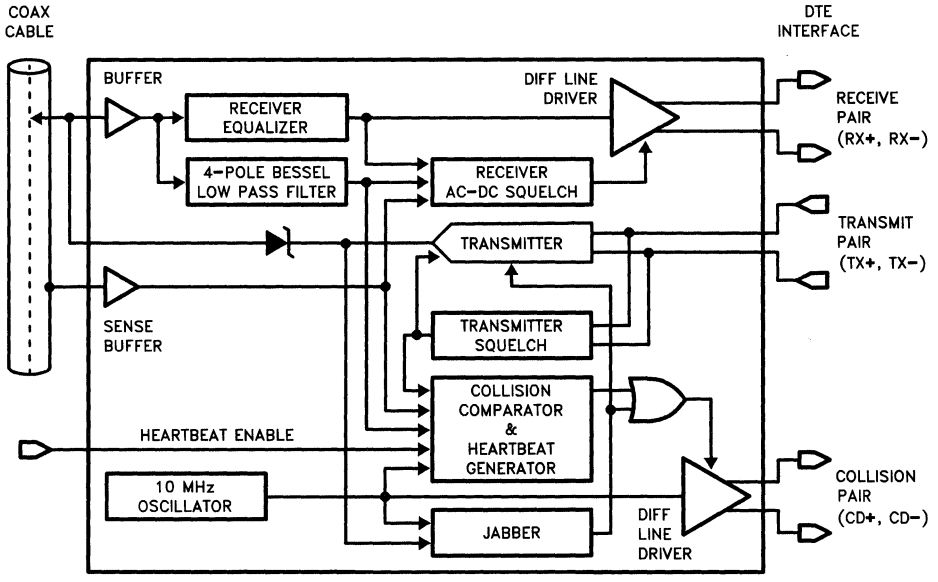
1.0 System Diagram



TL/F/7405–1

IEEE 802.3 Compatible Ethernet/Cheapernet Local Area Network Chip Set

2.0 Block Diagram



TL/F/7405-2

FIGURE 1. DP8392 Block Diagram

3.0 Functional Description

The CTI consists of four main logical blocks:

a) the Receiver - receives data from the coax and sends it to the DTE

b) the Transmitter - accepts data from the DTE and transmits it onto the coax

c) the Collision Detect circuitry - indicates to the DTE any collision on the coax

d) the Jabber Timer - disables the Transmitter in case of longer than legal length packets

3.1 RECEIVER FUNCTIONS

The Receiver includes an input buffer, a cable equalizer, a 4-pole Bessel low pass filter, a squelch circuit, and a differential line driver.

The buffer provides high input impedance and low input capacitance to minimize loading and reflections on the coax.

The equalizer is a high pass filter which compensates for the low pass effect of the cable. The composite result of the maximum length cable and the equalizer is a flatband response at the signal frequencies to minimize jitter.

The 4-pole Bessel low pass filter extracts the average DC level on the coax, which is used by both the Receiver squelch and the collision detection circuits.

The Receiver squelch circuit prevents noise on the coax from falsely triggering the Receiver in the absence of the signal. At the beginning of the packet, the Receiver turns on when the DC level from the low pass filter is lower than the DC squelch threshold. However, at the end of the packet, a quick Receiver turn off is needed to reject dribble bits. This is accomplished by an AC timing circuit that reacts to high level signals of greater than typically 200 ns in duration. The Receiver then stays off only if within about 1 μs, the DC level from the low pass filter rises above the DC squelch threshold. *Figure 2* illustrates the Receiver timing.

The differential line driver provides ECL compatible signals to the DTE with 5 ns rise and fall times. In its idle state, its outputs go to differential zero to prevent DC standing current in the isolation transformer.

3.2 TRANSMITTER FUNCTIONS

The Transmitter has a differential input and an open collector output current driver. The differential input common mode voltage is established by the CTI and should not be altered by external circuitry. The transformer coupling of TX$\pm$ will satisfy this condition. The driver meets all IEEE 802.3/Ethernet Specifications for signal levels. Controlled rise and fall times (25 ns $+/-5$ ns) minimize the higher harmonic components. The rise and fall times are matched to within 1 ns to minimize jitter. The drive current levels of the DP8392 meet the tighter recommended limits of IEEE 802.3 and are set by a built-in bandgap reference and an external 1% resistor. An on chip isolation diode is provided to reduce the Transmitter's coax load capacitance. For Ethernet compatible applications, an external isolation diode (see *Figure 4*) may be added to further reduce coax load capacitance. In Cheapernet compatible applications the external diode is not required as the coax capacitive loading specifications are relaxed.

The Transmitter squelch circuit rejects signals with pulse widths less than typically 25 ns (negative going), or with levels less than -175 mv. The Transmitter turns off at the end of the packet if the signal stays higher than -175 mv for more than approximately 300 ns. *Figure 3* illustrates the Transmitter timing.

3.0 Functional Description (Continued)

3.3 COLLISION FUNCTIONS

The collision circuitry consists of a 4-pole Bessel low pass filter (section 3.1), a comparator, a heartbeat generator, a 10 MHz oscillator, and a differential line driver.

The collision comparator monitors the DC level from the low pass filter. If the level is more negative than the collision threshold, the collision output is enabled.

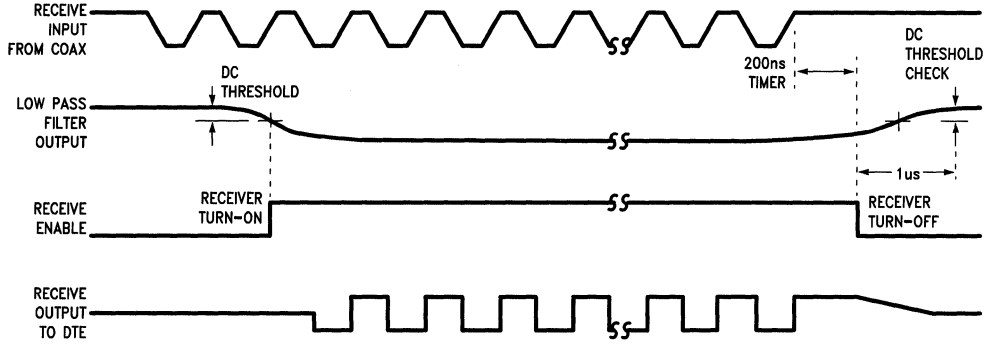
At the end of every transmission, the heartbeat generator creates a pseudo collision for a short time to ensure that the collision circuitry is properly functioning. This burst on collision output occurs typically 1.1 μs after the transmission, and has a duration of about 1 μs. This function can be disabled externally with the HBE (Heartbeat Enable) pin to allow operation with repeaters.

The 10 MHz oscillator generates the signal for the collision and heartbeat functions. It is also used as the timebase for all the jabber functions. It does not require any external components.

The collision differential line driver transfers the 10 MHz signal to the CD± pair in the event of collision, jabber, or heartbeat conditions. This line driver also features zero differential idle state.

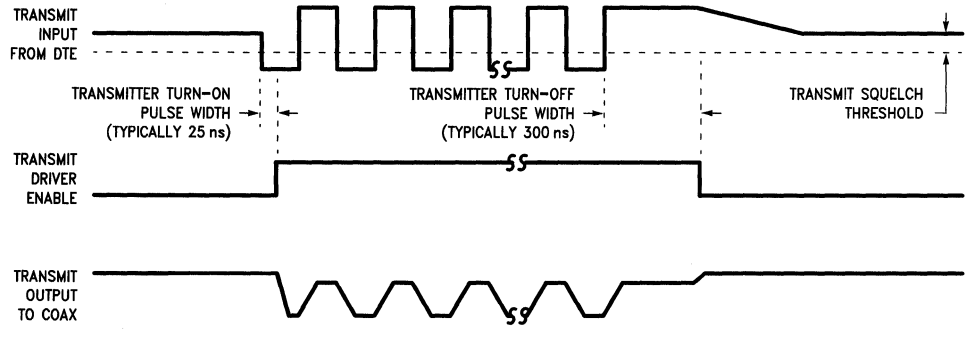
3.4 JABBER FUNCTIONS

The Jabber Timer monitors the Transmitter and inhibits transmission if the Transmitter is active for longer than 20 ms (fault). It also enables the collision output for the fault duration. After the fault is removed, The Jabber Timer waits for about 500 ms (unjab time) before re-enabling the Transmitter. The transmit input must stay inactive during the unjab time.



TL/F/7405–3

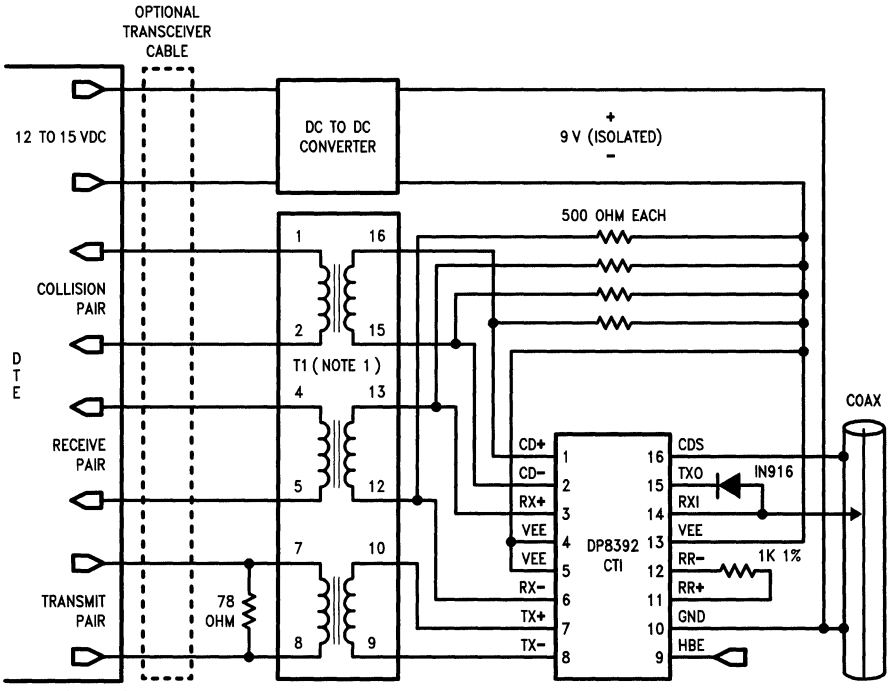
FIGURE 2. Receiver Timing



TL/F/7405–4

FIGURE 3. Transmitter Timing

4.0 Connection Diagram



Note 1: T1 is a 1:1 pulse transformer, L = 100 μH
Pulse Engineering (San Diego) Part No. 64103
Valor Electronics (San Diego)
Part No. 1101 or equivalent

Top View

Order Number DP8392N
See NS Package Number N16A

FIGURE 4. Connection Diagram

5

5.0 Pin Descriptions

Pin No.	Name	I/O	Description
1 2	CD+* CD−	O	**Collision Output.** Balanced differential line driver outputs from the collision detect circuitry. The 10 MHz signal from the internal oscillator is transferred to these outputs in the event of collision, excessive transmission (jabber), or during CD Heartbeat condition. These outputs are open emitters; pulldown resistors to VEE are required. When operating into a 78Ω transmission line, these resistors should be 500Ω. In Cheapernet applications, where the 78Ω drop cable is not used, higher resistor values (up to 1.5k) may be used to save power.
3 6	RX+* RX−	O	**Receive Output.** Balanced differential line driver outputs from the Receiver. These outputs also require 500Ω pulldown resistors.
7 8	TX+* TX−	I	**Transmit Input.** Balanced differential line receiver inputs to the Transmitter. The common mode voltage for these inputs is determined internally and must not be externally established. Signals meeting Transmitter squelch requirements are waveshaped and output at TXO.
9	HBE	I	**Heartbeat Enable.** This input enables CD Heartbeat when grounded, disables it when connected to VEE.
11 12	RR+ RR−	I	**External Resistor.** A fixed 1k 1% resistor connected between these pins establishes internal operating currents.
° 14	RXI	I	**Receive Input.** Connects directly to the coaxial cable. Signals meeting Receiver squelch requirements are equalized for inter-symbol distortion, amplified, and outputted at RX±.
15	TXO	O	**Transmit Output.** Connects either directly (Cheapernet) or via an isolation diode (Ethernet) to the coaxial cable.
16	CDS	I	**Collision Detect Sense.** Ground sense connection for the collision detect circuit. This pin should be connected separately to the shield to avoid ground drops from altering the receive mode collision threshold.
10	GND		**Positive Supply Pin.** A 0.1 μF ceramic decoupling capacitor must be connected across GND and VEE as close to the device as possible.
4 5 13	VEE		**Negative Supply Pins.** In order to make full use of the 3.5W power dissipation capability of this package, these pins should be connected to a large metal frame area on the PC board. Doing this will reduce the operating die temperature of the device thereby increasing the long term reliability.

* IEEE names for CD± = CI±, RX± = DI±, TX± = DO±

6.0 Absolute Maximum Ratings (Note 1)

Supply Voltage (V_{EE}) −12V

Package Power Rating at 25°C 3.5 Watts*
(PC Board Mounted)
Derate linearly at the rate of 28.6 mW/°C

Input Voltage 0 to −12V

*For actual power dissipation of the device please refer to section 7.0.

Storage Temperature −65° to 150°C

Lead Temp. (Soldering, 10 seconds) 300°C

Recommended Operating Conditions

Supply Voltage (V_{EE}) −9v ±5%

Ambient Temperature 0° to 70°C

7.0 Electrical Characteristics V_{EE} = −9V ±5%, T_A = 0° to 70°C (Notes 2 & 3)

Symbol	Parameter	Min	Typ	Max	Units
I_{EE1}	Supply current—non transmitting		−85	−130	mA
I_{EE2}	Supply current—transmitting		−125	−180	mA
I_{RXI}	Receive input bias current (RXI)	−2		+25	μA
I_{TDC}	Transmit output dc current level (TXO)	37	41	45	mA
I_{TAC}	Transmit output ac current level (TXO)	±28		I_{TDC}	mA
V_{CD}	Collision threshold (Receive mode)	TBD	−1.53	TBD	V
V_{OD}	Differential output voltage (RX±, CD±)	±600		±1200	mV
V_{OC}	Common mode output voltage (RX±, CD±)	−1.5	−2.0	−2.5	V
V_{OB}	Diff. output voltage imbalance (RX±, CD±)			±40	mV
V_{TS}	Transmitter squelch threshold (TX±)	−175	−225	−300	mV
C_X	Input capacitance (RXI)		1.2		pF
R_{RXI}	Shunt resistance—non transmitting (RXI)	100			KΩ
R_{TXO}	Shunt resistance—transmitting (TXO)	TBD	10		KΩ

8.0 Switching Characteristics V_{EE} = −9V ±5%, T_A = 0° to 70°C (Note 3)

Symbol	Parameter	Fig	Min	Typ	Max	Units
t_{RON}	Receiver startup delay (RXI to RX±)	5 & 11		4	TBD	bits
t_{Rd}	Receiver propagation delay (RXI to RX±)	5 & 11		15	50	ns
t_{Rr}	Differential outputs rise time (RX±, CD±)	5 & 11		4	TBD	ns
t_{Rf}	Differential outputs fall time (RX±, CD±)	5 & 11		4	TBD	ns
t_{RJ}	Receiver & cable total jitter	10		±2	TBD	ns
t_{TST}	Transmitter startup delay (TX± to TXO)	6 & 11		1	TBD	bits
t_{Td}	Transmitter propagation delay (TX± to TXO)	6 & 11		25	50	ns
t_{Tr}	Transmitter rise time —10% to 90% (TXO)	6 & 11	TBD	25	TBD	ns
t_{Tf}	Transmitter fall time —90% to 10% (TXO)	6 & 11	TBD	25	TBD	ns
t_{TM}	t_{Tr} and t_{Tf} mismatch			0.5	TBD	ns
t_{TS}	Transmitter skew (TXO)			±0.5	TBD	ns
t_{TON}	Transmit turn-on pulse width at V_{TS} (TX±)	6 & 11	TBD	25		ns
t_{TOFF}	Transmit turn-off pulse width at V_{TS} (TX±)	6 & 11	TBD	300	TBD	ns
t_{CON}	Collision turn-on delay	7 & 11		7	TBD	bits
t_{COFF}	Collision turn-off delay	7 & 11			20	bits
f_{CD}	Collision frequency (CD±)	7 & 11	8.0		12.5	MHz
t_{CP}	Collision pulse width (CD±)	7 & 11	35		70	ns
t_{HON}	CD Heartbeat delay (TX± to CD±)	8 & 11	0.6		1.6	μs
t_{HW}	CD Heartbeat duration (CD±)	8 & 11	0.5	1.0	1.5	μs
t_{JA}	Jabber activation delay (TX± to TXO and CD±)	9 & 11	20	29	50	ms
t_{JR}	Jabber reset unjab time (TX± to TXO and CD±)	9 & 11	250	500	750	ms

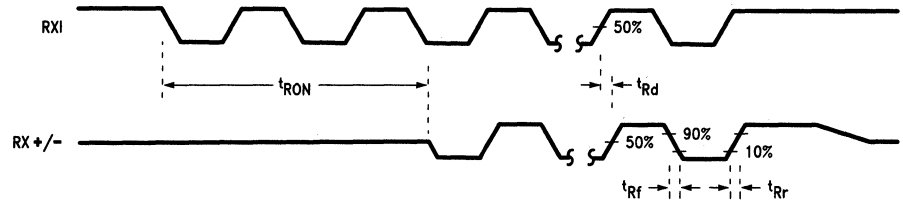
TBD = To be determined

Note 1: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits.

Note 2: All currents into device pins are positive, all currents out of device pins are negative. All voltages referenced to ground unless otherwise specified.

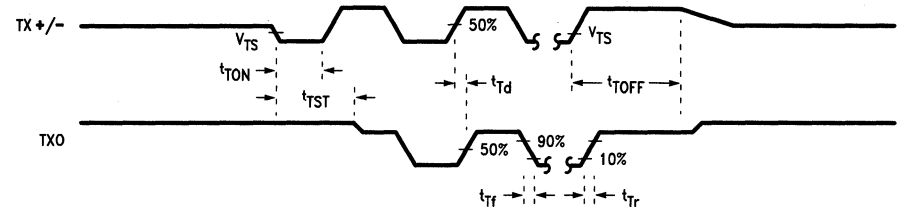
Note 3: All typicals are given for V_{EE} = −9V and T_A = 25°C.

9.0 Timing and Load Diagrams



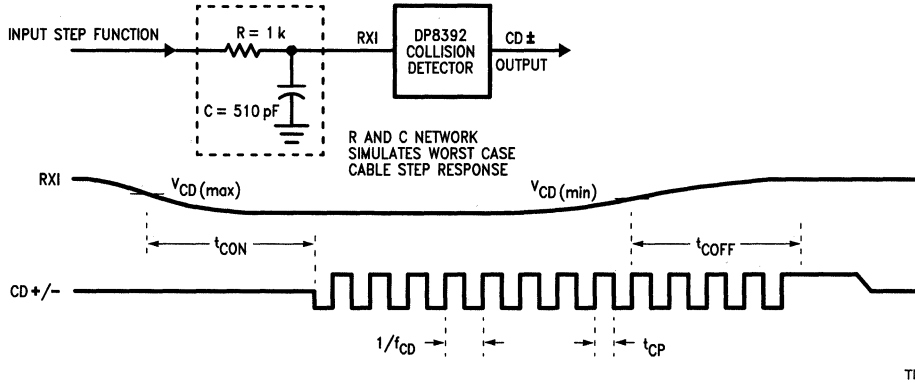
TL/F/7405–6

FIGURE 5. Receiver Timing



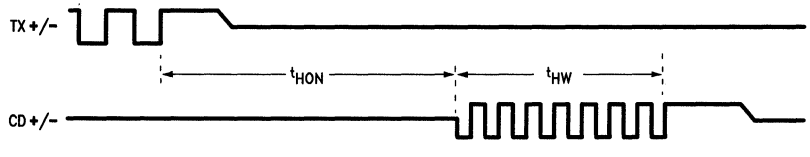
TL/F/7405–7

FIGURE 6. Transmitter Timing



TL/F/7405–8

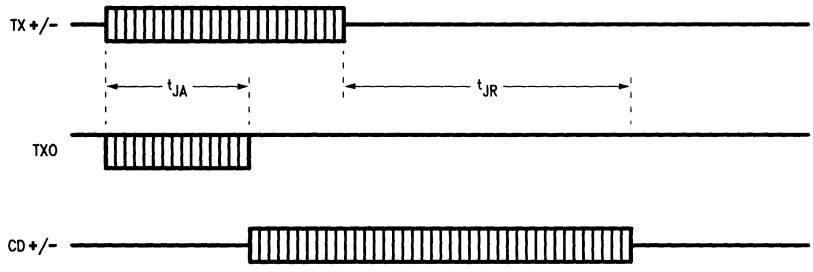
FIGURE 7. Collision Timing



TL/F/7405–9

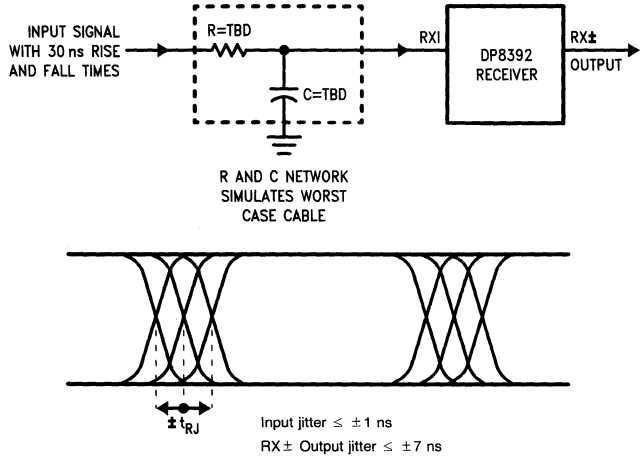
FIGURE 8. Heartbeat Timing

9.0 Timing and Load Diagrams (Continued)



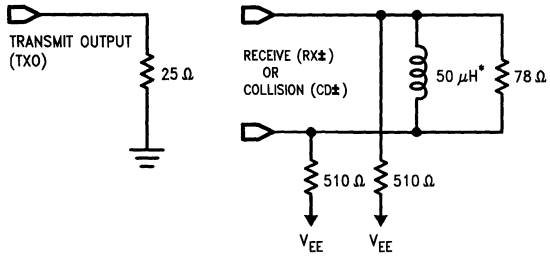
TL/F/7405–10

FIGURE 9. Jabber Timing



TL/F/7405–11

FIGURE 10. Receive Jitter Timing



TL/F/7405–12

*The 50 μH inductance is for testing purposes. Pulse transformers with higher inductances are recommended (see *Figure 4*)

FIGURE 11. Test Loads

National Semiconductor Corporation

DP8340/NS32440 Serial Bi-Phase Transmitter/Encoder

General Description

The DP8340/NS32440 generates a complete encoding of parallel data for high speed serial transmission which conforms to the protocol as defined by the IBM 3270 information display system standard. The DP8340/NS32440 converts parallel input data into a serial data stream. Although the IBM standard covers bi-phase serial data transmission over a coax line, the DP8340/NS32440 also adapts to general high speed serial data transmission over other than coax lines, at frequencies either higher or lower than the IBM standard.

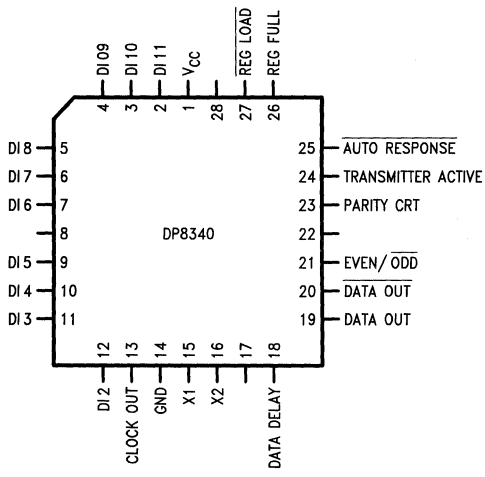
The DP8340/NS32440 and its complementary chip, the DP8341 (receiver/decoder) have been designed to provide maximum flexibility in system designs. The separation of the transmitter/receiver functions provides convenient addition of more receivers at one end of a bi-phase line without the need of unused transmitters. This is specifically advantageous in control units where typical bi-phase data is multiplexed over many bi-phase lines and the number of receivers generally exceeds the number of transmitters.

Features

- Ten bits per data byte transmission
- Single-byte or multi-byte transmission
- Internal parity generation (even or odd)
- Internal crystal controlled oscillator used for the generation of all required chip timing frequencies
- Clock output directly drives receiver (DP8341) clock input
- Input data holding register
- Automatic clear status response feature
- Line drivers at data outputs provide easy interface to bi-phase coax line or general transmission lines
- < 2 ns driver output skew
- Bipolar technology provides TTL input/output compatibility
- Data outputs power up/down glitch free
- Internal power up clear and reset
- Single $+5$V power supply

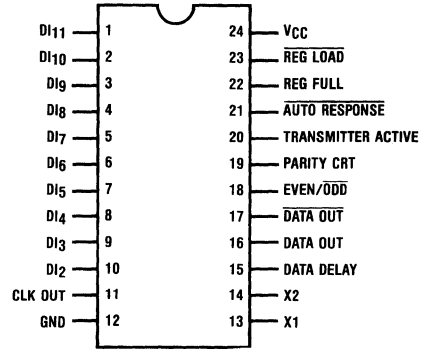
Connection Diagrams

Plastic Chip Carrier



TL/F/5251–19

Dual-In-Line Package



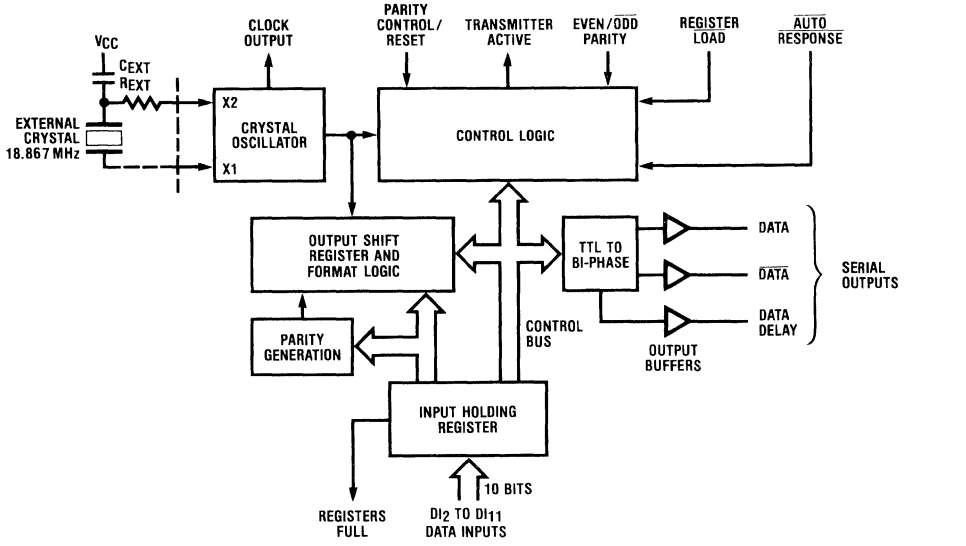
TL/F/5251–1

Top View

FIGURE 1

Order Number DP8340/NS32440 J, N or V
See NS Package Number J24A, N24A or V28A

Block Diagram



TL/F/5251–2

FIGURE 2. DP8340/NS32440 Serial Bi-Phase Transmitter/Encoder Block Diagram

Functional Description

Figure 2 is a block diagram of the DP8340/NS32440 Bi-Phase Transmitter/Encoder. The transmitter/encoder contains a crsytal oscillator whose input is a crystal with a frequency eight (8) times the data rate. A Clock Output is provided to drive the DP8341 receiver/decoder Clock Input and other system components at the oscillator frequency. Additionally, the oscillator drives the control logic and output shift register/format logic blocks.

Data is parallel loaded from the sytem data bus to the transmitter/encoder's input holding register. This data is in turn loaded by the transmitter/encoder to its output shift register if this register was empty at the time of the load. During this load, message formatting and parity are generated. The formatted message is then shifted out at the bit rate frequency to the TTL to Bi-Phase block which generates the proper data bit formatting. The three data outputs, DATA, $\overline{\text{DATA}}$, and DATA DELAY provide for flexible interface to the coax line with a minimum of external components.

The Control Logic block interfaces to all blocks to insure proper chip operation and sequencing. It controls the type of parity generation through the Even/Odd Parity input. An additional feature provided by the transmitter/encoder is generation of odd parity and placement in bit 10 position while still maintaining even or odd parity in the bit 12 position. This is the format of data word bytes and other commands in the 3270 Standard. The Parity Control input is the pin which controls when this operation is in effect.

Another feature of the transmitter/encoder is the internal TT/AR (Transmission Turnaround/Auto Response) capability. After each Write type message from the control unit in the 3270 Standard, the receiving unit must respond with clean status (bits 2 through 11). With the transmitter/encoder, this function is accomplished simply by forcing the $\overline{\text{Auto-Response}}$ input to the Logic "0" state.

Operation of the transmitter/encoder is automatic. After the first data byte is loaded, the Transmitter Active output is set and the transmitter/encoder immediately formats the input data and serially shifts it out its data outputs. If the message is a multi-byte message, the internal format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding register before the last data bit of the previous data byte is transferred out of the internal output shift register. After all data is shifted out of the transmitter/encoder the Transmitter Active output will return to the inactive state.

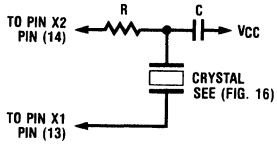
Detailed Pin/Functional Description

Crystal Inputs X1 and X2

The oscillator is controlled by an external, parallel resonant crystal connected between the X1 and X2 pins. Normally, a fundamental mode crystal is used to determine the operating frequency of the osicllator; however, overtone mode crystals may be used.

Crystal Specifications (Parallel Resonant)

Type	AT-cut crystal
Tolerance	0.005% at 25°C
Stability	0.01% from 0°C to +70°C
Resonance	Fundamental (Parallel)
Maximum Series Resistance	Dependent on Frequency (For 18.867 MHz, 50Ω)
Load Capacitance	15 pF



FREQ	R	C
10 MHz to 20 MHz	500Ω ±10%	30 pF
>20 MHz	120Ω ±10%	15 pF

TL/F/5251–3

FIGURE 3. Connection Diagram

If the DP8340/NS32440 transmitter is clocked by a system (clock crystal oscillator not used), pin 13 (X1 input) should be clocked directly using a Schottky series (74S) circuit. Pin 14 (X2 input) may be left open. The clocking frequency must be set at eight times the data bit rate. Maximum input frequency is 28 MHz. For the IBM 3270 Interface, this frequency is 18.867 MHz. At this frequency, the serial bit rate will be 2.358 Mbits/sec.

Clock Output

The Clock Output is a buffered output derived directly from the crystal oscillator block and clocks at the oscillator frequency. It is designed to directly drvie the DP8341 receiver/decoder Clock Input as well as other system components.

Registers Full

This output is used as a flag by the external operating system. A logic "1" (active state) on this output indicates that both the internal output shift register and the input holding register contain active data. No additional data should be loaded until this output returns to the logic "0" state (inactive state).

Transmitter Active

This output will be in the logic "1" state while the transmitter/encoder is about to transmit or in the process of transmitting data. Otherwise, it will assume the logic "0" state indicating no data presently in either the input holding or output shift registers.

$\overline{\text{Register Load}}$

The $\overline{\text{Register Load}}$ input is used to load data from the Data Inputs to the input holding register. The loading function is edge sensitive, the data present during the logic "0" state of this input is loaded, and the input data must be valid before the logic "0" to logic "1" transition. It is after this transition that the transmitter/encoder begins formatting of data for serial transmission.

$\overline{\text{Auto Response}}$ (TT/AR)

This input provides for automatic clear data transmission (all bits in logic "0") without the need of loading all zero's. When a logic "0" is forced on this inpiut the transmitter/encoder immediately responds with transmission of "clean status". This function is necessary after the completion of each write type command and in other functions in the 3270 specification. In the logic "1" state the transmitter/encoder transmits data entered on the Data Inputs.

Even/$\overline{\text{Odd}}$ Parity

This input sets the internal logic of the DP8340/NS32440 transmitter/encoder to generate either even or odd parity for the data byte in the bit 12 position. When this pin is in the logic "0" state odd parity is generated. In the logic "1" state even parity is generated. This feature is useful when the control unit is performing a loop back check and at the same time the controller wishes to verify proper data transmission with its receiver/decoder.

Parity Control/Reset

Depending on the type of message transmitted, it is at times necessary in the IBM 3270 specification to generate an additional parity bit in the bit 10 position. The bit generated is odd parity on the previous eight (8) bits of data. When the Parity Control input is in the logic "1" state the data entered at the Data Bit 10 position is placed in the transmitted word. With the Parity Control input in the logic "0" state the Data Bit 10 input is ignored and odd parity on the previous data bits is placed in the normal bit 10 position while overall word parity (bit 12) is even or odd (controlled by Even/$\overline{\text{Odd}}$ Parity input). This eliminates the need for external logic to generate the parity on the data bits.

Truth Table

Parity Control Input	Transmitted Data Bit 10
Logic "1"	Data entered on Data Input 10
Logic "0"	Odd Parity on 8-bit data byte

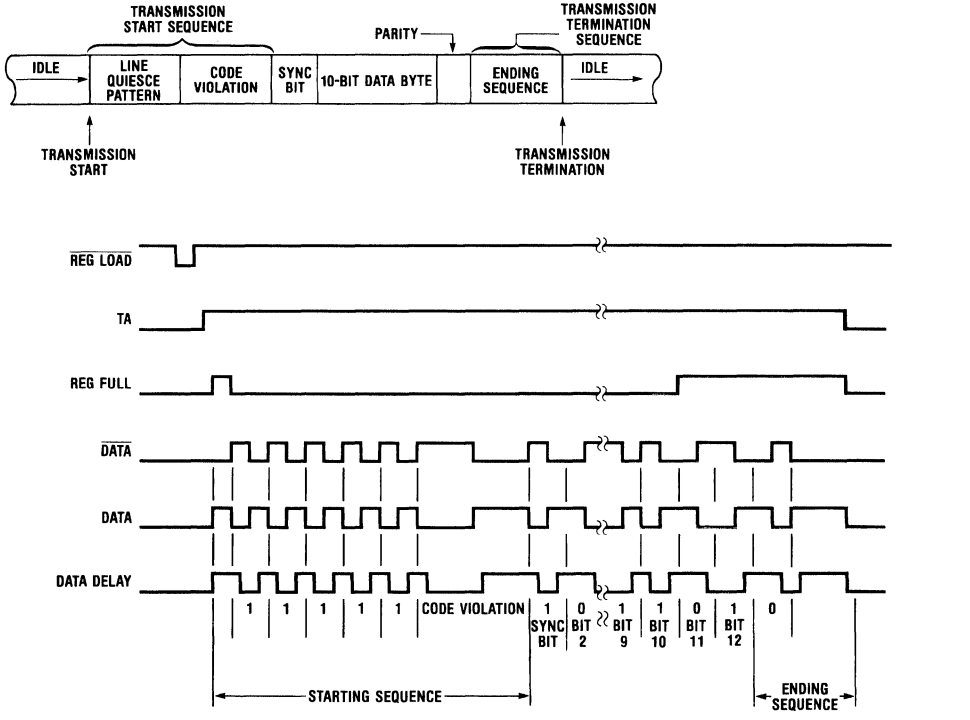
When this input is driven to a voltage that exceeds the power supply level (9V to 13V) the transmitter/encoder is reset.

Serial Outputs—DATA, $\overline{\text{DATA}}$, and DATA DELAY

These three output pins provide for convenient application of data to the Bi-Phase Coax line (see *Figure 15* for application). The Data outputs are a direct bit representation of the Bi-Phase data while the DATA DELAY output provides the necessary increment to clearly define the four (4) DC levels of the pulse. The DATA and $\overline{\text{DATA}}$ outputs add flexibility to the DP8340/NS32440 transmitter/encoder for use in high speed differential line driving applications.

Functional Timing Waveforms—Message Format

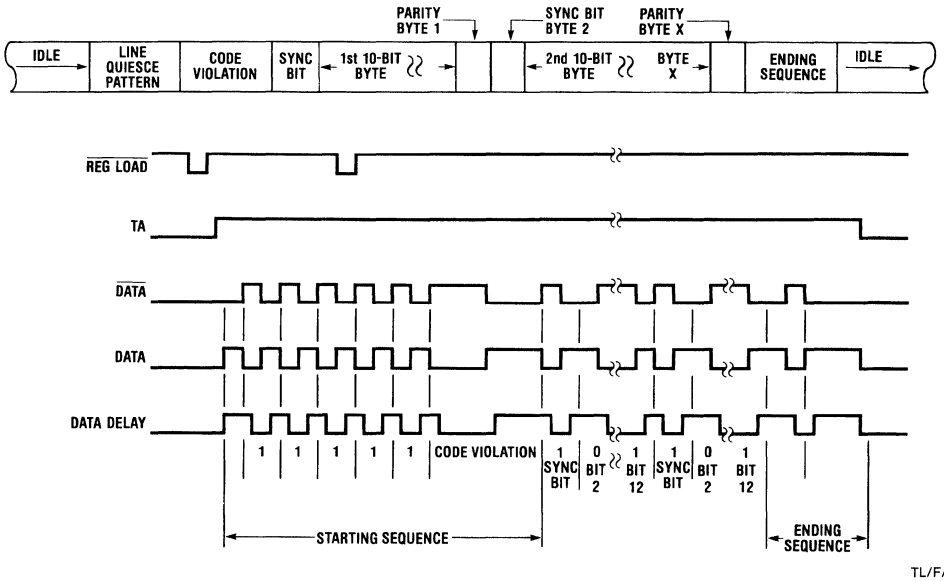
Single Byte Transmission



TL/F/5251–4

FIGURE 4. Overall Timing Waveforms for Single Byte

Multi-Byte Transmission



TL/F/5251–5

FIGURE 5. Overall Timing Waveforms for Multi-Byte

5

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage, V_{CC}	7V
Input Voltage	5.5V
Output Voltage	5.25V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Maximum Power Dissipation @25°C*

Cavity Package	2237 mW
Dual-In-Line Package	2500 mW
Plastic Chip Carrier	1720 mw

*Derate cavity package 14.9 mW/°C above 25°C; derate dual-in-line package 20 mW/°C above 25°C; derate PCC package 13.8 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, T_A	0	+70	°C

Electrical Characteristics (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logic "1" Input Voltage (All Inputs Except X1 and X2)		2.0			V
V_{IL}	Logic "0" Input Voltage (All Inputs Except X1 and X2)				0.8	V
V_{CLAMP}	Input Clamp Voltage (All Inputs Except X1 and X2)	I_{IN} = −12 mA		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current Register Load Input	V_{CC} = 5.25V, V_{IN} = 5.25V		0.3	120	μA
	All Others Except X1 and X2			0.1	40	μA
I_{IL}	Logic "0" Input Current Register Load Input	V_{CC} = 5.25V, V_{IN} = 0.5V		−15	−300	μA
	All Inputs Except X1 and X2			−5	−100	μA
V_{OH1}	Logic "1" All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY	I_{OH} = −100 μA	3.2	3.9		V
		I_{OH} = −1 mA	2.5	3.4		V
V_{OH2}	Logic "1" for CKL OUT, DATA, $\overline{DATA}$ and DATA DELAY Outputs	I_{OH} = −10 mA	2.6	3.0		V
V_{OL1}	Logic "0" All Outputs Except CLK OUT, DATA, $\overline{DATA}$ and DATA DELAY Outputs	I_{OL} = 5 mA		0.35	0.5	V
V_{OL2}	Logic "0" for CLK OUT, DATA, $\overline{DATA}$ and DATA DELAY Outputs	I_{OL} = 20 mA		0.4	0.6	V
I_{OS1}	Short Circuit Current for All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY	V_{OUT} = 0V (Note 4)	−10	−30	−100	mA
I_{OS2}	Short Circuit Current for DATA, $\overline{DATA}$, and DATA DELAY Outputs	V_{OUT} = 0V (Note 4)	−50	−140	−350	mA
I_{OS3}	Short Circuit Current for CLK OUT	(Note 4)	−30	−90	−200	mA
I_{CC}	Power Supply Current	V_{CC} = 5.25V		170	250	mA

Timing Characteristics Oscillator Frequency = 18.867 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd1}	$\overline{REG\ LOAD}$ to Transmitter Active (T_A) Positive Edge	Load Circuit 1 *Figure 7*		60	90	ns
t_{pd2}	$\overline{REG\ LOAD}$ to REG Full; Positive Edge	Load Circuit 1 *Figure 7*		45	75	ns
t_{pd3}	Register Full to T_A; Negative Edge	Load Circuit 1 *Figure 7*		40	70	ns
t_{pd4}	Positive Edge of $\overline{REG\ LOAD}$ to Positive Edge of DATA	Load Circuits 1 & 2 *Figure 9*		50	80	ns

Timing Characteristics Oscillator Frequency = 18.867 MHz (Notes 2 and 3) (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd5}	$\overline{\text{REG LOAD}}$ to $\overline{\text{DATA}}$; Positive Edge	Load Circuits 1 & 2 *Figure 9,* (Note 6)		380	475	ns
t_{pd6}	$\overline{\text{REG LOAD}}$ to DATA DELAY; Positive Edge	Load Circuits 1 & 2 *Figure 9,* (Note 6)		160	250	ns
t_{pd7}	Positive Edge of $\overline{\text{DATA}}$ to Negative Edge of DATA DELAY	Load Circuit 2 *Figure 9,* (Note 6)		100	115	ns
t_{pd8}	Positive Edge of DATA DELAY to Negative Edge of $\overline{\text{DATA}}$	Load Circuit 2 *Figure 9,* (Note 6)		110	125	ns
t_{pd9}, t_{pd10}	Skew between DATA and $\overline{\text{DATA}}$	Load Circuit 2 *Figure 9*		2	6	ns
t_{pd11}	Negative Edge of Auto Response to Positive Edge of TA	Load Circuit 1 *Figure 10*		70	110	ns
t_{pd12}	Maximum Time Delay to Load Second Byte after Positive Edge of REG FULL	Load Circuit 1 *Figure 8,* (Note 6)			$4 \times T - 50$	ns
t_{pd13}	X1 to CLK OUT; Positive Edge	Load Circuit 2 *Figure 13*		21	30	ns
t_{pd14}	X1 to CLK OUT; Negative Edge	Load Circuit 2 *Figure 13*		23	33	ns
t_{pd15}	Negative Edge of AR to Positive Edge of REG FULL	Load Circuit 1 *Figure 10*		45	75	ns
t_{pd16}	Skew between TA and REG FULL during Auto Response	Load Circuit 1 *Figure 10*		50	80	ns
t_{pd17}	$\overline{\text{REG LOAD}}$ to REG FULL; Positive Edge for Second Byte	Load Circuit 1 *Figure 14*		45	75	ns
t_{pw1}	$\overline{\text{REG LOAD}}$ Pulse Width	*Figure 12*	40			ns
t_{pw2}	First REG FULL Pulse Width (Note 5)	Load Circuit 1 *Figure 7,* (Note 6)		$8 \times T + 60$	$8 \times T + 100$	ns
t_{pw3}	REG FULL Pulse Width prior to Ending Sequence (Note 5)	Load Circuit 1, *Figure 7,* (Note 6)		$5 \times B$		ns
t_{pw4}	Pulse Width for Auto Response	*Figure 10*	40			ns
t_S	Data Setup Time prior to REG LOAD Positive Edge, Hold Time (t_H) = 0 ns	*Figure 12*		15	25	ns
t_{r1}	Rise Time for DATA, $\overline{\text{DATA}}$, and DATA DELAY Output Waveform	Load Circuit 2 *Figure 11*		7	13	ns
t_{f1}	Fall Time for DATA, $\overline{\text{DATA}}$, and DATA DELAY Output Waveform	Load Circuit 2 *Figure 11*		5	11	ns
t_{r2}	Rise Time for TA and REG FULL	Load Circuit 1 *Figure 15*		20	30	ns
t_{f2}	Fall Time for TA and REG FULL	Load Circuit 1 *Figure 15*		15	25	ns
f_{MAX}	Data Rate Frequency (Clock Input must be 8X this Frequency)	(Note 7)	DC		3.5	Mbits/s

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute basis.

Note 4: Only one output should be shorted at a time. Output should not be shorted for more than one second at a time.

Note 5: T = 1/(Oscillator Frequency), unit for T should be ns. B = 8T

Note 6: Oscillator Frequency Dependent.

Note 7: For the IBM 3270 Interface, the data rate frequency is 2.358 Mbits/s. 28 MHz clock frequency corresponds to 3.75% jitter when referenced to *Figure 10* of DP8341 Datasheet.

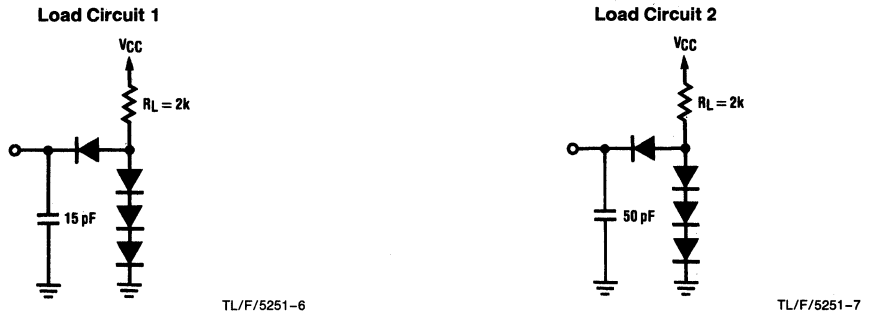


FIGURE 6. Test Load Circuits

Timing Waveforms

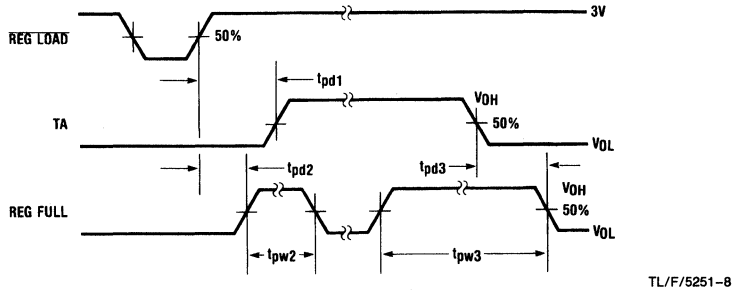


FIGURE 7. Timing Waveforms for Single Byte Transfer

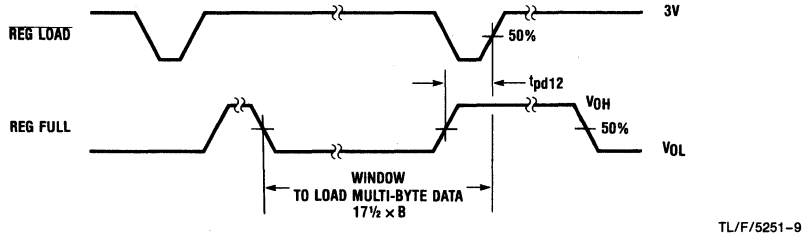


FIGURE 8. Maximum Window to Load Multi-Byte Data

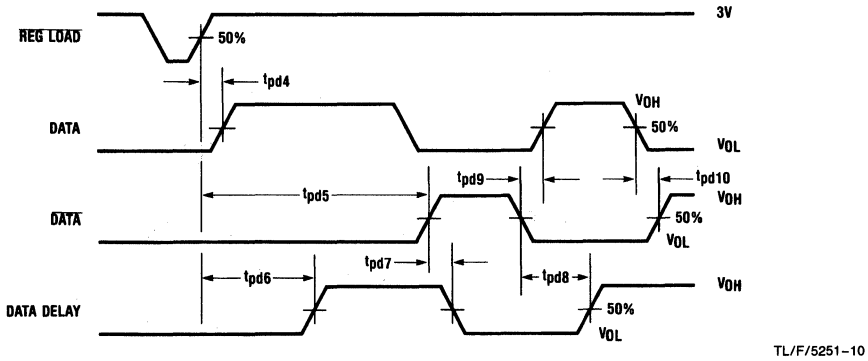
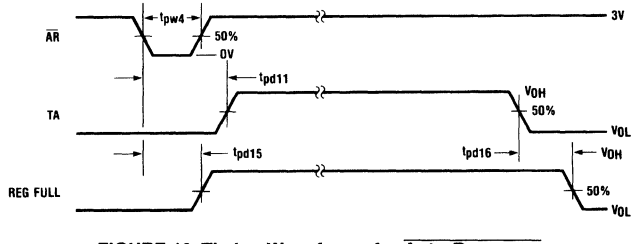


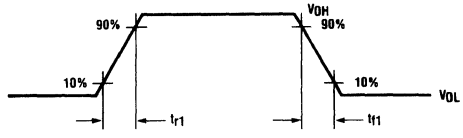
FIGURE 9. Timing Waveforms for Three Serial Outputs

Timing Waveforms (Continued)



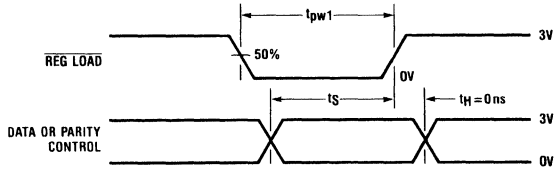
TL/F/5251–11

FIGURE 10. Timing Waveforms for $\overline{\text{Auto-Response}}$



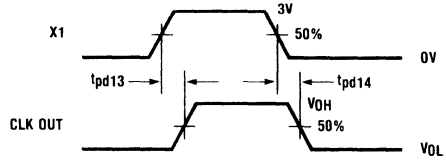
TL/F/5251–12

FIGURE 11. Output Waveform for DATA, $\overline{\text{DATA}}$, DATA DELAY (Load Circuit 2)



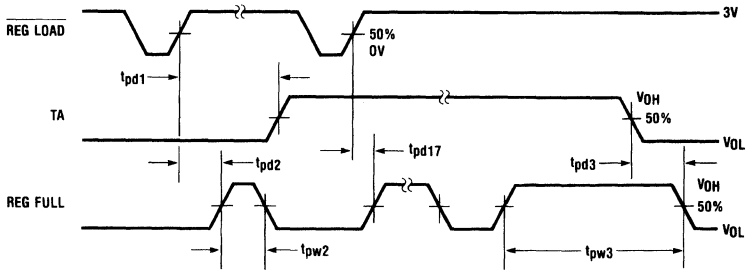
TL/F/5251–13

FIGURE 12. Register Load Waveform Requirement



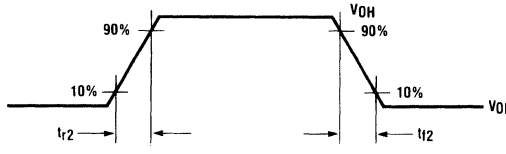
TL/F/5251–14

FIGURE 13. Timing Waveforms for Clock Pulse



TL/F/5251–15

FIGURE 14. Timing Waveforms for Two Byte Transfer

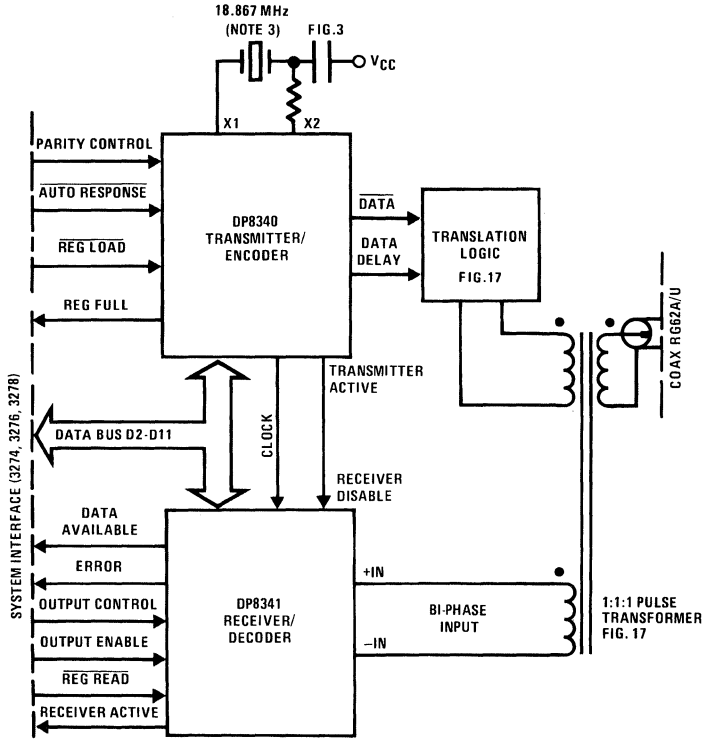


TL/F/5251–16

FIGURE 15. Rise and Fall Time Measurement for TA and REG Full

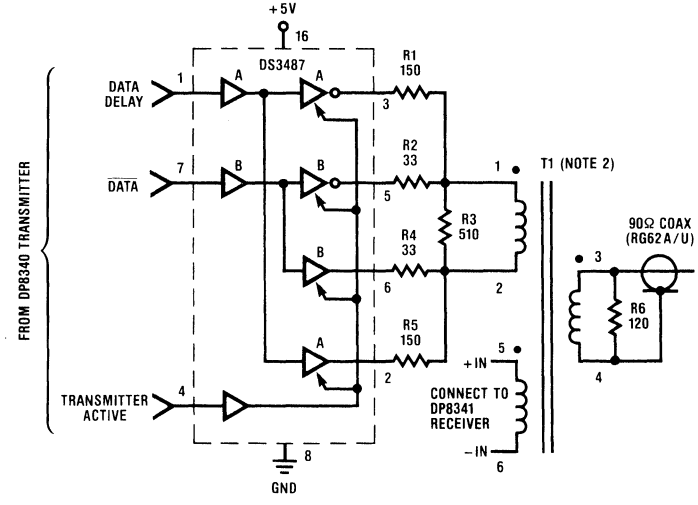
5

Typical Applications



TL/F/5251–17

FIGURE 16. Typical Applications for IBM 3270 Interface



TL/F/5251–18

Note 1: Resistance values are in Ω, ±5%, ¼ W

Note 2: T1 is a 1:1:1 pulse transformer, L_{MIN} = 500 μH for 18 MHz system clock. Pulse Engineering Part No. 5762/Surface Mount, 5762M/PE-85762. Technitrol Part No. 11LHA, Valor Electronics Part No. CT1501 or equivalent transformers.

Note 3: Crystal manufacturer's Midland Ross Corp. NEL Unit Part No. NE-18A (C2560N) @ 18.867 MHz and the Viking Group of San Jose, CA Part No. VXB46NS @ 18.867 MHz.

FIGURE 17. Translation Logic

National Semiconductor Corporation

DP8341/NS32441 Serial Bi-Phase Receiver/Decoder

General Description

The DP8341/NS32441 provides complete decoding of data for high speed serial data communications. In specific, the DP8341/NS32441 recognizes serial data that conforms to the IBM 3270 Information Display System Standard and converts it into ten (10) bits of parallel data. Although this standard covers Bi-Phase serial data transmission over a coax line, this device easily adapts to generalized high speed serial data transmission on other than coax lines at frequencies either higher or lower than the IBM 3270 standard.

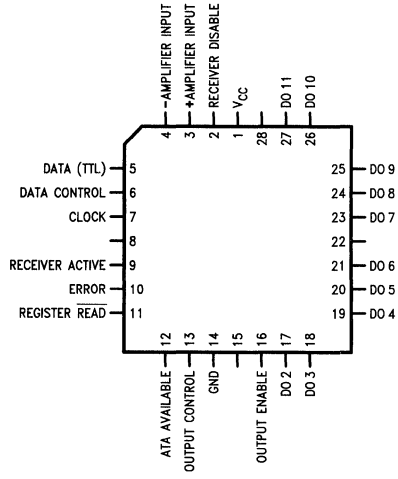
The DP8341/NS32441 receiver and its complementary chip, the DP8340 transmitter, are designed to provide maximum flexibility in system designs. The separation of transmitter and receiver functions allows addition of more receivers at one end of the Bi-Phase line without the necessity of adding unused transmitters. This is advantageous specifically in control units where typically Bi-Phase data is multiplexed over many Bi-Phase lines and the number of receivers generally outnumber the number of transmitters. The separation of transmitter and receiver function provides an additional advantage in flexibility of data bus organization. The data bus outputs of the receiver are TRI-STATE®, thus enabling the bus configuration to be organized as either a common transmit/receive (bi-directional) bus or as separate transmit and receive busses for higher speed.

Features

- DP8341/NS32441 receivers ten (10) bit data bytes and conforms to the IBM 3270 Interface Display System Standard
- Separate receiver and transmitter provide maximum system design flexibility
- Even parity detection
- High sensitivity input on receiver easily interfaces to coax line
- Standard TTL data input on receiver provides generalized transmission line interface and also provides hysteresis
- Data holding register
- Multi-byte or single byte transfers
- TRI-STATE receiver data outputs provide flexibility for common or separated transmit/receive data bus operation
- Data transmission error detection or receiver provides for both error detection and error type definition
- Bi-polar technology provides TTL input/output compatibility with excellent drive characteristics
- Single +5V power supply operation

Connection Diagrams

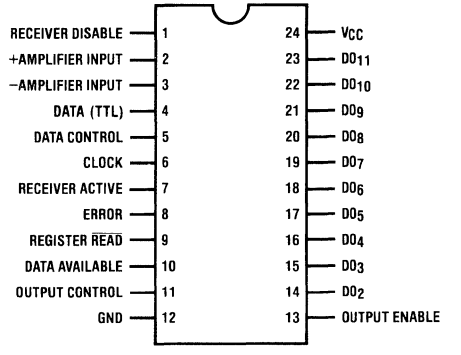
Plastic Chip Carrier



TL/F/5238–1

Order Number DP8341V or NS32441V
See NS Package Number V28A

Dual-In-Line Package



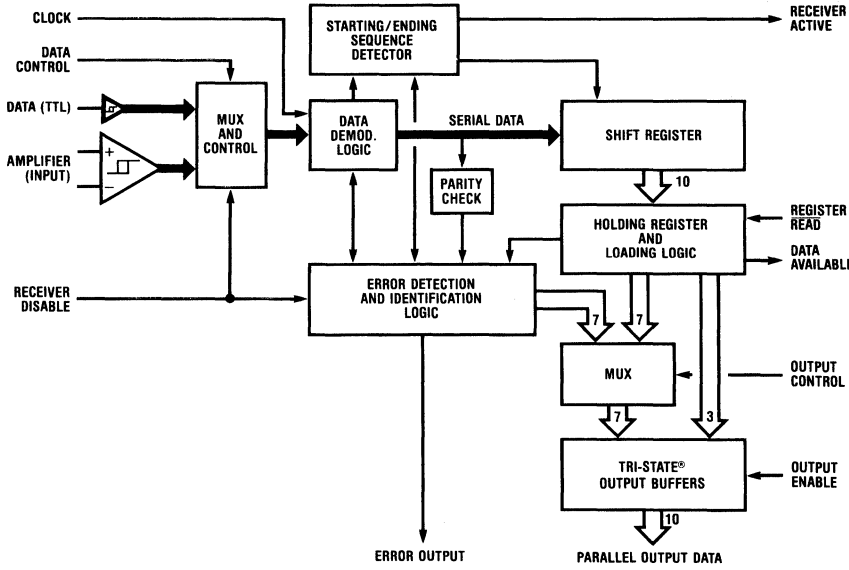
TL/F/5238–2

Top View

Order Number DP8341J or DP8341N
See NS Package Number J24A or N24A

FIGURE 1

Block Diagram



TL/F/5238–3

FIGURE 2. DP8341/NS32441 Serial Bi-Phase Receiver/Decoder Block Diagram

Block Diagram Functional Description

Figure 2 is a block diagram of the DP8341/NS32441. This chip is essentially a serial in/parallel out shift register. However, the serial input data must conform to a very specific format (see *Figures 3–5*). The message will not be recognized unless the format of the starting sequence is correct. Deviations from the format in the data, sync bit, parity or ending sequence will cause an error to be detected, terminating the message.

Data enters the receiver through the differential input amplifier or the TTL Data input. The differential amplifier is a high sensitivity input which may be used by connecting it directly to a transformer coupled coax line, or other transmission medium. The TTL Data input provides 400 mV of hysteresis and recognizes TTL logic levels. The data then enters the demodulation block.

The data demodulation block samples the data at eight (8) times the data rate and provides signals for detecting the starting sequence, ending sequence, and errors. Detection of the starting sequence sets the Receiver Active output high and enables the input shift register.

As the ten bits of data are shifted into the shift register, the receiver will verify that even parity is maintained on the data bits and the sync bit. After one complete data byte is received, the contents of the input shift register is parallel loaded to the holding register, assuming the holding register is empty, and the Data Available output is set. If the holding register is full, this load will be delayed until that register has been read. If another data byte is received when the shift register and the holding register are full a Data Overflow Error will be detected, terminating the message. Data is read from the holding register through the TRI-STATE Output Buffers. The Output Enable input is the TRI-STATE control for these outputs and the Register Read input signals the receiver that the read has been completed.

When the receiver detects an ending sequence the Receiver Active output will be reset to a logic "0" indicating the message has been terminated. A message will also terminate when an error is detected. The Receiver Active output used in conjunction with the Error output allows quick response to the transmitting unit when an error free message has been received.

The Error Detection and Identification block insures that valid data reaches the outputs of the receiver. Detection of an error sets the Error output to a logic "1" and resets the Receiver Active output to a logic "0" terminating the message. The error type may be read from the data bus outputs by setting the Output Control input to logic "0" and enabling the TRI-STATE outputs. The data bit outputs have assigned error definitions (see error code definition table). The Error output will return to a logic "0" when the next starting sequence is received, or when the error is read (Output Control to logic "0" and a Register Read performed).

The Receiver Disable input is used to disable both the amplifier and TTL Data receiver inputs. It will typically be connected directly to the Transmitter Active output of the DP8340 transmitter circuit (see *Figure 12*).

Detailed Functional Pin Description

RECEIVER DISABLE

This input is used to disable the receiver's data inputs. The Receiver Disable input will typically be connected to the Transmitter Active output of the DP8340. However, at the system controller it is necessary for both the transmitter and receiver to be active at the same time in the loop-back check condition. This variation can be accomplished with the addition of minimal external logic.

Truth Table

Receiver Disable	Data Inputs
Logic "0"	Active
Logic "1"	Disabled

AMPLIFIER INPUTS

The receiver has a differential input amplifier which may be directly connected to the transformer coupled coax line. The amplifier may also be connected to a differential type TTL line. The amplifier has 20 mV of hysteresis.

DATA INPUT

This input can be used either as an alternate data input or as a power-up check input. If the system designer prefers to use his own amplifier, instead of the one provided on the receiver, then this TTL input may be used. Using this pin as an alternate data input allows self-test of the peripheral system without disturbing the transmission line.

DATA CONTROL

This input is the control pin that selects which of the inputs are used for data entry to the receiver.

Truth Table

Data Control	Data input To
Logic "0"	Data Input
Logic "1"	Amplifier Inputs

Note: This input is also used for testing. When the input voltage is raised to 7.5V the chip resets.

CLOCK INPUT

The input is the internal clock of the receiver. It must be set at eight (8) times the line data bit rate. For the IBM 3270 Standard, this frequency is 18.87 MHz or a data bit rate of 2.358 MHz. The crystal-controlled oscillator provided in the DP8340 transmitter also operates at this frequency. The Clock Output of the transmitter is designed to directly drive the receiver's Clock Input. In addition, the receiver is designed to operate correctly to a data bit rate of 3.5 MHz.

RECEIVER ACTIVE

The purpose of this output is to inform the external system when the DP8341/NS32441 is in the process of receiving a message. This output will transition to a logic "1" state after the receipt of a valid starting sequence and transition to logic "0" when a valid ending sequence is received or an error is detected. This output combined with the Error output will inform the operating system of the end of an error free data transmission.

ERROR

The Error output transitions to a logic "1" when an error is detected. Detection of an error causes the Receiver Active and the Data Available outputs to transition to a logic "0". The Error output returns to a logic "0" after the error register has been read or when the next starting sequence is detected.

REGISTER READ

The Register Read input when driven to the logic "0" state signals the receiver that data in the holding register is being read by the external operating system. The data present in the holding register will continue to remain valid until the Register Read input returns to the logic "1" condition. At this time, if an additional byte is present in the input shift register it will be transferred to the holding register, otherwise the data will remain valid in the holding register. The Data Available output will be in the logic "0" state for a short interval while a new byte is transferred to the holding register after a register read.

DATA AVAILABLE

This output indicates the existence of a data byte within the output holding register. It may also indicate the presence of a data byte in both the holding register and the input shift register. This output will transition to the logic "1" state as soon as data is available and return to the logic "0" state after each data byte has been read. However, even after the last data byte has been read and the Data Available output has assumed the logic "0" state, the last data byte read from the holding register will remain until new data has been received.

Detailed Functional Pin Description (Continued)

OUTPUT CONTROL

The Output Control input determines the type of information appearing at the data outputs. In the logic "1" state data will appear, in the logic "0" state error codes are present.

Truth Table

Output Control	Data Outputs
Logic "0"	Error Codes
Logic "1"	Data

OUTPUT ENABLE

The Output Enable input controls the state of the TRI-STATE Data outputs.

Truth Table

Output Enable	TRI-STATE Data Outputs
Logic "0"	Disabled
Logic "1"	Active

DATA OUTPUTS

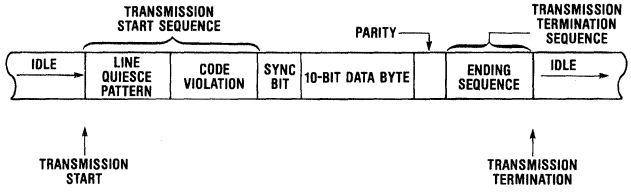
The DP8341 has a ten (10) bit TRI-STATE data bus. Seven bits are multiplexed with error bits. The error bits are defined in the table below. The Output Control input is the multiplexer control for the Data/Error bits.

Error Code Definition

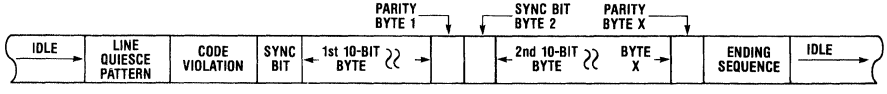
Data Bit	Error Type
DO2	Data Overflow (Byte not removed from holding register when it and the input shift register are both full and new data is received)
DO3	Parity Error (Odd parity detected)
DO4	Transmit Check conditions (existence of errors on any or all of the following data bits: DO3, DO5, and DO6
DO5	An invalid ending sequence
DO6	Loss of mid-bit transition detected at other than normal ending sequence time
DO7	New starting sequence detected before data byte in holding register has been read
DO8	Receiver disabled during receiver active mode

Message Format

Single Byte Transmission



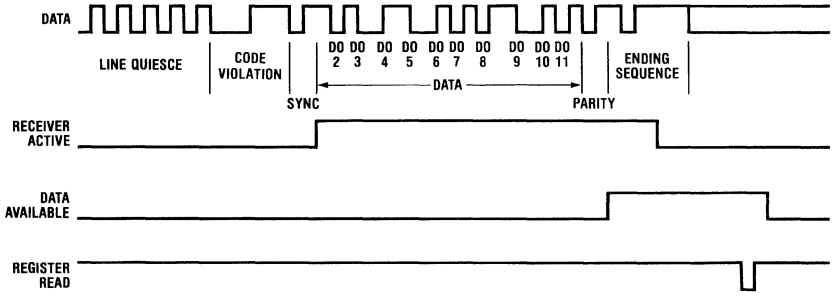
Multi-Byte Transmission



TL/F/5238–4

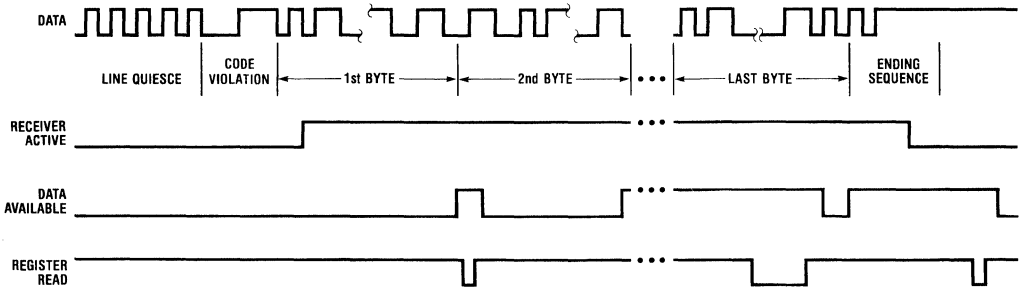
FIGURE 3. IBM 3270 Message Format

Message Format (Continued)



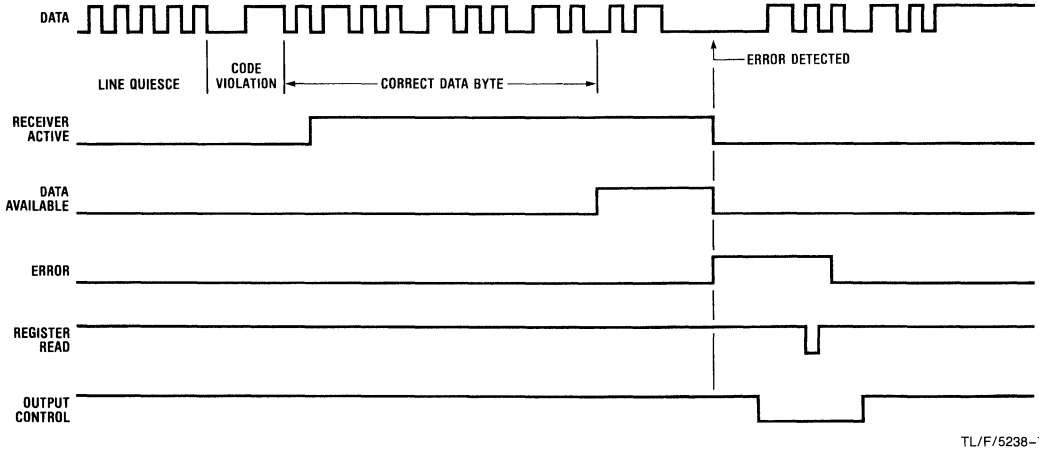
TL/F/5238–5

FIGURE 4a. Single Byte Message



TL/F/5238–6

FIGURE 4b. Multi-Byte Message



TL/F/5238–7

FIGURE 5. Message with Error

5

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage, V_{CC}	7V
Input Voltage	+5.5V
Output Voltage	5.25V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

Maximum Power Dissipation* at 25°C

Cavity Package	2040 mW
Dual-In-Line Package	2237 mW
Plastic Chip Carrier	1690 mW

*Derate cavity package 13.6 mW/°C above 25°C; derate PCC package 13.5 mW/°C above 25°C; derate Dual-In-Line package 17.9 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, (T_A)	0	+70	°C

Electrical Characteristics (Notes 2, 3, and 5)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Input High Level		2.0			V
V_{IL}	Input Low Level				0.8	V
$V_{IH}-V_{IL}$	Data Input Hysteresis (TTL, Pin 4)		2.0	0.4		V
V_{CLAMP}	Input Clamp Voltage	$I_{IN} = -12$ mA		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current	$V_{CC} = 5.25V, V_{IN} = 5.25V$		2	40	μA
I_{IL}	Logic "0" Input Current	$V_{CC} = 5.25V, V_{IN} = 0.5V$		−20	−250	μA
V_{OH}	Logic "1" Output Voltage	$I_{OH} = -100$ μA	3.2	3.9		V
		$I_{OH} = -1$ mA	2.5	3.2		V
V_{OL}	Logic "0" Output Voltage	$I_{OL} = 5$ mA		0.35	0.5	V
I_{OS}	Output Short Circuit Current	$V_{CC} = 5V, V_{OUT} = 0V$ (Note 4)	−10	−20	−100	mA
I_{OZ}	TRI-STATE Output Current	$V_{CC} = 5.25V, V_O = 2.5V$	−40	1	+40	μA
		$V_{CC} = 5.25V, V_O = 0.5V$	−40	−5	+40	μA
A_{HYS}	Amplifier Input Hysteresis		5	20	30	mV
I_{CC}	Power Supply Current	$V_{CC} = 5.25V$		160	250	mA

Timing Characteristics (Notes 2, 6, 7, and 8)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{D1}	Output Data to Data Available Positive Edge		5	20	40	ns
T_{D2}	Register Read Positive Edge to Data Available Negative Edge		10	25	45	ns
T_{D3}	Error Positive Edge to Data Available Negative Edge		10	30	50	ns
T_{D4}	Error Positive Edge to Receiver Active Negative Edge		5	20	40	ns
T_{D5}	Register Read Positive Edge to Error Negative Edge		20	45	75	ns
T_{D6}	Delay from Output Control to Error Bits from Data Bits		5	20	50	ns
T_{D7}	Delay from Output Control to Data Bits from Error Bits		5	20	50	ns
T_{D8}	First Sync Bit Positive Edge to Receiver Active Positive Edge			$3.5 \times T$ +70		ns

Timing Characteristics (Notes 2, 6, 7, and 8) (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{D9}	Receiver Active Positive Edge to First Data Available Positive Edge			92 × T		ns
T_{D10}	Negative Edge of Ending Sequence to Receiver Active Negative Edge			11.5 × T + 50		ns
t_{D11}	Data Control Set-Up Multiplexer Time Prior to Receiving Data through Selected Input		40	30		ns
T_{PW1}	Register Read (Data) Pulse Width		40	30		ns
T_{PW2}	Register Read (Error) Pulse Width		40	30		ns
T_{PW3}	Data Available Logic "0" State between Data Bytes		25	45		ns
T_S	Output Control Set-Up Time Prior to Register Read Negative Edge		0	−5		ns
T_H	Output Control Hold Time After the Register Read Positive Edge		0	−5		ns
T_{ZE}	Delay from Output Enable to Logic "1" or Logic "0" from High Impedance State	Load Circuit 2		25	35	ns
T_{EZ}	Delay from Output Enable to High Impedance State from Logic "1" or Logic "0"	Load Circuit 2		25	35	ns
F_{MAX}	Data Bit Frequency (Clock Input must be 8 × the Data Bit Frequency)	(Note 9)	DC		3.5	MBits/s

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute value basis.

Note 4: Only one output at a time should be shorted.

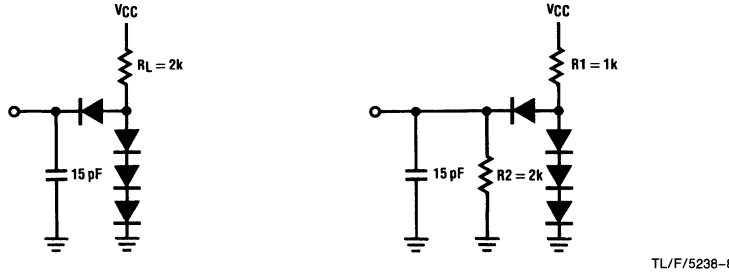
Note 5: Input characteristics do not apply to amplifier inputs (pins 2 and 3).

Note 6: Unless otherwise specified, all AC measurements are referenced to the 1.5V level of the input to the 1.5V level of the output and load circuit 1 is used.

Note 7: AC tests are done with input pulses supplied by generators having the following characteristics: Z_{OUT} = 50Ω and $T_r \leq 5$ ns, $T_f \leq 5$ ns.

Note 8: T = 1/(clock input frequency), units for "T" should be ns.

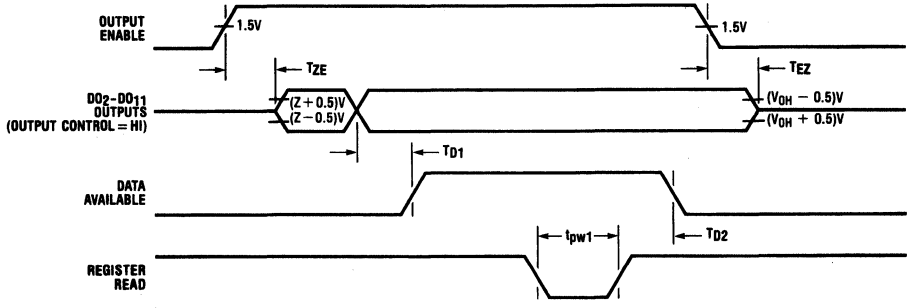
Note 9: 28 MHz clock frequency corresponds to 3.75% jitter when referenced to *Figure 10.*



Load Circuit 1 **Load Circuit 2**

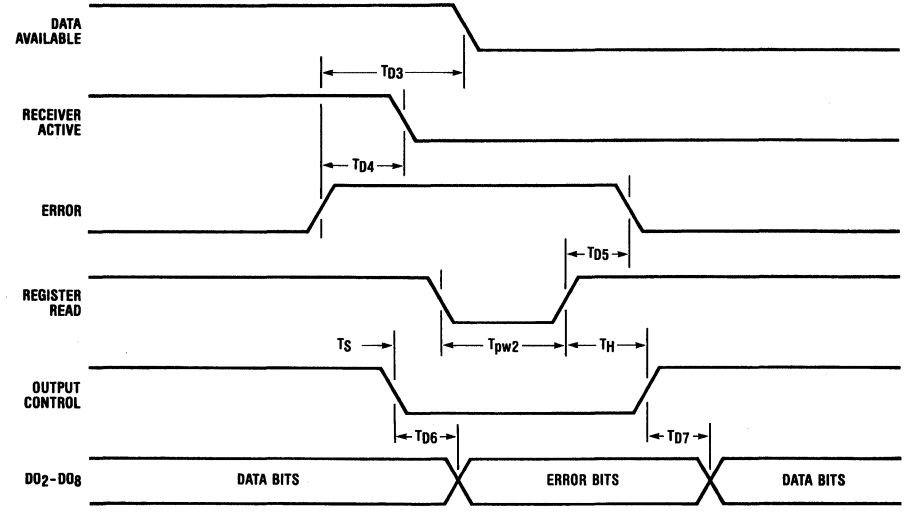
FIGURE 6. Test Load Circuits

Timing Waveforms



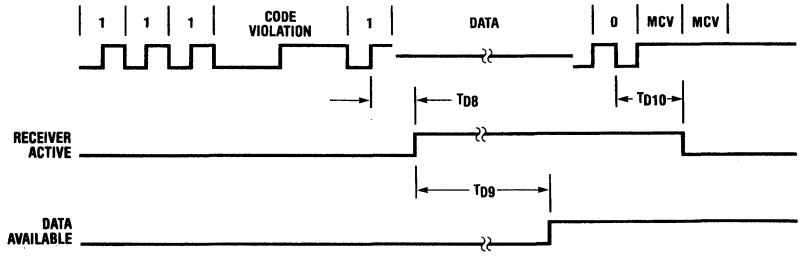
TL/F/5238–9

FIGURE 7. Data Sequence Timing



TL/F/5238–10

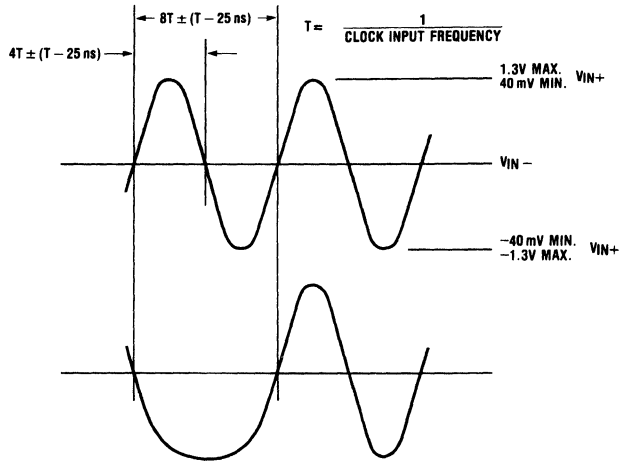
FIGURE 8. Error Sequence Timing



TL/F/5238–11

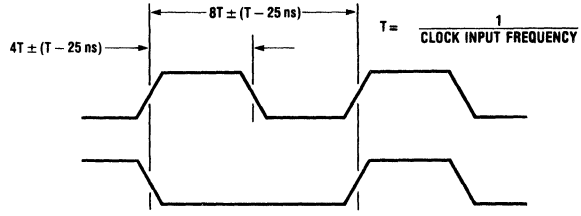
FIGURE 9. Message Timing

Timing Waveforms (Continued)



TL/F/5238–12

FIGURE 10. Data Waveform Constraints: Amplifier Inputs



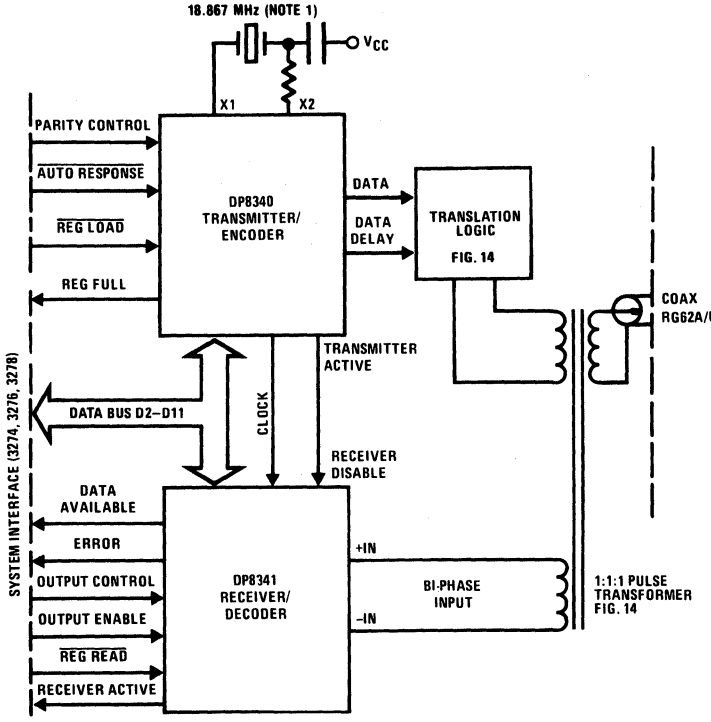
TL/F/5238–13

Note: $|T_r - T_f| \leq 10$ ns

FIGURE 11. Data Waveform Constraints: Data Input (TTL)

5

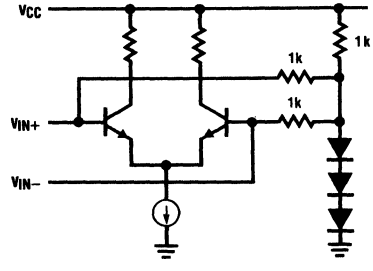
Typical Applications



TL/F/5238–14

Note 3: Crystal manufacturers: Midland Ross Corp.
NEL Unit Part No. NE18A (C2560N) @ 18.867 MHz
The Viking Group Part No. VXB-46NS @ 18,867 MHz. Located in San Jose, CA.

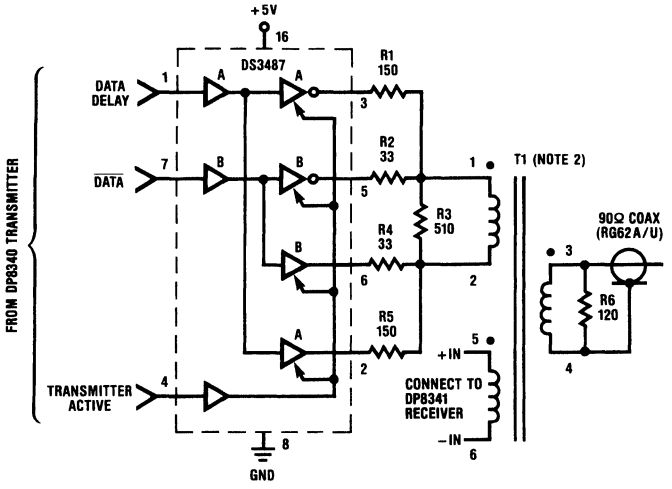
FIGURE 12. Typical Application for IBM 3270 Interface



TL/F/5238–15

FIGURE 13. Equivalent Circuit for DP8341/NS32441 Input Amplifier

Typical Applications (Continued)

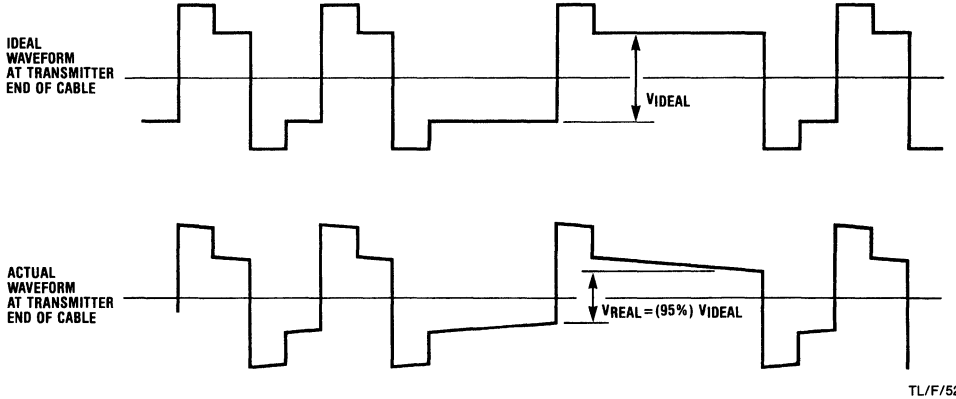


TL/F/5238–16

Note 1: Resistance values are in Ω, ±5%, ¼W

Note 2: T1 is a 1:1:1 pulse transformer, L_{MIN} = 500 μH for 18 MHz system clock
Pulse Engineering Part No. 5762/Surface Mount, 5762M/PE-85762
Valor Electronics Part No. CT1501
Technitrol Part No. 11LHA or equivalent transformers

FIGURE 14. Translation Logic



TL/F/5238–17

*To maintain loss at 95% of ideal signal, select transformer inductance such that:

$$L_{(MIN)} = \frac{10{,}000}{f_{CLK}} \qquad f_{CLK} = \text{System Clock Frequency (e.g., 18.87 MHz)}$$

EXAMPLE:

$$L = \frac{10{,}000}{18.87 \times 10^6} \rightarrow L(MIN) = 530\mu H$$

Note 1: Less inductance will cause greater amplitude attenuation

Note 2: Greater inductance may decrease signal rise time slightly and increase ringing, but these effects are generally negligible.

FIGURE 15. Transformer Selection

5

National Semiconductor Corporation

DP8342/NS32442 High-Speed Serial Transmitter/Encoder

General Description

The DP8342/NS32442 generates a complete encoding of parallel data for high speed serial transmission. It generates a five bit starting sequence, three bit code violation, followed by a syn bit and eight bit per byte of data plus a parity bit. A three-bit ending code signals the termination of the transmission. The DP8342/NS32442 adapts to generalized high speed serial data transmission as well as the coax lines at a maximum data rate of 3.5 MHz.

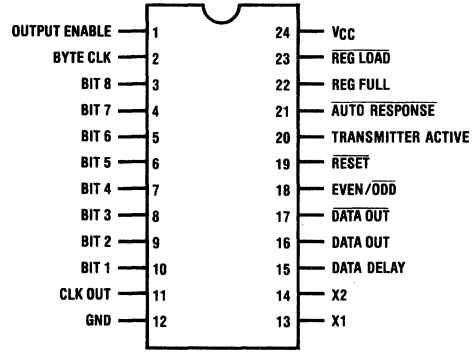
The DP8342/NS32442 and its complementary chip, the DP8343 (receiver/decoder) have been designed to provide maximum flexibility in system designs. The separation of the transmitter receiver functions provides convenient addition of more receivers at one end of a bi-phase line without the need of unused transmitters. This is specifically advantageous in control units where typical bi-phase data is multiplexed over many bi-phase lines and the number of receivers generally exceeds the number of transmitters.

Features

- Eight bits per data byte transmission
- Single-byte or multi-byte transmission
- Internal parity generation (even or odd)
- Internal crystal controlled oscillator used for the generation of all required chip timing frequencies
- Clock output directly drives receiver (DP8343) clock input
- Input data hold register
- Automatic clear status response feature
- Line drivers at data outputs provide easy interface to bi-phase coax line or general transmission media
- <2 ns driver output skew
- Bipolar technology provides TTL input/output compatibility
- Data outputs power up/down glitch free
- Internal power up clear and reset
- Single +5V power supply

Connection Diagram

Dual-In-Line Package

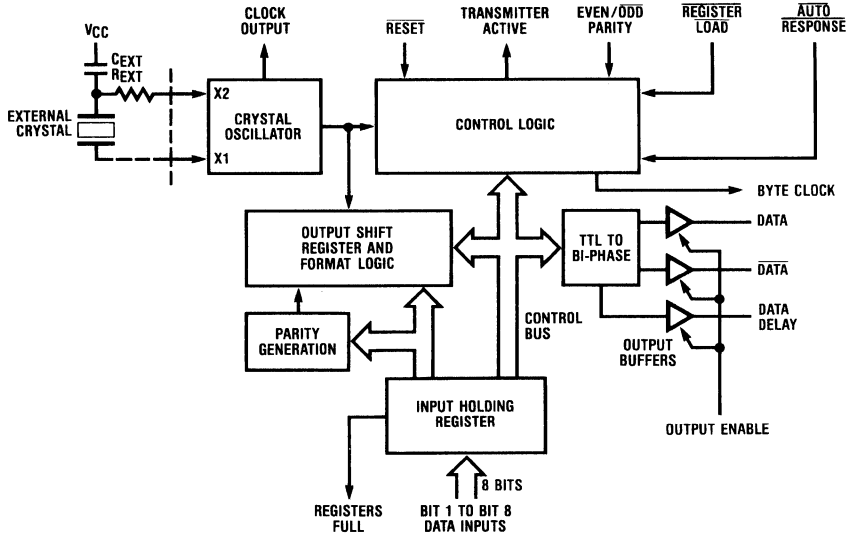


TL/F/5236–1

FIGURE 1

Order Number DP8342J, NS32442J
or DP8342J, NS32442N
See NS Package Number J24A or N24A

Block Diagram



TL/F/5236–2

FIGURE 2

Functional Description

Figure 2 is a block diagram of the DP8342/NS32442 Bi-Phase Transmitter/Encoder. The transmitter/encoder contains a crystal oscillator whose input is a crystal with a frequency eight (8) times the data rate. A Clock Output is provided to drive the DP8342/NS32442 receiver/decoder Clock Input and other system components at the oscillator frequency. Additionally, the oscillator drives the control logic and output shift register/format logic blocks.

Data is parallel loaded from the system data bus to the transmitter/encoder's input holding register. This data is in turn loaded by the transmitter/encoder to its output shift register if this register was empty at the time of the load. During this load, message formatting and parity are generated. The formatted message is then shifted out at the bit rate frequency to the TTL to Bi-Phase block which generates the proper data bit formatting. The data outputs, DATA, $\overline{\text{DATA}}$, and DATA DELAY provide for flexible interface to the transmission medium with little or no external components.

The control Logic block interfaces to all blocks to insure proper chip operation and sequencing. It controls the type of parity generation through the Even/$\overline{\text{Odd}}$ Parity input. An additional feature provided by the transmitter/encoder is the Reset and Output-TRI-STATE® capability. Another feature of the DP8342/NS32442 is the Byte Clock output which keeps track of the number of bytes transferred.

The transmitter/encoder is also capable of internal TT/AR (Transmission Turnaround/Auto Response). When the Auto-Response ($\overline{\text{AR}}$) input is forced to the logic "0" state, the transmitter/encoder responds with clean status (all zeros on data bits).

Operation of the transmitter/encoder is automatic. After the first data byte is loaded, the Transmitter Active output is set and the transmitter/encoder immediately formats the input data and serially shifts it out its data outputs. If the message is a mutli-byte message, the internal format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding format logic will modify the message data format for multibyte as long as the next byte is loaded to the input holding register before the last data bit of the previous data byte is transferred out of the internal output shift register. After all data is shifted out of the transmitter/encoder the Transmitter Active output will return to the inactive state.

Detailed Pin/Functional Description

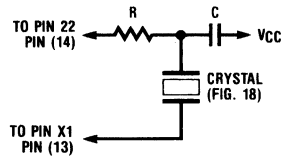
CRYSTAL INPUTS X1 AND X2

The oscillator is controlled by an external, parallel resonant crystal connected between the X1 and X2 pins. Normally, a fundamental mode crystal is used to determine the operating frequency of the oscillator; however, over-tone mode crystals may be used.

CRYSTAL SPECIFICATIONS (PARALLEL RESONANT)

Type	<20 MHz AT-cut or > 20 MHz BT-cut
Tolerance	0.005% at 25°C
Stability	0.01% from 0°C to +70°C
Resonance	Fundamental (Parallel)
Maximum Series Resistance	Dependent on Frequency (For 20 MHz, 50Ω)
Load Capacitance	15 pF

Connection Diagram



TL/F/5236–3

Freq	R	C
10 MHz–20 MHz	500Ω	30 pF
>20 MHz	120Ω	15 pF

If the DP8342/NS32442 transmitter is clocked by a system clock (crystal oscillator not used), pin 13 (X1 input) should be clock directly using a Schottky series (74S) circuit. Pin 14 (X2 input) may be left open. The clocking frequency must be set at eight times the data bit rate. Maximum input frequency is 28 MHz.

CLOCK OUTPUT

The Clock Output is a buffered output derived directly from the crystal oscillator block and clocks at the oscillator frequency. It is designed to directly drive the DP8343 receiver/decoder Clock Input as well as other system components.

REGISTERS FULL

This output is used as a flag by the external operating system. A logic "1" (active state) on this output indicates that both the internal output shift register and the input holding register contain active data. No additional data should be loaded until this output returns to the logic "0" state (inactive state).

TRANSMITTER ACTIVE

This output will be in the logic "1" state while the transmitter/encoder is about to transmit or is in the process of transmitting data. Otherwise, it will assume the logic "0" state indicating no data presently in either the input holding or output shift registers.

$\overline{\text{REGISTER LOAD}}$

The Register Load input is used to load data from the Data Inputs to the input holding register. The loading function is level sensitive, the data present during the logic "0" state of this input is loaded, and the input data must be valid before the logic "0" to logic "1" transition. It is after this transition that the transmitter/encoder begins formatting of data for serial transmission.

$\overline{\text{AUTO RESPONSE}}$ (TT/AR)

This input provides for automatic clear data transmission (all bits in logic "0") without the need of loading all zero's. When a logic "0" is forced on this input the transmitter/encoder immediately responds with transmission of "clean status". When this input is in the logic "1" state the transmitter/encoder transmits data entered on the Data Inputs.

EVEN/$\overline{\text{ODD}}$ PARITY

This input sets the internal logic of the DP8342/NS32442 transmitter/encoder to generate either even or odd parity for the data byte in the bit 10 position. When this pin is in the logic "0" state odd parity is generated. In the logic "1" state even parity is generated. This feature is useful when the control unit is performing a loop back check and at the same time the controller wishes to verify proper data transmission with its receiver/decoder.

SERIAL OUTPUTS—DATA, $\overline{\text{DATA}}$, AND DATA DELAY

These three output pins provide for convenient application of data to the Bi-Phase transmission line. The Data outputs are a direct bit representation of the Bi-Phase data while the Data Delay output provides the necessary increment to clearly define the four (4) DC levels of the pulse. The DATA and $\overline{\text{DATA}}$ outputs add flexibility to the DP8342/NS32442 transmitter/encoder for use in high speed differential line driving applications. The typical DATA to $\overline{\text{DATA}}$ skew is 2 ns.

$\overline{\text{RESET}}$

When a logic "0" is forced on this input, all outputs except Clock Output are latched low.

OUTPUT ENABLE

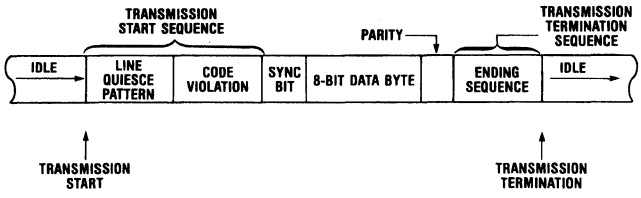
When a logic "0" is forced on this input the three serial data outputs are in the high impedence state.

BYTE CLOCK

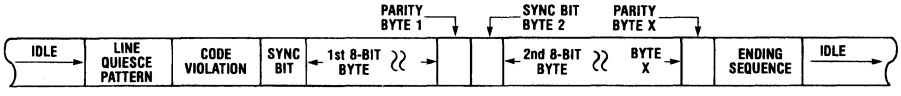
This pin registers a pulse at the end of each byte transmission. The number of pulses registered corresponds to the number of bytes transmitted.

Message Format

Single Byte Transmission



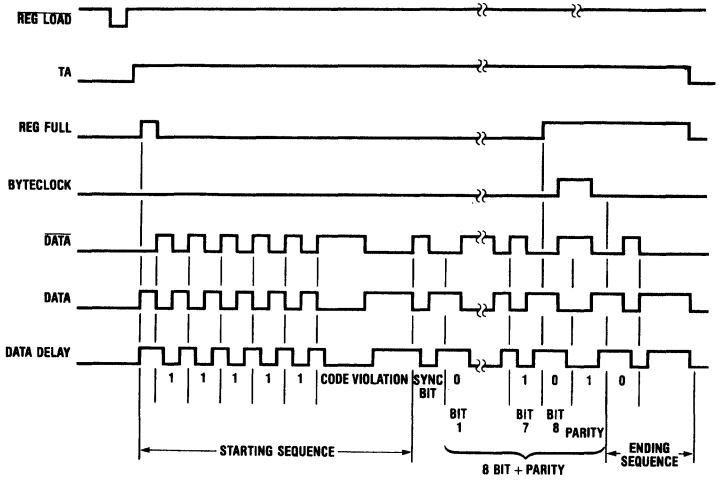
Multi-Byte Transmission



TL/F/5236–4

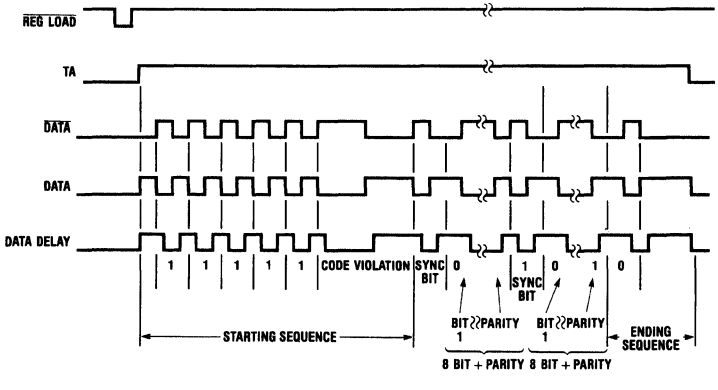
FIGURE 3

Functional Timing Waveforms



TL/F/5236–5

FIGURE 4. Overall Timing Waveforms for Single Byte



TL/F/5236–6

FIGURE 5. Overall Timing Waveforms for Multi-Byte

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage, V_{CC}	7V
Input Voltage	5.5V
Output Voltage	5.25V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Maximum Power Dissipation* at 25°C

Cavity Package	2237 mW
Dual-In-Line package	2500 mW

*Derate cavity package 14.9 mW/°C above 25°C; derate dual in line package 20 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, T_A	0	+70	°C

Electrical Characteristics (Notes 2 and 3)

Symbol	Parameter		Conditions	Min	Typ	Max	Units
V_{IH}	Logic "1" Input Voltage (All Inputs Except X1 and X2)		V_{CC} = 5V	2.0			V
V_{IL}	Logic "0" Input Voltage (All Inputs Except X1 and X2)		V_{CC} = 5V			0.8	V
V_{CLAMP}	Input Clamp Voltage (All Inputs Except X1 and X2)		I_{IN} = −12 mA		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current	Register Load Input	V_{CC} = 5.25V, V_{IN} = 5.25V		0.3	120	μA
		All Others Except X1 and X2			0.1	40	μA
I_{IL}	Logic "0" Input Current	Register Load Input	V_{CC} = 5.25V, V_{IN} = 0.5V		−15	−300	μA
		All Inputs Except X1 and X2			−5	−100	μA
V_{OH1}	Logic "1" All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY		I_{OH} = −100 μA, V_{CC} = 4.75V	3.2	3.9		V
			I_{OH} = −1 mA	2.5	3.4		V
V_{OH2}	Logic "1" for CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY Outputs		V_{CC} = 4.75V, I_{OH} = −10 mA	2.6	3.0		V
V_{OL1}	Logic "0" All Outputs Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY		V_{CC} = 4.75V, I_{OL} = 5 mA		0.35	0.5	V
V_{OL2}	Logic "0" for CLK OUT, DATA $\overline{DATA}$, and DATA DELAY Outputs		V_{CC} = 4.75V, I_{OL} = 20 mA		0.4	0.6	V
I_{OS1}	Output Short Circuit Current for All Except CLK OUT, DATA, $\overline{DATA}$, and DATA DELAY Outputs		(Note 5) V_{OUT} = 0V	−10	−30	−100	mA
I_{OS2}	Output Short Circuit Current DATA, $\overline{DATA}$, and DATA DELAY Outputs		(Note 5) V_{OUT} = 0V	−50	−140	−350	mA
I_{OS3}	Output Short Circuit Current for CLK OUT		(Note 5) V_{OUT} = 0V	−30	−90	−200	mA
I_{CC}	Power Supply Current		V_{CC} = 5.25V		170	250	mA

Timing Characteristics V_{CC} = 5V ±5%, T_A = 0°C to 70°C, Oscillator Frequency = 28 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd1}	$\overline{REG\ LOAD}$ to Transmitter Active (TA) Positive Edge	Load Circuit 1 *Figure 6*		60	90	ns
t_{pd2}	$\overline{REG\ LOAD}$ to Register Full; Positive Edge	Load Circuit 1 *Figure 6*		45	75	ns
t_{pd3}	TA to Register Full; Negative Edge	Load Circuit 1 *Figure 6*		40	70	ns
t_{pd4}	Positive Edge of $\overline{REG\ LOAD}$ to Positive Edge of DATA	Load Circuit 2 *Figure 9*		50	80	ns
t_{pd5}	$\overline{REG\ LOAD}$ to $\overline{DATA}$; Positive Edge	Load Circuit 2 *Figure 9*		280	380	ns
t_{pd6}	$\overline{REG\ LOAD}$ to DATA DELAY; Positive Edge	Load Circuit 2 *Figure 9*		150	240	ns

Timing Characteristics (Continued)

V_{CC} = 5V ±5%, T_A = 0°C to 70°C, Oscillator Frequency = 28 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd7}	Positive Edge of $\overline{DATA}$ to Negative Edge of DATA DELAY	Load Circuit 2 *Figure 9*		70	85	ns
t_{pd8}	Positive Edge of DATA DELAY to Negative Edge of $\overline{DATA}$	Load Circuit 2 *Figure 9*		80	95	ns
t_{pd9}, t_{pd10}	Skew between DATA and $\overline{DATA}$	Load Circuit 2 *Figure 9*		2	6	ns
t_{pd11}	Negative Edge of $\overline{\text{Auto Response}}$ ($\overline{AR}$) to Positive Edge of TA	Load Circuit 1 *Figure 10*		70	100	ns
t_{pd12}	Maximum Time Delay to Load Second Byte after Positive Edge of REG FULL	Load Circuit 1 *Figure 8*, (Note 7)			4 × T − 50	ns
t_{pd13}	X1 to CLK OUT; Positive Edge	Load Circuit 2 *Figure 11*		21	30	ns
t_{pd14}	X1 to CLK OUT; Negative Edge	Load Circuit 2 *Figure 11*		23	33	ns
t_{pd15}	Negative Edge of $\overline{AR}$ to Positive Edge of REG FULL	Load Circuit 1 *Figure 10*		45	75	ns
t_{pd16}	Skew between TA and REG FULL during $\overline{\text{Auto Response}}$	Load Circuit 1 *Figure 10*		50	80	ns
t_{pd17}	$\overline{\text{REG LOAD}}$ to REG FULL; Positive Edge for Second Byte	Load Circuit 1 *Figure 7*		45	75	ns
t_{pd18}	REG FULL to BYTE CLK; Negative Edge	Load Circuit 1 *Figure 7*		60	90	ns
t_{pd19}	REG FULL to BYTE CLK; Positive Edge	Load Circuit 1 *Figure 7*		145	180	ns
t_{ZH}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY outputs: HiZ to High	CL = 50 pF *Figures 16, 17*		25	45	ns
t_{ZL}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY Outputs; HiZ to High	CL = 50 pF *Figures 16, 17*		15	30	ns
t_{HZ}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY Outputs; High to HiZ	CL = 15 pF *Figures 16, 17*		65	100	ns
t_{LZ}	Output Enable to DATA, $\overline{DATA}$, or DATA DELAY Outputs; Low to HiZ	CL = 15 pF *Figures 16, 17*		45	70	ns
t_{pw1}	$\overline{\text{REG LOAD}}$ Pulse Width	*Figure 12*	40			ns
t_{pw2}	First REG FULL Pulse Width (Note 6)	Load Circuit 1 *Figure 7*, (Note 7)		8 × T + 60	8 × T + 100	ns
t_{pw3}	REG FULL Pulse Width Prior to Ending Sequence (Note 6)	Load Circuit 1 *Figure 7*		5 × B		ns
t_{pw4}	Pulse Width for $\overline{\text{Auto Response}}$	*Figure 10*	40			ns
t_{pu5}	Pulse Width for BYTE CLK	Load Circuit 1 *Figure 7*, (Note 7)		8 × T + 30	8 × T + 80	ns
t_s	Data Setup Time prior to $\overline{\text{REG LOAD}}$ Positive Edge; Hold Time = 0 ns	*Figure 12*		15	23	ns
t_{r1}	Rise Time for DATA, $\overline{DATA}$, and DATA DELAY Output Waveform	Load Circuit 2 *Figure 13*		7	13	ns
t_{f1}	Fall Time for DATA, $\overline{DATA}$, and DATA DELAY Output Waveform	Load Circuit 2 *Figure 13*		5	11	ns
t_{r2}	Rise Time for TA and REG FULL	Load Circuit 1 *Figure 14*		20	30	ns
t_{f2}	Fall Time for TA and REG FULL	Load Circuit 1 *Figure 14*		15	25	ns

Timing Characteristics (Continued)

V_{CC} = 5V ±5%, T_A = 0°C to 70°C, Oscillator Frequency = 28 MHz (Notes 2 and 3)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{MAX}	Data Rate Frequency (Clock Input must be 8× this Frequency)		DC		3.5	Mbits/s
C_{IN}	Input Capacitance—Any Input	(Note 4)		5	15	pF

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min/max limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max or min are so classified on absolute basis.

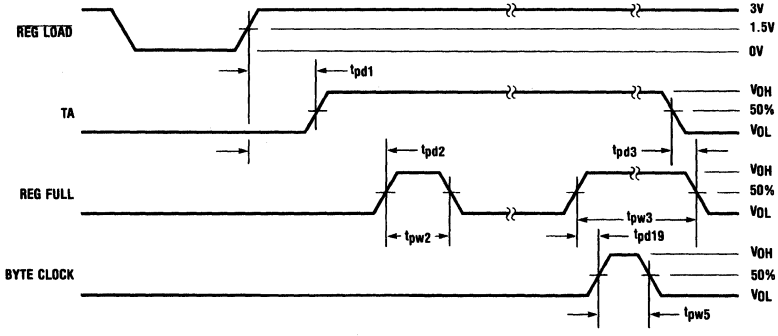
Note 4: Input capacitance is guaranteed by periodic testing. f_{TEST} = 10 kHz at 300 mV, T_A = 25°C.

Note 5: Only one output should be shorted at a time.

Note 6: T = 1/(Oscillator Frequency). Unit for T should be in ns. B = 8T.

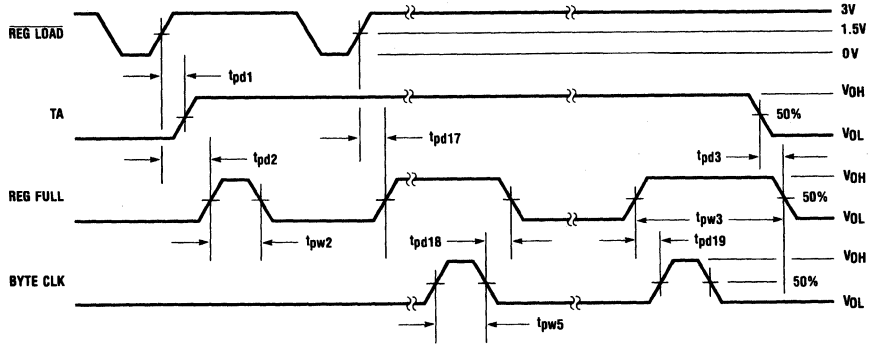
Note 7: Oscillator Frequency Dependent.

Timing Waveforms (Continued)



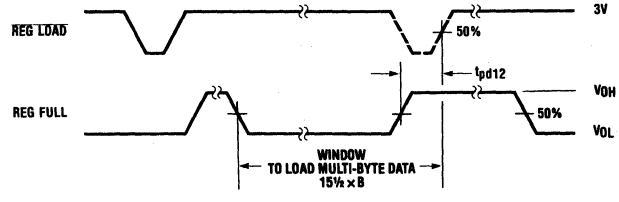
TL/F/5236–7

FIGURE 6. Single Byte Transfer



TL/F/5236–8

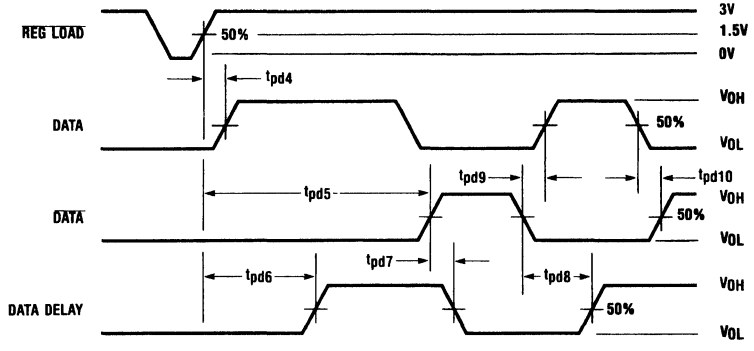
FIGURE 7. Two-Byte Transfer



TL/F/5236–9

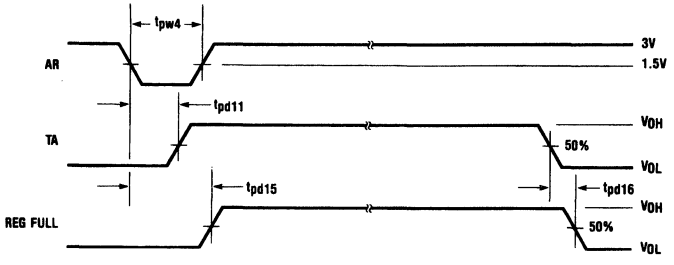
FIGURE 8. Maximum Window to Load Multi-Byte Data

Functional Timing Waveforms (Continued)



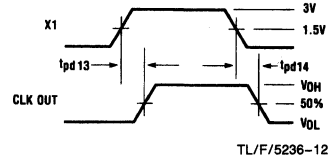
TL/F/5236–10

FIGURE 9. Three Serial Outputs



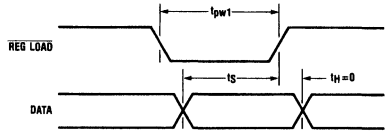
TL/F/5236–11

FIGURE 10. $\overline{\text{Auto-Response}}$



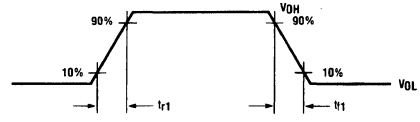
TL/F/5236–12

FIGURE 11. Clock Pulse



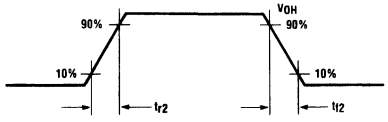
TL/F/5236–13

FIGURE 12. $\overline{\text{REG LOAD}}$



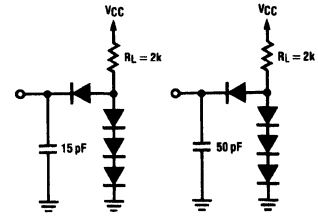
TL/F/5236–14

FIGURE 13. Output Waveform for DATA, $\overline{\text{DATA}}$, DATA DELAY (Load Circuit 2)



TL/F/5236–15

FIGURE 14. Rise and Fall Time Measurement for TA and REG FULL

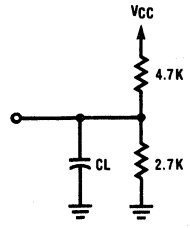


TL/F/5236–16

Load Circuit 1 **Load Circuit 2**

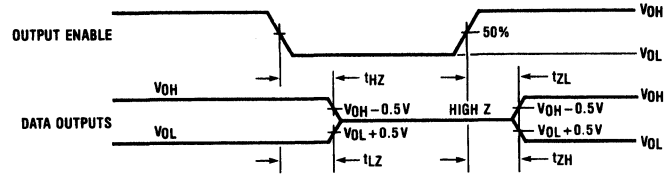
FIGURE 15. Test Load Circuits

Timing Waveforms (Continued)



TL/F/5236–17

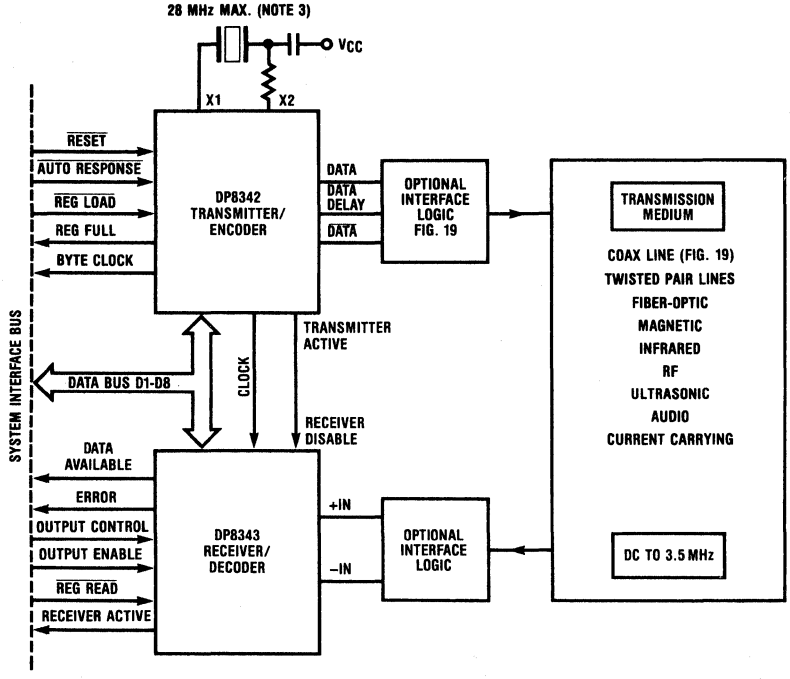
FIGURE 16. Load Circuit for Output TRI-STATE Test



TL/F/5236–18

FIGURE 17. TRI-STATE Test

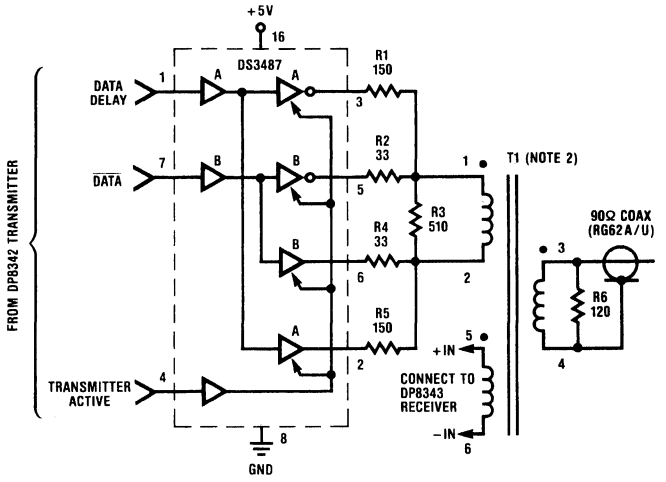
Typical Applications



TL/F/5236–19

FIGURE 18

Typical Applications (Continued)



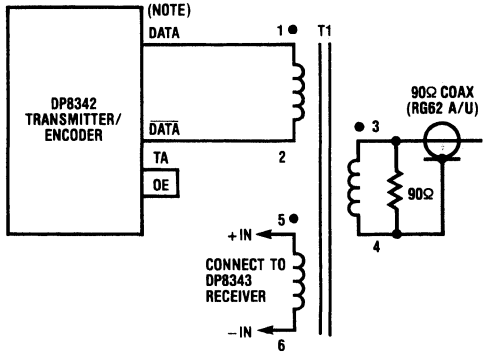
TL/F/5236–20

Note 1: Resistance values are in Ω, ±5%, ¼W.

Note 2: T1 is a 1:1:1 pulse transformer, L = 500 μH for 18 MHz to 28 MHz system clock. Pulse Engineering Part No. 5762; Technitrol Part No. 11LHA, Valor Electronics Part No. CT1501, or equivalent transformer.

Note 3: Crystal manufacturer Midland Ross Corp. NEL Unit Part No. NE-18A at 28 MHz.

FIGURE 19. Interface Logic for a Coax Transmission Line



TL/F/5236–21

Note: Data rates up to 3.5 Mbits/s at 5000′ still apply.

FIGURE 20. Direct Interface for a Coax Transmission Line (Non-IBM Voltage Levels)

5

National Semiconductor Corporation

DP8343/NS32443 High-Speed Serial Receiver/Decoder

General Description

The DP8343/NS32443 provides complete decoding of data for high speed serial data communications. In specific, the DP8343/NS32443 receiver recognizes Bi-Phase serial data sent from its complementary chip, the DP8342 transmitter, and converts it into 8 bits of parallel data. These devices are easily adapted to generalized high speed serial data transmission systems that operate at bit rates up to 3.5 MHz.

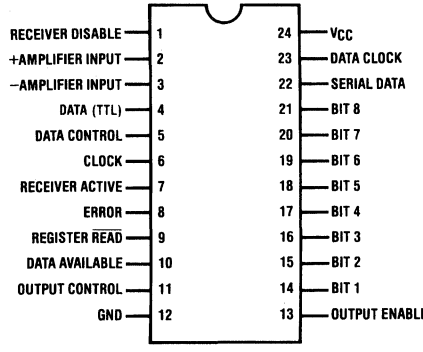
The DP8343/NS32443 receiver and the DP8342 transmitter are designed to provide maximum flexibility in system designs. The separation of transmitter and receiver functions allows addition of more receivers at one end of the Bi-Phase line without the necessity of adding unused transmitters. This is advantageous in control units where the data is typically multiplexed over many lines and the number of receivers generally exceeds the number of transmitters. The separation of transmitter and receiver function provides an additional advantage in flexibility of data bus organization. The data bus outputs of the receiver are TRI-STATE®, thus enabling the bus configuration to be organized as either a common transmit/receive (bi-directional) bus or as separate transmit and receive busses for higher speed.

Features

- DP8343/NS32443 receives 8-bit data bytes
- Separate receiver and transmitter provide maximum system design flexibility
- Even parity detection
- High sensitivity input on receiver easily interfaces to coax line
- Standard TTL data input on receiver provides generalized transmission line interface and also provides hysteresis
- Data holding register
- Multi-byte or single byte transfers
- TRI-STATE receiver date outputs provide flexibility for common or separated transmit/receive data bus operation
- Data transmission error detection on receiver provides for both error detection and error type definition
- Bipolar technology provides TTL input/output compatibility with excellent drive characteristics
- Single +5V power supply operation

Connection Diagram

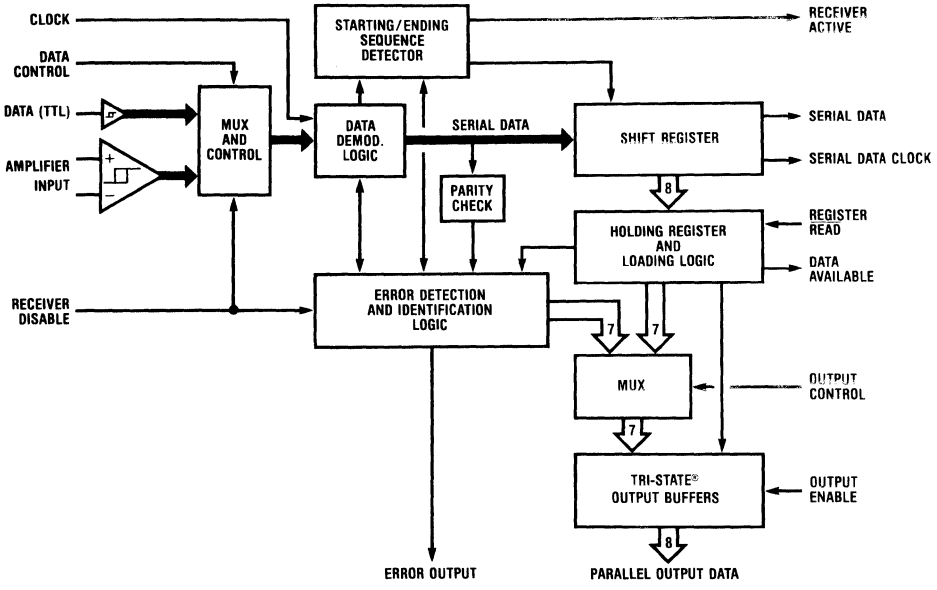
Dual-In-Line Package



TL/F/5237–1

FIGURE 1
Order Number DP8343/NS32443J
or DP8343/NS32443N
See NS Package Number J24A or N24A

Block Diagram



TL/F/5237–2

FIGURE 2. DP8343/NS32443 Bi-Phase Receiver

Functional Description

Figure 2 is a block diagram of the DP8343/NS32443 receiver. This chip is essentially a serial in/parallel out shift register. However, the serial input data must conform to a very specific format (see *Figures 3–6*). The message will not be recognized unless the format of the starting sequence is correct. Deviations from the format in the data, sync bit, parity or ending sequence will cause an error to be detected, terminating the message.

Data enters the receiver through the differential input amplifier or the TTL Data input. The differential amplifier is a high sensitivity input which may be used by connecting it directly to a transformer coupled coax line, or other transmission medium. The TTL Data input provides 400 mV of hysteresis and recognizes TTL logic levels. The data then enters the demodulation block.

The data demodulation block samples the data at eight (8) times the data rate and provides signals for detecting the starting sequence, ending sequence, and errors. Detection of the starting sequence sets the Receiver Active output high and enables the input shift register.

As the eight bits of data are shifted into the shift register, the receiver will verify that even parity is maintained on the data bits and the sync bit. Serial Data and Serial Data Clock, the inputs to the shift register, are provided for use with external error detecting schemes. After one complete data byte is received, the contents of the input shift register is parallel loaded to the holding register, assuming the holding register is empty, and the Data Available output is set. If the holding register is full, this load will be delayed until that register has been read or the start of another data byte is received, in which case a Data Overflow Error will be detected, terminating the message. Data is read from the holding register through the TRI-STATE Output Buffers. The Output Enable input is the TRI-STATE control for these outputs and the Register Read input signals the receiver that the read has been completed.

When the receiver detects an ending sequence the Receiver Active output will be reset to a logic "0" indicating the message has been terminated. A message will also terminate when an error is detected. The Receiver Active output used in conjunction with the Error output allows quick response to the transmitting unit when an error free message has been received.

The Error Detection and Identification block insures that valid data reaches the outputs of the receiver. Detection of an error sets the Error output to a logic "1" and resets the Receiver Active output to a logic "0" terminating the message. The error type may be read from the data bus outputs by setting the Output Control input to logic "0" and enabling the TRI-STATE outputs. The data bit outputs have assigned error definitions (see error code definition table). The Error output will return to a logic "0" when the next starting sequence is received, or when the error is read (Output Control to logic "0" and a Register Read performed).

The Receiver Disable input is used to disable both the amplifier and TTL Data receiver inputs. It will typically be connected directly to the Transmitter Active output of the DP8342 transmitter circuit.

Detailed Functional Pin Description

RECEIVER DISABLE

This input is used to disable the receiver's data inputs. The Receiver Disable input will typically be connected to the Transmitter Active output of the DP8342. However, at the system controller it may be necessary for both the transmitter and receiver to be active at the same time. This variation can be accomplished with the addition of minimal external logic.

Truth Table

Receiver Disable	Data Inputs
Logic "0"	Active
Logic "1"	Disabled

AMPLIFIER INPUTS

The receiver has a differential input amplifier which may be directly connected to the transformer coupled coax line. The amplifier may also be connected to a differential type TTL line. The amplifier has 20 mV of hysteresis.

DATA INPUT

This input can be used either as an alternate data input or as a power-up check input. If the system designer prefers to use his own amplifier, instead of the one provided on the receiver, then this TTL input may be used. Using this pin as an alternate data input allows self-test of the peripheral system without disturbing the transmission line.

DATA CONTROL

This input is the control pin that selects which of the inputs are used for data entry to the receiver.

Truth Table

Data Control	Data Input To
Logic "0"	Data Input
Logic "1"	Amplifier Inputs

Note: This input is also used for testing. When the input voltage is raised to 7.5V the chip resets.

CLOCK INPUT

This input is the internal clock of the receiver. It must be set at eight (8) times the line data bit rate. The crystal-controlled oscillator provided in the DP8342 transmitter also operates at this frequency. The Clock Output of the transmitter is designed to directly drive the receiver's Clock Input. In addition, the receiver is designed to operate correctly to a data bit rate of 3.5 MHz.

RECEIVER ACTIVE

The purpose of this output is to inform the external system when the DP8343/NS32443 is in the process of receiving a message. This output will transition to a logic "1" state after a receipt of a valid starting sequence and transition to logic "0" when a valid ending sequence is received or an error is detected. This output combined with the Error output will inform the operating system of the end of an error free data transmission.

ERROR

The Error output transitions to a logic "1" when an error is detected. Detection of an error causes the Receiver Active and the Data Available outputs to transition to a logic "0". The Error output returns to a logic "0" after the error register has been read or when the next starting sequence is detected.

REGISTER READ

The Register Read input when driven to the logic "0" state signals the receiver that data in the holding register is being read by the external operating system. The data present in the holding register will continue to remain valid until the Register Read input returns to the logic "1" condition. At this time, if an additional byte is present in the input shift register it will be transferred to the holding register, otherwise the data will remain valid in the holding register. The Data Available output will be in the logic "0" state for a short interval while a new byte is transferred to the holding register after a register read.

DATA AVAILABLE

This output indicates the existence of a data byte within the output holding register. It may also indicate the presence of a data byte in both the holding register and the input shift register. This output will transition to the logic "1" state as soon as data is available and return to the logic "0" state after each data byte has been read. However, even after the last data byte has been read and the Data Available output has assumed the logic "0" state, the last data byte read from the holding register will remain until new data has been received.

OUTPUT CONTROL

The Output Control input determines the type of information appearing at the data outputs. In the logic "1" state data will appear, in the logic "0" state error codes are present.

Truth Table

Output Control	Data Outputs
Logic "0"	Error Codes
Logic "1"	Data

OUTPUT ENABLE

The Output Enable input controls the state of the TRI-STATE Data outputs.

Truth Table

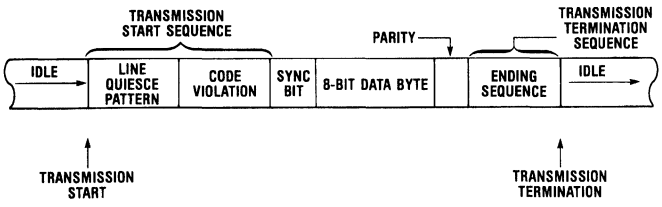
Output Enable	TRI-STATE Data Outputs
Logic "0"	Disabled
Logic "1"	Active

DATA OUTPUTS

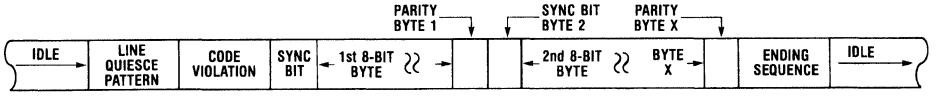
The DP8343/NS32443 has an 8-bit TRI-STATE data bus. Seven bits are multiplexed with error bits. The error bits are defined in the following table. The Output Control input is the multiplexer control for the Data/Error bits.

Message Format

Single Byte Transmission

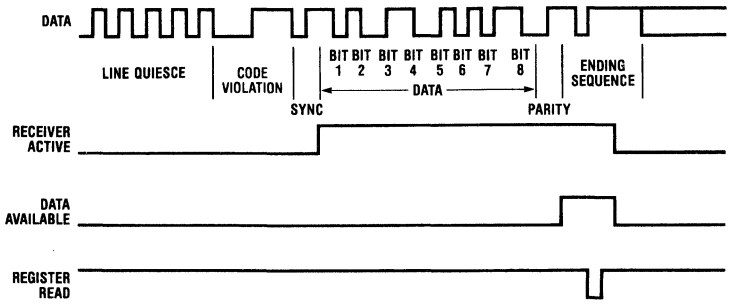


Multi-Byte Transmission



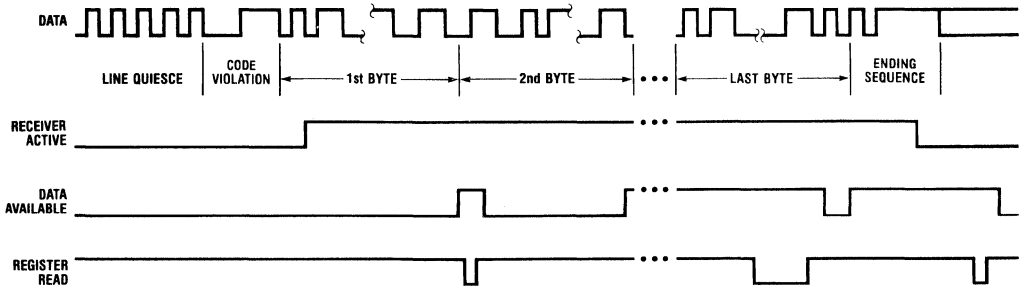
TL/F/5237–3

FIGURE 3



TL/F/5237–4

FIGURE 4a. Single Byte (8-Bit) Message



TL/F/5237–5

FIGURE 4b. Multi-Byte Message

5

Error Code Definition	
Data Bit DP8343	**Error Type**
Bit 1	Data Overflow (Byte not removed from holding register when it and the input shift register are both full and new data is received)
Bit 2	Parity Error (Odd parity detected)
Bit 3	Transmit Check conditions (existence of errors on any or all of the following data bits: Bit 2, Bit 4, and Bit 5)
Bit 4	An invalid ending sequence
Bit 5	Loss of mid-bit transition detected at other than normal ending sequence time
Bit 6	New starting sequence detected before data byte in holding register has been read
Bit 7	Receiver disabled during receiver active mode

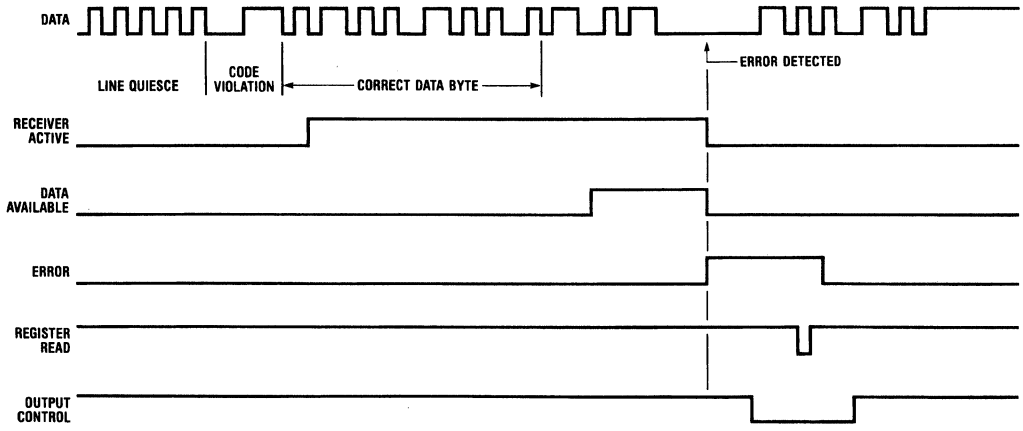
SERIAL DATA

The Serial Data output is the serial data coming into the input shift register.

DATA CLOCK

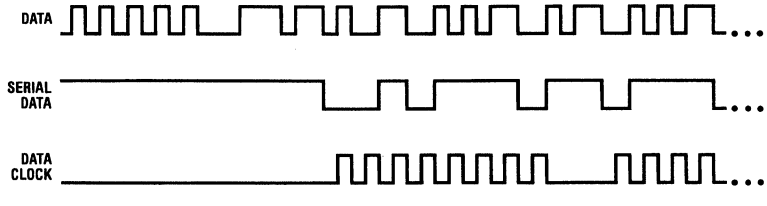
The Data Clock output is the clock to the input shift register.

Message Format (Continued)



TL/F/5237–6

FIGURE 5. Message with Error



TL/F/5237–7

FIGURE 6. Data Clock and Serial Data

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage, (V_{CC})	7.0V
Input Voltage	5.5V
Output Voltage	5.25V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C

Operating Conditions

	Min	Max	Units
Supply Voltage, (V_{CC})	4.75	5.25	V
Ambient Temperature, T_A	0	+70	°C

Electrical Characteristics (Notes 2, 3 and 5)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Input High Level		2.0			V
V_{IL}	Input Low Level				0.8	V
V_{IH}-V_{IL}	Data Input Hysteresis (TTL, Pin 4)		0.2	0.4		V
V_{CLAMP}	Input Clamp Voltage	I_{IN} = −12 mA		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current	V_{CC} = 5.25V, V_{IN} = 5.25V		2	40	μA
I_{IL}	Logic "0" Input Current	V_{CC} = 5.25V, V_{IN} = 0.5V		−20	−250	μA
V_{OH}	Logic "1" Output Voltage	I_{OH} = −100 μA	3.2	3.9		V
		I_{OH} = −1 mA	2.5	3.2		V
V_{OL}	Logic "0" Output Voltage	I_{OL} = 5 mA		0.35	0.5	V
I_{OS}	Output Short Circuit Current	V_{CC} = 5V, V_{OUT} = 0V (Note 4)	−10	−20	−100	mA
I_{OZ}	TRI-STATE Output Current	V_{CC} = 5.25V, V_O = 2.5V	−40	1	+40	μA
		V_{CC} = 5.25V, V_O = 0.5V	−40	−5	+40	μA
A_{HYS}	Amplifier Input Hysteresis		5	20	30	mV
I_{CC}	Power Supply Current	V_{CC} = 5.25V		160	250	mA

Timing Characteristics (Notes 2, 6, 7, and 8)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{D1}	Output Data to Data Available Positive Edge		5	20	40	ns
T_{D2}	Register Read Positive Edge to Data Available Negative Edge		10	25	45	ns
T_{D3}	Error Positive Edge to Data Available Negative Edge		10	30	50	ns
T_{D4}	Error Positive Edge to Receiver Active Negative Edge		5	20	40	ns
T_{D5}	Register Read Positive Edge to Error Negative Edge		20	45	75	ns
T_{D6}	Delay from Output Control to Error Bits from Data Bits		5	20	50	ns
T_{D7}	Delay from Output Control to Data Bits from Error Bits		5	20	50	ns
T_{D8}	First Sync Bit Positive Edge to Receiver Active Positive Edge			3.5 × T +70		ns
T_{D9}	Receiver Active Positive Edge to First Data Available Positive Edge			76 × T		ns
T_{D10}	Negative Edge of Ending Sequence to Receiver Active Negative Edge			11.5 × T +50		ns
T_{D11}	Data Control Set-up Multiplexer Time Prior to Receiving Data through Selected Input		40	30		ns
T_{D12}	Serial Data Set-Up Prior to Data Clock Positive Edge			3 × T		ns

Timing Characteristics (Notes 2, 6, 7, and 8) (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_{PW1}	Register Read (Data) Pulse Width		30	40		ns
T_{PW2}	Register Read (Error) Pulse Width		40	30		ns
T_{PW3}	Data Available Logic "0" State between Data Bytes		25	45		ns
T_S	Output Control Set-Up Time Prior to Register Read Negative Edge		0	−5		ns
T_H	Output Control Hold Time after the Register Read Positive Edge		0	−5		ns
T_{ZE}	Delay from Output Enable to Logic "1" or Logic "0" from High Impedance State	Load Circuit 2		25	35	ns
T_{EZ}	Delay from Output Enable to High Impedance State from Logic "1" or Logic "0"	Load Circuit 2		25	35	ns
F_{MAX}	Data Bit Frequency (Clock Input must be 8 × the Data Bit Frequency)		DC		3.5	MBits/s

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute value basis.

Note 4: Only one output at a time should be shorted.

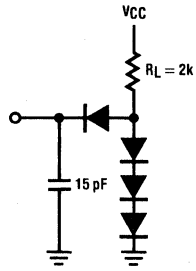
Note 5: Input characteristics do not apply to amplifier inputs (pins 2 & 3).

Note 6: Unless otherwise specified, all AC measurements are referenced to the 1.5V level of the input to the 1.5V level of the output and load circuit 1 is used.

Note 7: AC tests are done with input pulses supplied by generators having the following characteristics: Z_{OUT} = 5Ω, $T_r \leq 5$ ns, and $T_f \leq 5$ ns.

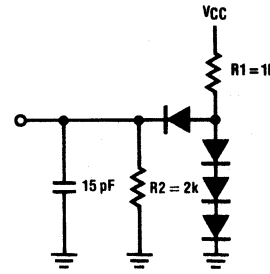
Note 8: T = 1/(clock input frequency), units for "T" should be ns.

Test Load Circuits



TL/F/5237–8

Load Circuit 1

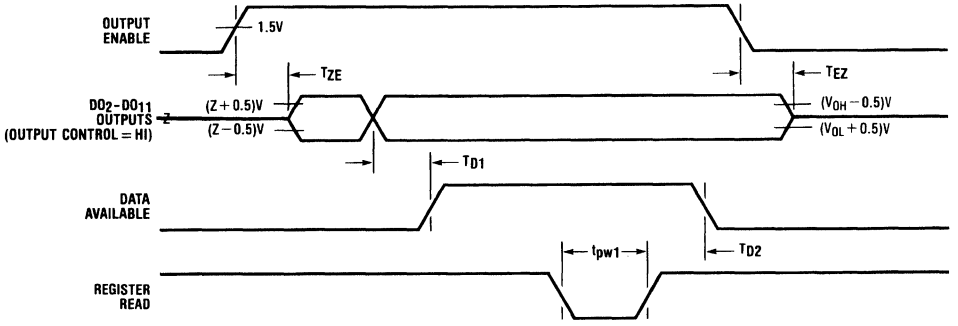


TL/F/5237–9

Load Circuit 2

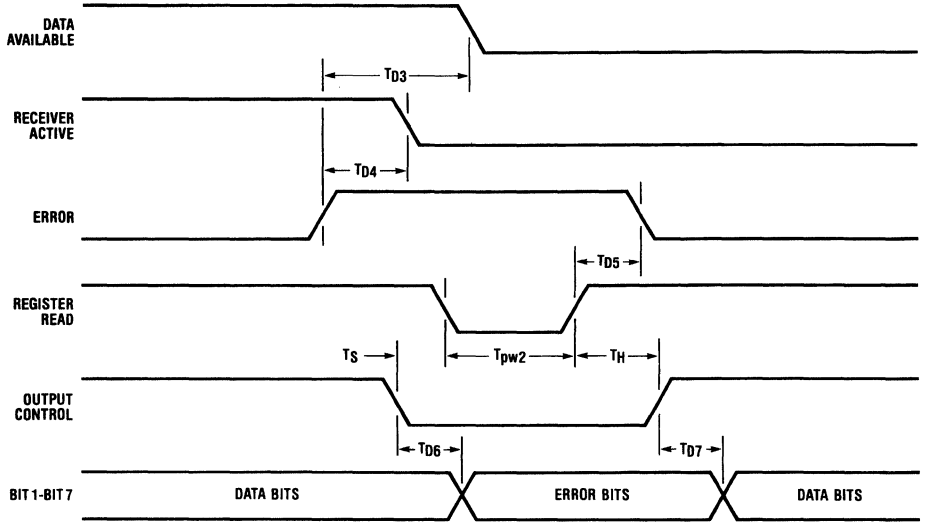
FIGURE 7

Timing Waveforms



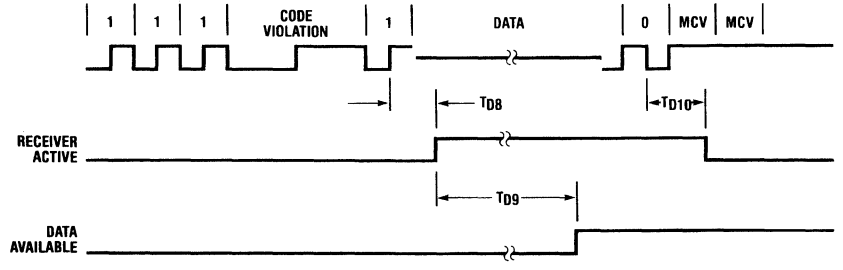
TL/F/5237–10

FIGURE 8. Data Sequence Timing



TL/F/5237–11

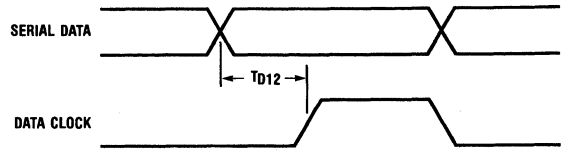
FIGURE 9. Error Sequence Timing



TL/F/5237–12

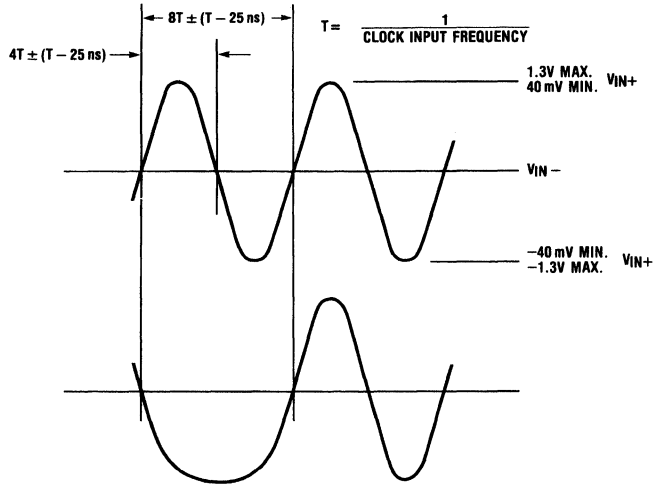
FIGURE 10. Message Timing

Timing Waveforms (Continued)



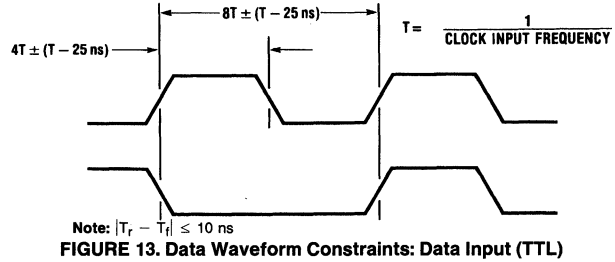
TL/F/5237–13

FIGURE 11. Data Clock and Serial Data Timing



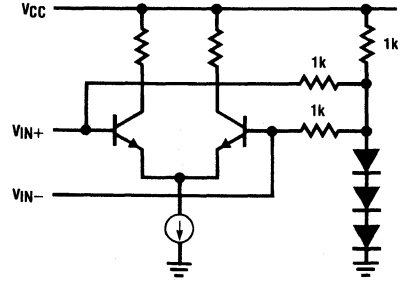
TL/F/5237–14

FIGURE 12. Data Waveform Constraints: Amplifier Inputs



TL/F/5237–15

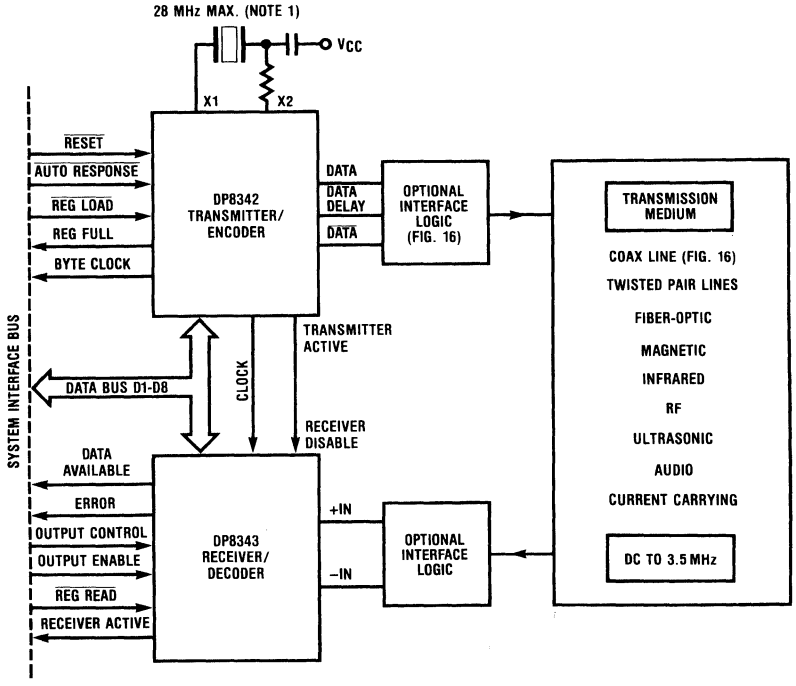
FIGURE 13. Data Waveform Constraints: Data Input (TTL)



TL/F/5237–16

FIGURE 14. Equivalent Circuit for DP8343/NS32443 Input Amplifier

Typical Applications

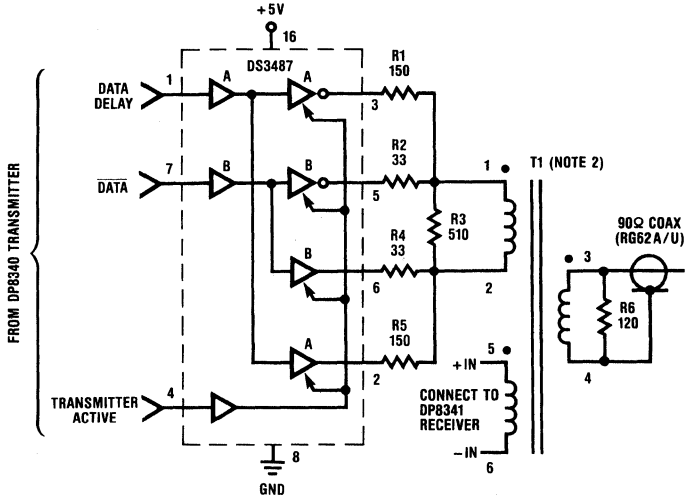


TL/F/5237–17

Note 1: Crystal manufacturer Midland Ross Corp., NEL Unit Part No. NE-18A @ 28 MHz

FIGURE 15

Typical Applications (Continued)

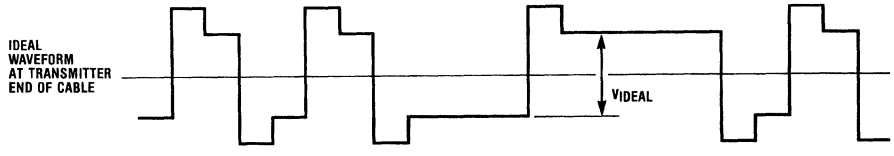


TL/F/5237–18

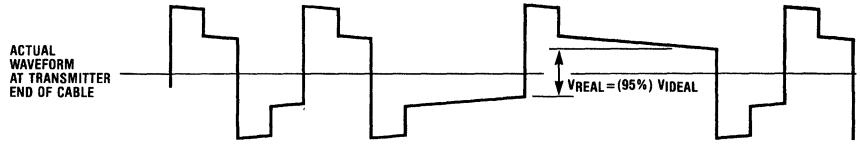
Note 1: Resistance values are in Ω, ±5%, ¼W.

Note 2: T1 is a 1:1:1 pulse transformer, L_{MIN} = 500 μH for 18 MHz system clock.
Pulse Engineering Part No. 5762,
Valor Electronics Part No. CT1501
Technitrol Part No. 11LHA or equivalent transformers.

FIGURE 16. Interface Logic for a Coax Transmission Line



TL/F/5237–19



TL/F/5237–20

*To maintain loss at 95% of ideal signal, select transformer inductance such that:

$$L_{(MIN)} = \frac{10{,}000}{f_{CLK}}$$

f_{CLK} = System Clock Frequency (e.g., 18.87 MHz)

Example:

$$L = \frac{10{,}000}{18.87 \times 10^6} \rightarrow L(MIN) = 530\ \mu H$$

Note 1: Less inductance will cause greater amplitude attenuation.

Note 2: Greater inductance may decrease signal rise time slightly and incease ringing, but these effects are generally negligible.

FIGURE 17. Transformer Selection

Section 6

Disk Control and Interface

Section 6 Contents

DP8461/NS32961/DP8465/NS32965 Data Separator 6-3
DP8451/NS32951/DP8455/NS32955 Data Synchronizer 6-3
DP8462/NS32962 2, 7 Code Data Synchronizer 6-29
DP8463B/NS32963 2, 7 ENDEC 6-49
DP8464B/NS32964 Disk Pulse Detector 6-58
DP8466/NS32966 Disk Data Controller 6-81
DP8470/NS32970 Floppy Disk Support Chip Data Separator and Write Precompensation 6-133
DP8472/NS32972/DP8474/NS32974 Floppy Disk Controller Plus 6-143

National Semiconductor Corporation

DP8461/65 Data Separator
DP8451/55 Data Synchronizer

General Description

DP8461/65

The DP8461/65 Data Separators are designed for applications in disk drive memory systems, and depending on system requirements, may be located either in the drive or in the controller. They receive digital pulses from a pulse detector circuit (such as the DP8464 Disk Pulse Detector) if situated in the drive, or from an ST506 type interface if situated in the controller. After locking on to the frequency of these input pulses, they separate them into synchronized data and clock signals. While in the non-read mode, both of these circuits employ a phase-frequency comparator to keep the VCO locked to the 2F input (this signal may be derived from a crystal or a servo track). The DP8465 switches to a phase only comparator when the read mode is entered. The DP8461 continues to use a phase-frequency comparator until the preamble detection circuit has detected two bytes of preamble. This feature thus restricts the DP8461 to use with codes employing the 1010 . . . preamble. MFM, and certain RLL Codes such as 1,7 and 1,8 employ such a preamble. If a Run Length Limited code is used or if the user wishes to do his own data separation, the synchronized data output is available to allow external circuitry to perform the data decoding function.

All of the digital input and output signals are TTL compatible and only a single +5V supply is required. The chip is housed in a narrow 24-pin dual-in-line package (DIP) and is fabricated using Advanced Schottky bipolar analog and digital circuitry. This high speed I.C. process allows the chip to work with data rates up to 20 Mbit/sec. There are two versions of the chip, each having a different decode window error specification. These two versions (−3, −4) are designed to operate from 2 to 20 Mbit/sec and are tested for their respective window tolerances, as specified in the Electrical Characteristics Table.

The DP8461/65 feature a phase-lock-loop (PLL) consisting of a phase-frequency comparator, pulse gate (to allow for phase-only operation in the read mode), charge pump, buffering amplifier, and voltage-controlled-oscillator (VCO). Pins are provided for the user to select the values of the external filtering components required for the VCO, and two current setting resistors for the charge pump. The DP8461/65 have been designed to be capable of locking onto the incoming preamble data pattern within the first two bytes, using an available high rate of charge pump current. Once lock-on has been achieved, the charge pump can be switched to a lower rate (both rates being determined by the external resistors) to improve bit-jitter immunity for the remainder of the read operation. At this time the READ CLOCK OUTPUT switches, without glitching, from half the 2F-CLOCK frequency to half the VCO CLOCK frequency. After lock-on, with soft sectored disks, the MISSING CLOCK DETECTED output indicates when a missing clock occurs so the controller can align byte boundaries to begin deserialization of the incoming data.

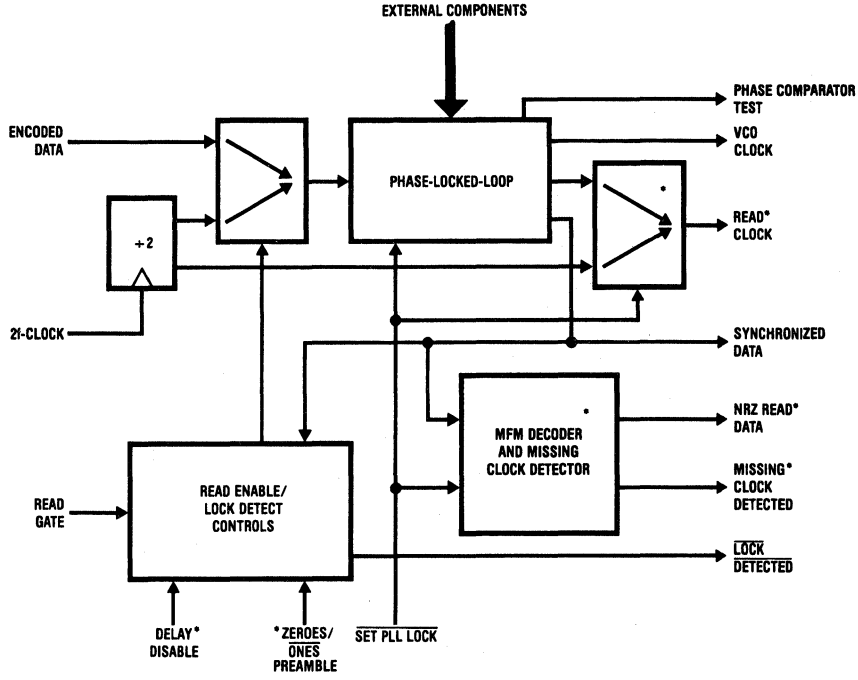
DP8451/55

The DP8451/55 perform the same data synchronization function of the DP8461/65 with no MFM related circuitry. As with the DP8461, the DP8451 continues in the phase-frequency comparison mode until two bytes of preamble are detected. The DP8451/55, which are packaged in 20 pin DIPs or 20-pin PCC's, exclude the READ CLOCK generating circuitry along with the MFM Decoder, Missing Clock Detector, and Read Enable Delay. Users who require only the SYNCHRONIZED DATA OUTPUT and VCO CLOCK OUTPUT can use the DP8451/55 as alternatives to the DP8461/65.

Features

- Operates at data rates up to 20 Mbit/sec
- Phase-Frequency comparison in non-read mode
- Phase-Frequency comparison in preamble—DP8461/51
- Separates MFM data into read clock and serial NRZ data (DP8461/65)
- 4 byte preamble-lock indication capability
- Preamble recognition of MFM encoded "0"s or "1"s
- User-determined PLL loop filter network
- PLL charge pump has two user-determined tracking rates
- External control of track rate switchover
- No glitch on READ CLOCK at switchover (DP8461/65)
- Synchronized data provided as an output (for RLL codes) (all four devices)
- ORed phase comparator outputs for monitoring bit-shift
- Missing clock detected for soft sectored disks
- Less then ½W power consumption
- Standard narrow 24-pin DIP or 28 pin Plastic Chip Carrier Package
- Single +5V supply

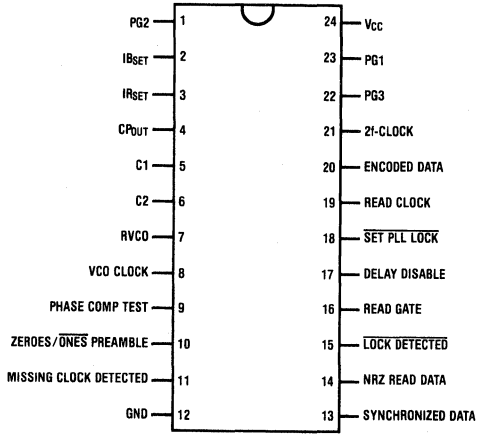
Simplified Block and Connection Diagrams



*Available only on DP8461/65

TL/F/8445-1

DP8461/65
Dual-In-Line Package

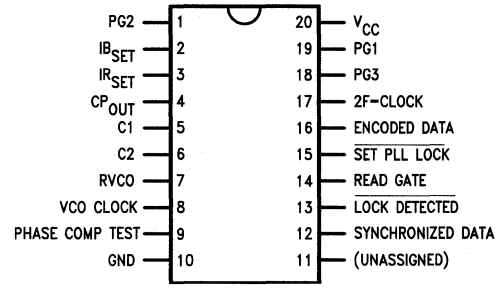


Top View

TL/F/8445-2

Order Number DP8461/65J or N
See NS Package Number J24F or N24C

DP8451/55
Dual-In-Line Package



Top View

TL/F/8445-3

Order Number DP8451/55N
See NS Package Number N20A

Pin Descriptions*

Power Supply

24 V_{CC} +5V ±5%

12 Ground

TTL Level Logic Inputs

16 READ GATE: This is an active high input signal that sets the DP8461/65 Data Separator into the Read Mode.

17 DELAY DISABLE: This input determines the delay from READ GATE going high to the time the DP8461/65 enters the Read Mode. If DELAY DISABLE is set high, this delay is within one cycle of the V_{CO}-CLOCK signal. If DELAY DISABLE is set low, the delay is thirty two-cycles of the VCO CLOCK, as shown in *Figure 1*.

18 $\overline{\text{SET PLL LOCK}}$: This input allows the user to control the on-chip PLL track rate. A high level at this input results in the PLL being in the high track rate. If this input is connected to the $\overline{\text{LOCK DETECTED}}$ output, the PLL will go into the low track rate mode immediately after lock is detected. A low level on this pin is also used to enable the MFM Decoder, the Missing Clock Detector, and to switch the Read Clock Multiplexer from half-2F-CLOCK to half-VCO.

10 ZEROES/$\overline{\text{ONES}}$ PREAMBLE: A high level on this input enables the MFM Decoder circuit to recognize an All Zeros data preamble. A low level results in the recognition of an All Ones data preamble.

20 ENCODED DATA: This input is connected to the output of the head amplifier/pulse-detecting network located in the disk drive. Each positive edge of the ENCODED DATA waveform identifies a change of flux on the disk. In the case of MFM encoded data, the input will be raw MFM.

21 2f-CLOCK: This is a system clock input, which is either a signal generated from the servo track (for systems utilizing servo tracks), or a signal buffered from a crystal. 2f CLOCK MUST ALWAYS BE APPLIED TO THIS INPUT FOR PROPER OPERATION.

TTL Level Logic Outputs

8 VCO CLOCK: This is the output of the on-chip VCO, transmitted from an Advanced Schottky-TTL buffer. It is synchronized to the MFM data output.

15 $\overline{\text{LOCK DETECTED}}$: This output goes active low only after both PLL Lock has occurred and 16 pulses of the preamble pattern have been recognized. It remains low until READ GATE goes inactive.

14 NRZ READ DATA: This is the NRZ (decoded MFM) data output, whose leading edges coincide with the trailing edge of READ CLOCK.

13 SYNCHRONIZED DATA: This output is the same encoded data that is input to the chip, but is synchronous with the negative edge of the VCO CLOCK.

11 MISSING CLOCK DETECTED: When an MFM missing clock is detected, this output will be a single pulse (of width equal to one cycle of READ CLOCK) occurring as shown in *Figure 2*.

19 READ CLOCK: This is half VCO CLOCK frequency when $\overline{\text{SET PLL LOCK}}$ is low; it is half 2f-CLOCK frequency at all other times. A deglitcher is utilized to ensure that no short clock periods occur during either switchover.

9 PHASE COMP TEST: This output is the logical "OR" of the Phase Comparator outputs, and may be used as a bit-shift indicator on for PLL analysis purpose.

Analog Signals

23, 22, PG1, PG3: The external capacitors and resistor of the Pulse Gate filter are connected to these pins. PG1 should be connected directly to the ground pin, pin 12.

1 PG2: This is the Pulse Gate current supply.

3 IRSET: The current into the rate set pin (V_{BE}/R_{RATE}) is used to set the charge pump output current for the low tracking rate.

2 IBSET: The current into the boost set pin (V_{BE}/R_{BOOST}) is used to set the amount by which the charge pump current is increased for the high tracking rate. (I_{INPUT} = I_{RATE} Set + I_{BOOST} Set).

4 CPOUT: CHARGE PUMP OUT/BUFFER AMP IN is available for connection of external filter components for the phase-lock-loop. In addition to being the charge pump output node, this pin is also the noninverting input to the Buffer Amplifier.

7 RVCO: The current at this pin determines the operating currents within the VCO.

5, 6 VCO C1, C2: An external capacitor connected between these pins sets the nominal VCO frequency.

*Pin Number Designations apply only to the DP8461/65. See Connection Diagram for DP8451/55.

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage	7V
TTL Inputs	7V
Output Voltages	7V
Input Current (CPOUT, IRSET, IBSET, RVCO)	2 mA
Storage Temperature	−65°C to 150°C

Operating Conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{CC}	Supply Voltage		4.75	5.00	5.25	V
T_A	Ambient Temperature		0	25	70	°C
I_{OH}	High Logic Level Output Current	VCO Clock Others			−2000 −400	μA
I_{OL}	Low Logic Level Output Current	VCO Clock Others			20 8	mA

Operating Conditions (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
f_{DATA}	Input Data Rate		2.0		20	Mbit/sec
t_{WCK}	Width of 2f-CLOCK, High or Low		10			ns
t_{WPD}	Width of ENCODED DATA Pulse, (Note 2)	HIGH	5 ns + 0.10t			ns
		LOW	0.4t			
V_{IH}	High Logic Level Input Voltage		2			V
V_{IL}	Low Logic Level Input Voltage				0.8	V
t_{SETUP} (READ Gate)	Min. Amount of Time Which a Positive Edge of READ Gate Must Precede a Negative Edge of a VCO (Pin 8)		20			ns
t_{HOLD} (READ Gate)	Min. Time Required for a Positive Edge of a READ Gate to be Held after a Negative Edge of a VCO (Pin 8)		10			ns

DC Electrical Characteristics Over Recommended Operating Temperature Range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IC}	Input Clamp Voltage	V_{CC} = Min., I_I = −18 mA			−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min., I_{OH} = Max.	V_{CC}−2V	V_{CC}−1.6V		V
V_{OL}	Low Level Output Voltage	V_{CC} = Min., I_{OL} = Max.			0.5	V
I_{IH}	High Level Input Current	V_{CC} = Max., V_I = 2.7V			20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max., V_I = 0.4V			−200	μA
I_O	Output Drive Current (Note 1)	V_{CC} = Max., V_O = 2.125V	−12		−110	mA
I_{CC}	Supply Current	V_{CC} = Max.			100	mA
I_{OUT}	Charge Pump Output Current	500 μA ≤ I_{RSET} + I_{BSET} ≤ 2000 μA 200 μA ≤ I_{RSET} + I_{BSET} < 500 μA	I_{TYP} − .18 (I_R + I_B) − 30 μA I_{TYP} − .08 (I_R + I_B) − 80 μA	1.95 (I_{RSET} + I_{BSET}) − 70 μA 1.95 (I_{RSET} + I_{BSET}) − 70 μA	I_{TYP} + .18 (I_R + I_B) + 30 μA I_{TYP} + .08 (I_R + I_B) + 80 μA	μA

Note 1: This value has been chosen to produce a current that closely approximates one-half of the true short-circuit output current. I_{OS}.

Note 2: t is defined as the period of the encoded MFM data, or two times the VCO period.

AC Electrical Characteristics Over Recommended V_{CC} and Operating Temperature Range.

(All Parts unless stated otherwise) (t_R = t_F = 2.0 ns, V_{IH} = 3.0V, V_{IL} = 0V)

Symbol	Parameter	Min	Typ	Max	Units
t_{READ}	Positive READ CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE low)		16	17	—
t_{READ}	Positive READ CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE high)		1	1	—
$t_{DECODE\ NRZ}$	Number of READ CLOCK cycles required to output each decoded MFM data bit (Note 3, 4)	—	2	3	T-clock
$t_{TRANSMIT\ MFM}$	Positive READ CLOCK transitions required to transmit input MFM to output	1	2	3	—

Note: For Further Information Refer to Application Notes AN-414, AN-415, and AN-416.

AC Electrical Characteristics Over Recommended V_{CC} and Operating Temperature Range. (Continued)

(All Parts unless stated otherwise) ($t_R = t_F = 2.0$ ns, $V_{IH} = 3.0V$, $V_{IL} = 0V$)

Symbol	Parameter	Min	Typ	Max	Units
$t_{READ\ ABORT}$	Number of READ CLOCK cycles after READ GATE set low to read operation abort			2	T-clock
t_{WINDOW}	Variance of center of decode window from nominal (Note 7) DP84XX-3			6	ns
	DP84XX-4			10	
$\phi_{LINEARITY}$	Phase range for charge pump output linearity (Note 2)	$-\pi$		$+\pi$	Radians
K_1	Phase Comparator—Charge Pump gain constant (Note 5) ($N = f_{VCO}/f_{INPUT\ DATA}$, $2 \leq N \leq 4$ for MFM)		$\frac{1.78V_{BE}}{N2\pi R}$		Amps/rad
$V_{CONTROL}$	Charge pump output voltage swing from nominal		± 100		mV
K_{VCO} ($=A \times K_2$)	VCO gain constant (ω_{VCO} = VCO center frequency in rad/s) (Note 1, 6)	$\frac{1.20\omega_C}{V_{BE}}$	$\frac{1.40\omega_C}{V_{BE}}$	$\frac{1.60\omega_C}{V_{BE}}$	rad/sec. V
f_{VCO}	VCO center frequency variation over temperature and V_{CC}	-5		$+5$	%
$f_{MAX\ VCO}$	VCO maximum frequency		60		MHz
t_{HOLD}	Time READ CLOCK is held low during changeover after lock detection has occurred (Note 3)			1½	T-clock
t_{PHL}	Prop. Delay. VCO Neg. Edge to Synchronized Data Neg. Edge		15	30	ns
t_{PLH}	Prop. Delay. VCO Negative Edge to Synchronized Data Positive Edge		10	25	ns
$t_{2F/RC}$	Delay from 2F positive edge to READ CLOCK positive on negative edge ($\overline{\text{SET PLL LOCK}}$ high)	10		35	ns

Note 1: A sample calculation of frequency variation vs. control voltage: $V_{IN} = \pm 0.1V$;

$$K_{VCO} = \frac{\omega OUT}{V_{IN}} = \frac{0.4\omega_C}{0.2V} = \frac{2.0\omega_C}{V} \frac{(rad/sec)}{(volt)}$$

Note 2: $-\pi$ to $+\pi$ with respect to 2f VCO CLOCK

Note 3: T-clock is defined as the time required for one period of the READ CLOCK to occur.

Note 4: This number remains fixed after PLL Lock occurs.

Note 5: With respect to VCO CLOCK; $I_{PUMP\ OUT} = 1.9\ I_{SET}$

$$I_{SET} = \frac{V_{BE}}{R_{SET}}$$

Note 6: Although specified as the VCO gain constant, this is the gain from the Buffer Amplifier input to the VCO output.

Note 7: This specification is guaranteed only for the conditions under which the parts were tested. However, significant variation from the formula is not expected for other data rates and filters. The filter values below were chosen for operation in an automatic test system (static window) environment. Different criteria may apply for choosing filter values in a disk system. See Loop Filter section for sample calculations of other filter values.

Static Window Margin Test Loop Filter Component Values

Part Type	Data Rate Tested	C_1	C_2	R_1	R_{RATE}	R_{BOOST}
DP8451/55/61/65-4	5 Mbit/Sec	0.02 μF	150 pF	200Ω	750Ω	1.6 kΩ
DP8451/55/61/65-3	10 Mbit/Sec	.082 μF	1600 pF	27Ω	820Ω	619Ω

External Component Selection (All Parts) (Note 1)

Symbol	Component	Min	Typ	Max	Units
R_{VCO}	VCO Frequency Setting Resistor (Note 2)	990		1010	Ω
C_{VCO}	VCO Frequency Setting Capacitor (Note 3, 4)	20		245	pF
R_{RATE}	Charge Pump I_{RATE} Set Resistor (Note 6)	0.4		4.0	kΩ
R_{BOOST}	Charge Pump (High Rate) I_{BOOST} Resistor (Note 6)	0.5		∞	kΩ
C_R	I_{RATE} Bypass Capacitor (Note 5)	.01			μF
C_B	I_{BOOST} Bypass Capacitor (Note 5)	.01			μF

Note 1: External component values for the Loop Filter and Pulse Gate are shown in tables 1 & 2.

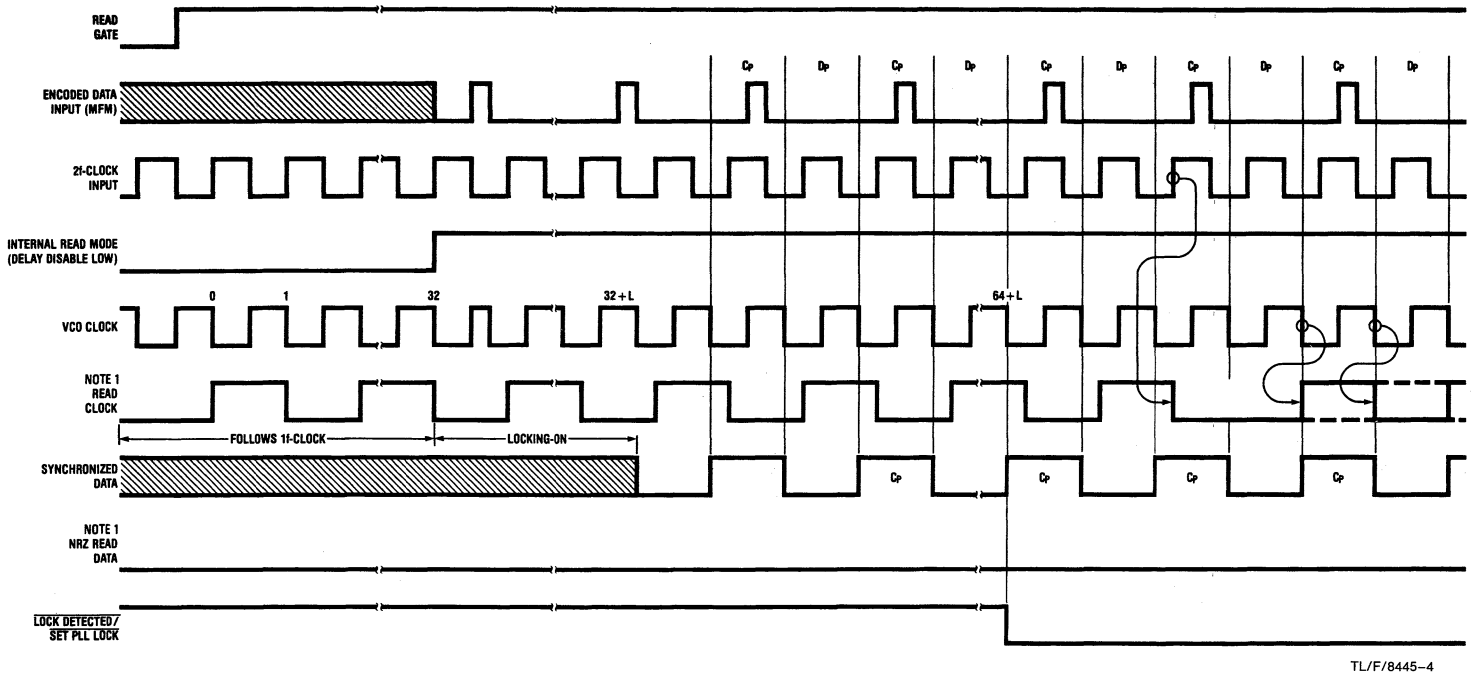
Note 2: A 1% Component Tolerance is Required.

Note 3: These MIN and MAX values correspond to the MAX and MIN data rates respectively.

Note 4: The Component Tolerance is system dependent on how much center frequency deviation can be tolerated.

Note 5: Component Tolerance 15%.

Note 6: The minimum value of the parallel combination of R_{RATE} and R_{BOOST} is 350Ω.



Note 1: Not included on the DP8451/55.

C_P, D_P = preamble clock and preamble data bits respectively.

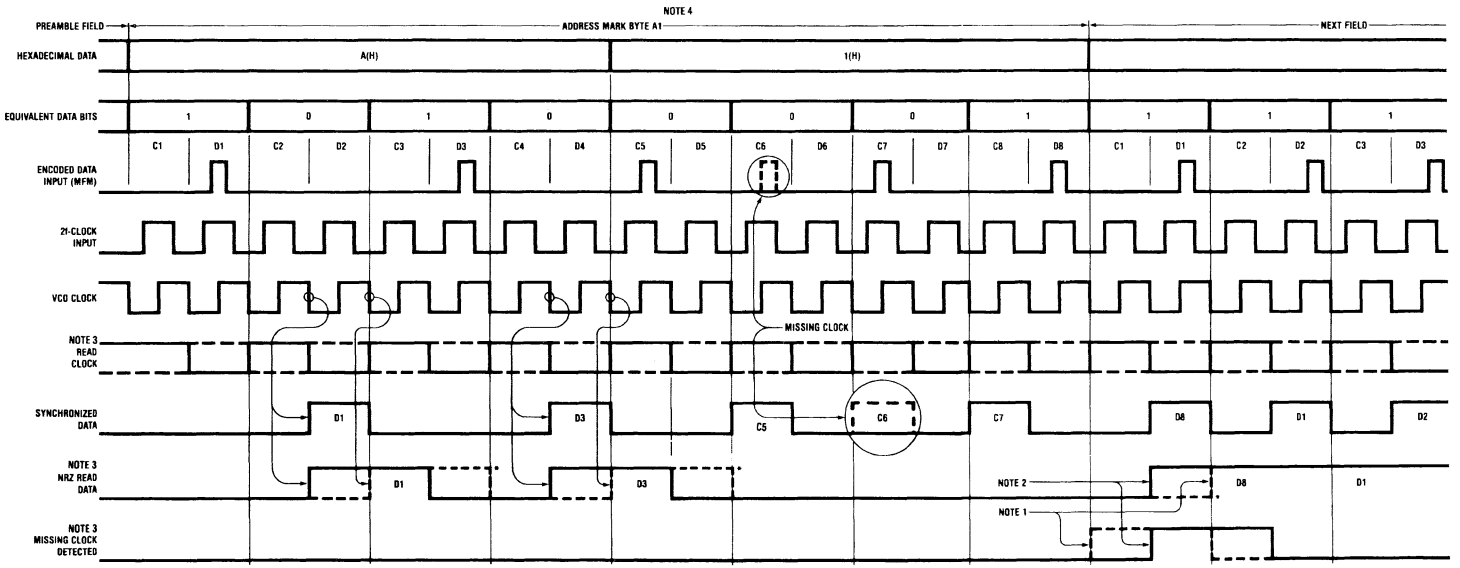
L = Number of 2f-clock cycles required for VCO to lock, determined by external loop filter component values

At 32 + L, VCO has just locked.

At 64 + L, circuit has confirmed lock (has been in lock for 16 MFM clock bits). This sequence shows the MFM all-zeros preamble pattern.

For DP8451/55 delay disable does not exist and part functions as if this input is always high.

FIGURE 1. Lock-on Sequence Waveform Diagram



TL/F/8445–5

* READ CLOCK and NRZ READ DATA may be delayed by one VCO clock period depending on the phase of the internal clock at activation of READ GATE input.

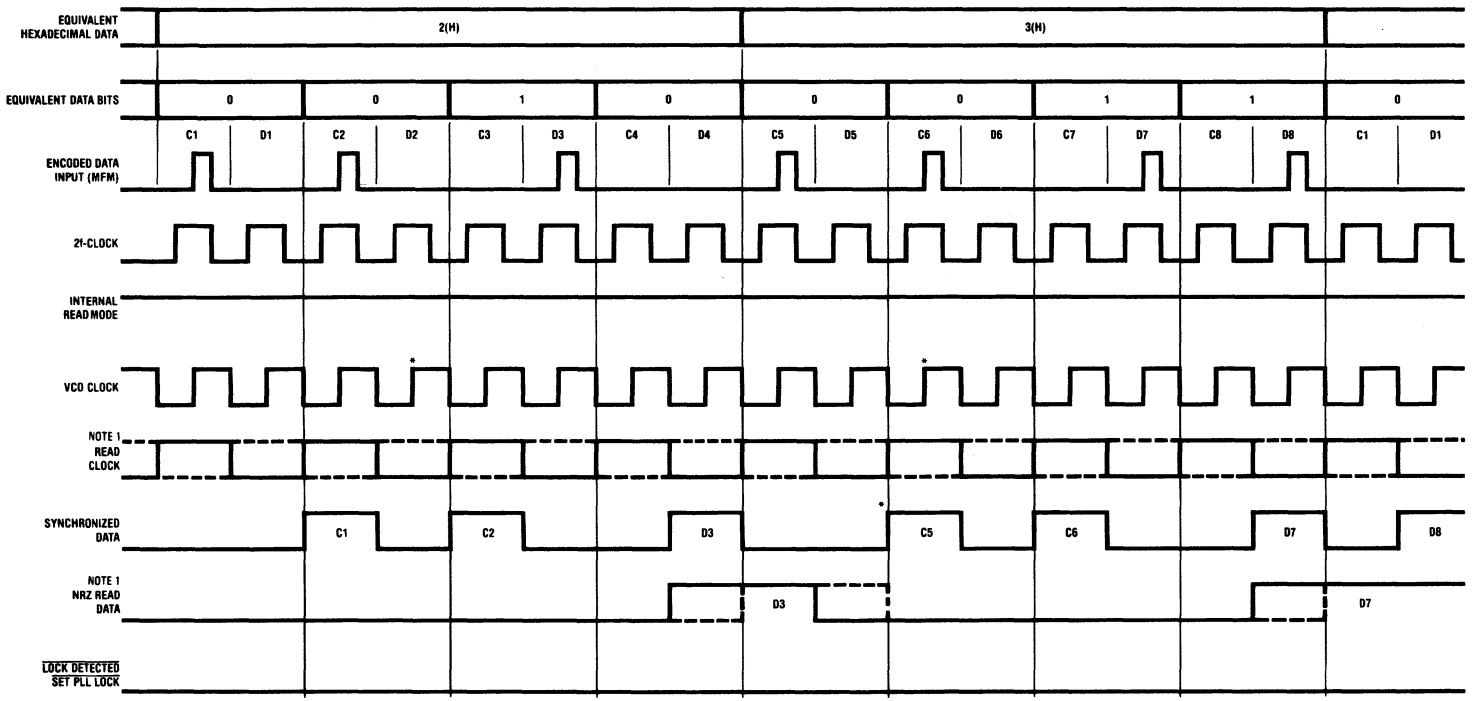
① MISSING CLOCK DETECTED is one READ CLOCK period ahead of the chip issuing D8 on the NRZ READ DATA output when READ CLOCK is delayed by one VCO clock period.

② MISSING CLOCK DETECTED is synchronous with the chip issuing D8 on the NRZ READ DATA Output when READ CLOCK is not delayed.

③ Not included on the DP8451/55.

④ The AI byte is shown only as an example address mark byte. *Any* missing clock bit which is framed by two existing clock bits will produce a missing clock detected pulse.

FIGURE 2. Missing Clock Detection Waveform Diagram

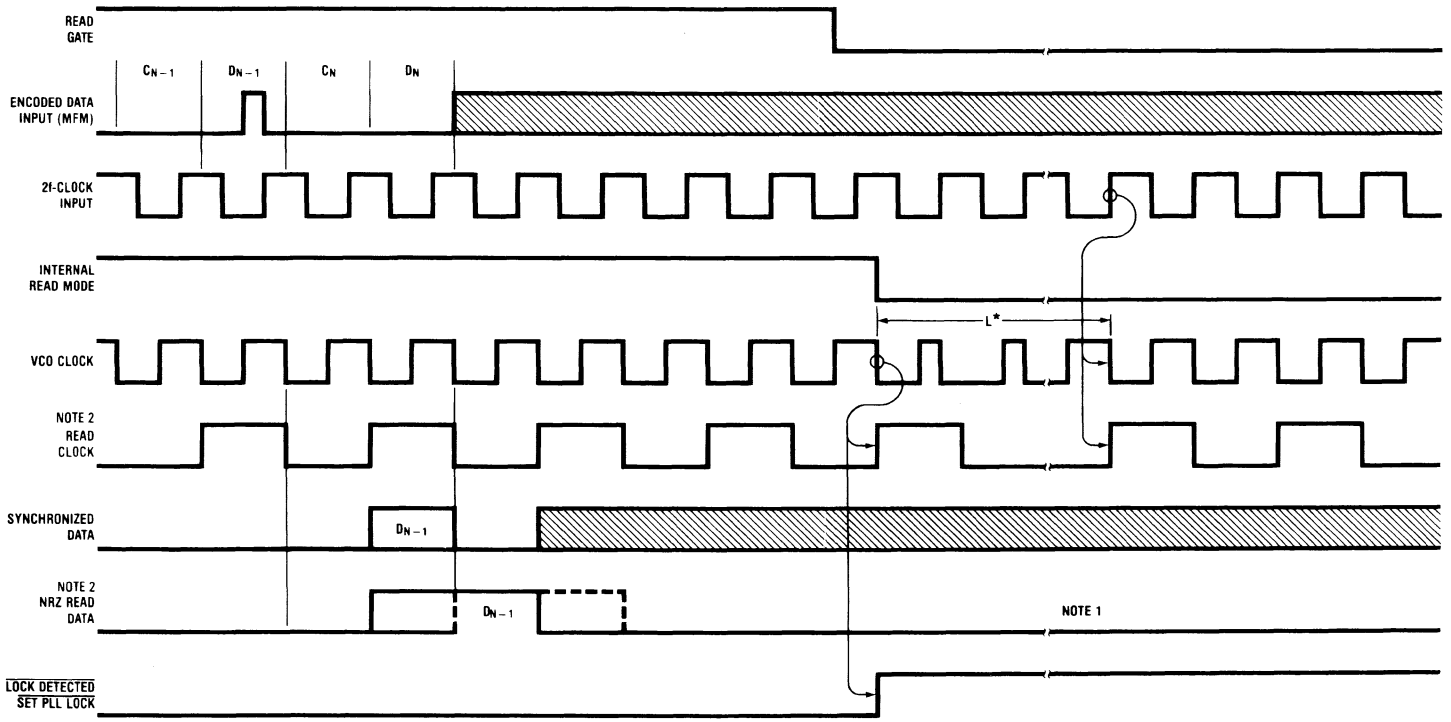


TL/F/8445–6

* **READ CLOCK and NRZ READ DATA may be delayed by one VCO clock period with respect to Synchronized Data depending on the phase of the internal clock at activation of READ GATE input.**

Note 1: Not included on the DP8451/55.

FIGURE 3. Locked-On Waveform Diagram



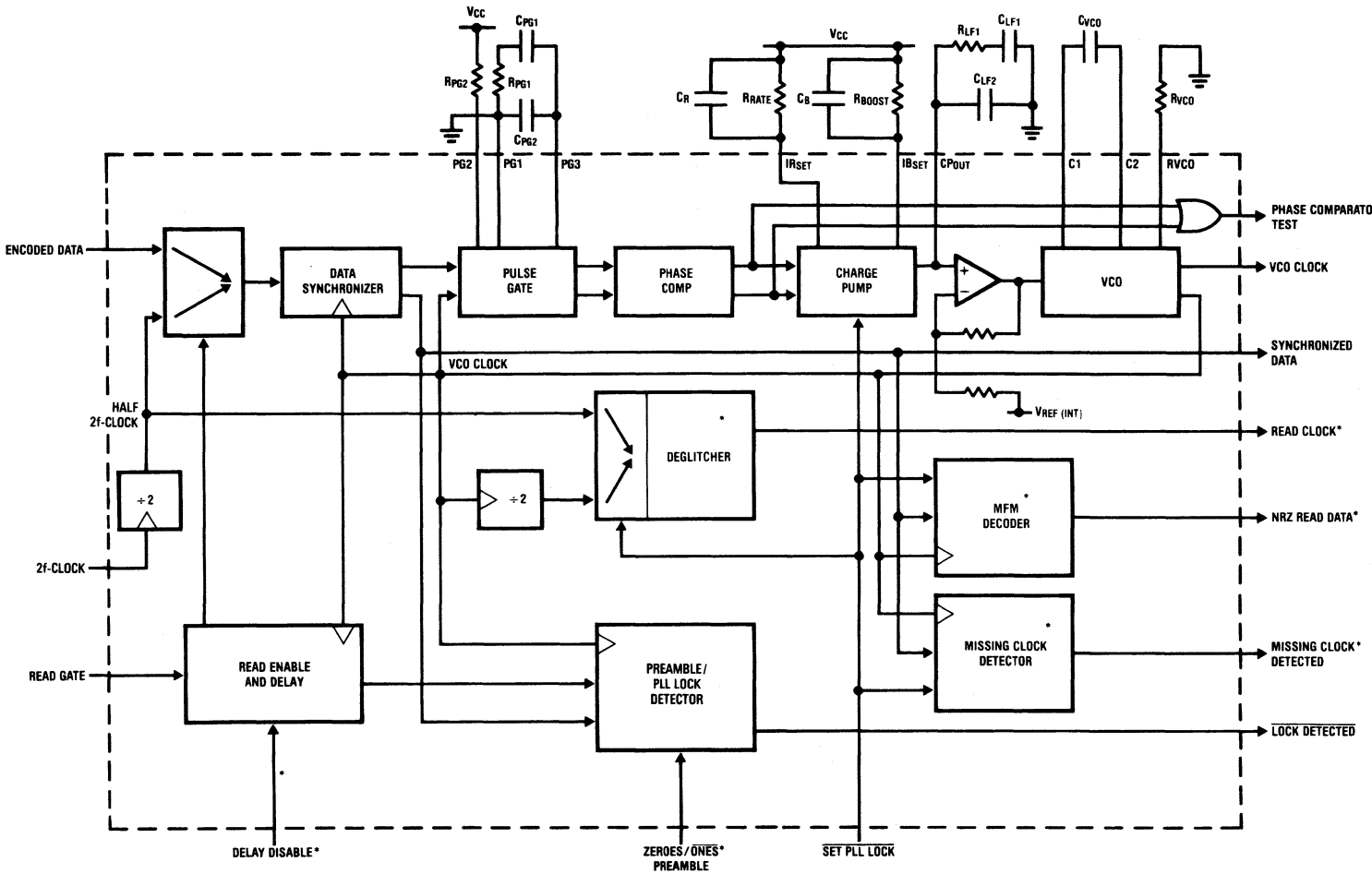
* **L indicates the number of cycles required for the VCO to lock to the 2f-CLOCK.**

Note 1: READ GATE going low will always result in NRZ READ DATA going low regardless of the state of the last bit.

Note 2: Not included on the DP8451/55.

FIGURE 4. Lock-Ending Sequence Waveform Diagram

Detailed Block Diagram



TL/F/8445–8

*Not included on the DP8451/55

Circuit Operation

When the READ GATE input goes high, the DP8461/65 will enter the read mode after a period determined by the state of the DELAY DISABLE pin. This may be either one or thirty two VCO CLOCK cycles. Once in the read mode the DP8465 switches from using a phase-frequency comparator to a phase-only comparator, i.e. the pulse gate is activitated. At this time, however, the DP8461 continues to use a phase-frequency comparator. Referring to *Figure 1*, as the read mode is entered, the phase-locked-loop reference signal is switched from 2F-CLOCK INPUT to the ENCODED DATA. The PLL, initially in the high-tracking rate mode, then attempts to lock onto the incoming encoded data stream. As soon as two bytes of the selected preamble are detected, (the selection is determined by the ZEROES/$\overline{\text{ONES}}$ PREAMBLE pin) the $\overline{\text{LOCK DETECTED}}$ OUTPUT goes low. At this time the DP8461 switches from using a phase-frequency comparator to using the pulse gate, thus beginning phase only comparisons. In a typical MFM disk drive application, the $\overline{\text{LOCK DETECTED}}$ OUTPUT is directly connected to the $\overline{\text{SET PLL LOCK}}$ INPUT. With this connection, track rate selection, clock output switchover, and data output enabling will occur after two consecutive preamble bytes have been detected by the chip. Typically it takes less than one byte time for the VCO to lock to the data sufficiently for preamble detection to begin following the start of the Read operation.

A low level on the $\overline{\text{SET PLL LOCK}}$ causes the PLL Charge Pump to switch from the high to low tracking rate. At the same time, the source of the READ CLOCK signal is switched from half the frequency of the 2F-CLOCK to half the VCO CLOCK. The MFM decoder also becomes enabled and begins to output decoded NRZ data. If a zeroes data preamble is present, the NRZ READ DATA OUTPUT will remain low until the end of the preamble. It will then output whatever NRZ data is present after the preamble field has ended, as shown in *Figures 2* and *3*.

When the READ GATE goes low, signifying the end of a read operation, the DP8461/65 will return to phase-frequency comparator operation. *Figure 4* shows the sequence when READ GATE goes low. The PLL reference signal is switched back to half the 2F-CLOCK and the $\overline{\text{LOCK DETECTED}}$ OUTPUT (and therefore the $\overline{\text{SET PLL LOCK}}$ INPUT) goes high. The PLL then returns to the high track rate, and the output signals return to their initial conditions. The 2F-CLOCK MUST BE APPLIED AT ALL TIMES to the DP8461/65 and DP8451/55 for proper operation.

Since the DP8461/51 employs a phase-frequency comparator until two bytes of the preamble (actually any 16 pulses within a 1010 . . . pattern) have been detected, care must be taken to ensure that when using this circuit the READ GATE is applied only within a field containing the 1010 . . . pattern. In soft sectored drives the head may be positioned anywhere on the track when initiating a read operation. Therefore, either a controller which only issues READ GATE when a high frequency synchronization field is present, or a simple external circuit between the controller and DP8461/51 to qualify the READ GATE, must be used.

CIRCUIT DESCRIPTION

1. Read Enable and Delay (DP8461/65 only): If the DELAY DISABLE input is connected low, then thirty two VCO CLOCK cycles after READ GATE goes active, the DP8461/65 will go into the read mode. If the DELAY DISABLE input is connected high, the chip will go into the read mode one VCO CLOCK cycle after READ GATE goes active. (The 32 cycle delay is permanently disabled in the DP8451/55).

2. Pulse Gate, including Multiplexer and Data Synchronizer: The Input Multiplexer selects the input to the phase-lock-loop. While the chip is in the bypassed (non-read) mode, the VCO frequency is phase and frequency locked to the 2F-CLOCK INPUT frequency. In the read mode the Input Multiplexer switches to the ENCODED DATA signal and the VCO CLOCK then begins to synchronize with the ENCODED DATA signal. Also, as soon as the read mode is entered, the DP8455/65 cease phase and frequency comparisons by employing the Pulse Gate.

In the DP8461/51 option, switchover from the phase-frequency comparator to the pulse gate (phase-only comparator) occurs after two bytes of the 1010 . . . pattern have been detected by the preamble pattern detector.

The Pulse Gate allows a reference pulse from the VCO into the Phase Comparator only after an ENCODED DATA bit has arrived. It utilizes a scheme which delays the incoming data by one-half the period of the 2F-CLOCK. This optimizes the position of the decode window and allows input jitter of approximately half the 2F-CLOCK period. The decode window error can be determined from the specification in the Electrical Characteristics Table.

3. Phase Comparator: The Phase Comparator receives its inputs from the Pulse Gate, and is edge-triggered from these inputs to provide charge-up and charge-down outputs.

4. Charge Pump: The high speed charge pump consists of a switchable constant current source and sink. The charge pump constant current is set by connecting external resistors to V_{CC} from the charge current rate set (IRSET) and current boost set (IBSET) pins. Before lock is indicated, the PLL is in the high tracking rate and the parallel combination of the resistors determines the current. In the low tracking rate after lock-on, only the IRSET resistor determines the charge pump current. The output of the charge pump subsequently feeds into external filter components and the Buffer Amplifier.

5. Buffer Amplifier: The input of the Buffer Amplifier is connected to the charge pump's constant current source/sink output as well as the external Loop Filter components. The Buffer Amplifier is configured as a high input impedance amplifier which allows for the connection of external PLL filter components to the Charge Pump output pin CPOUT. The output of the Buffer Amplifier is internally connected to the VCO control input.

6. VCO: The Voltage-Controlled-Oscillator requires a resistor from the RVCO pin to ground and a capacitor between pins C1 and C2, to set the center frequency. The VCO frequency can be varied from nominal by approximately ±20%, as determined by its control input voltage.

7. PLL Lock-on/Preamble Pattern Detector: To recognize preamble, the preamble pattern from the disk must consist exclusively of either MFM data bit zeroes (encoded into ..10.. MFM clock pulses) when the ZEROES/$\overline{\text{ONES}}$ PREAMBLE pin is set high, or MFM data bit ones (encoded into ..01.. MFM pulses) when set low (DP8461/65 only). The preamble pattern must be long enough to allow the PLL to lock, and subsequently for the Preamble Pattern Detector circuit to detect two complete bytes.

Once the chip is in the read mode, the VCO proceeds to lock on to the incoming data stream. The Preamble Pattern

Circuit Operation (Continued)

Detector then searches for a continuous pattern of 16 consecutive pulses at one-half the VCO frequency to indicate lock has been achieved.

The $\overline{\text{LOCK DETECTED}}$ output then goes low. At this time, in the DP8461/51 option, the PLL switches from using a phase-frequency comparator to employing a pulse gate and thus doing only phase comparisons. Any deviation from the above-mentioned one-zero pattern at any time before PLL Lock is detected will reset the PLL Lock Detector. The lock detection procedure will then start again.

8. MFM Decoder (DP8461/65 only): The MFM Decoder receives synchronized MFM data from the Pulse Gate and converts it to NRZ READ DATA. For run-length-limited codes the MFM Decoder and Missing Clock Detector will not be used.

9. Missing Clock Detector (DP8461/65 only): This block is only required for soft-sectored drives, and is used to detect a missing clock violation of the MFM pattern. The missing clock is inserted when writing to soft-sectored disks to indicate the location of the Address Mark in both the ID and the Data fields of each sector. Once PLL Lock has been indicated, the Missing Clock Detector circuit is enabled. MISSING CLOCK DETECTED will go active if at any time the incoming data pattern contains one suppressed clock bit framed by two adjacent clock bits. (This condition is not constrained to any particular byte pattern such as "A1.") The output signal goes high for one cycle of READ CLOCK.

10. Clock Multiplexer and Deglitcher (DP8461/65 only): When the $\overline{\text{SET PLL LOCK}}$ input changes state this circuit switches the source of the READ CLOCK signal between the half 2f-CLOCK frequency and the half VCO CLOCK frequency. A deglitcher circuit is utilized to ensure that no short clock periods occur during either switchover.

BIT JITTER TOLERANCE

The spec, t-window, as defined in the AC Electrical Characteristics table, describes the distance from the optimum window boundary a single shifted data bit may be placed (following complete PLL lock and stabilization) before it risks being interpreted as residing in the adjacent synchronization window. This is known as the **static window measurement**, which combines all contributing factors of window jitter and displacement within the data separator into a single specification.

The two options of the DP8451/55/61/65, the −4 and −3 offer decreasing static window errors (respectively) so that the parts may be selected for different data rates (up to 20 Mbit/sec). The −4 part will be used in most low data rate applications. As an example, at the 5 Mbit/sec MFM data rate of most 5¼ inch drives, the chip contributes up to ±10 ns of window error, out of the total available window of 100 ns. This allows the disk drive to have a margin of 40 ns of jitter from nominal bit position before an error will occur.

ANALOG CONNECTIONS

External passive components are required for the Pulse Gate, Charge Pump, Loop Filter and VCO as shown in *Figure 5*. The information provided here is for guidelines only. The user should select values according to his own system requirements. Phase-Locked Loops are complex circuits that require detailed knowledge of the specific system. Factors such as loop gain, stability, response to change of signal, lock-on time, etc are all determined by the external components. In many disk systems these factors are critical, and National Semiconductor recommends the designer be knowledgeable of phase-locked-loops, or seek the advice of an expert. Inaccurate design will probably result in excessive disk error rates. The phase-locked-loop in the DP8461/65 has many advantages over all but the most sophisticated discrete designs, and if the component values are selected correctly, it will offer significant performance advantages. This should result in a reduction of disk error rates over equivalent discrete designs. Please refer to the National Semiconductor Application Note AN-414, Precautions for Disk Data Separator Designs, AN-415, Designing with the DP8461, AN-416, Designing with the DP8465, and to the Disk Interface Design Guide and User's Manual, Chapter 1.

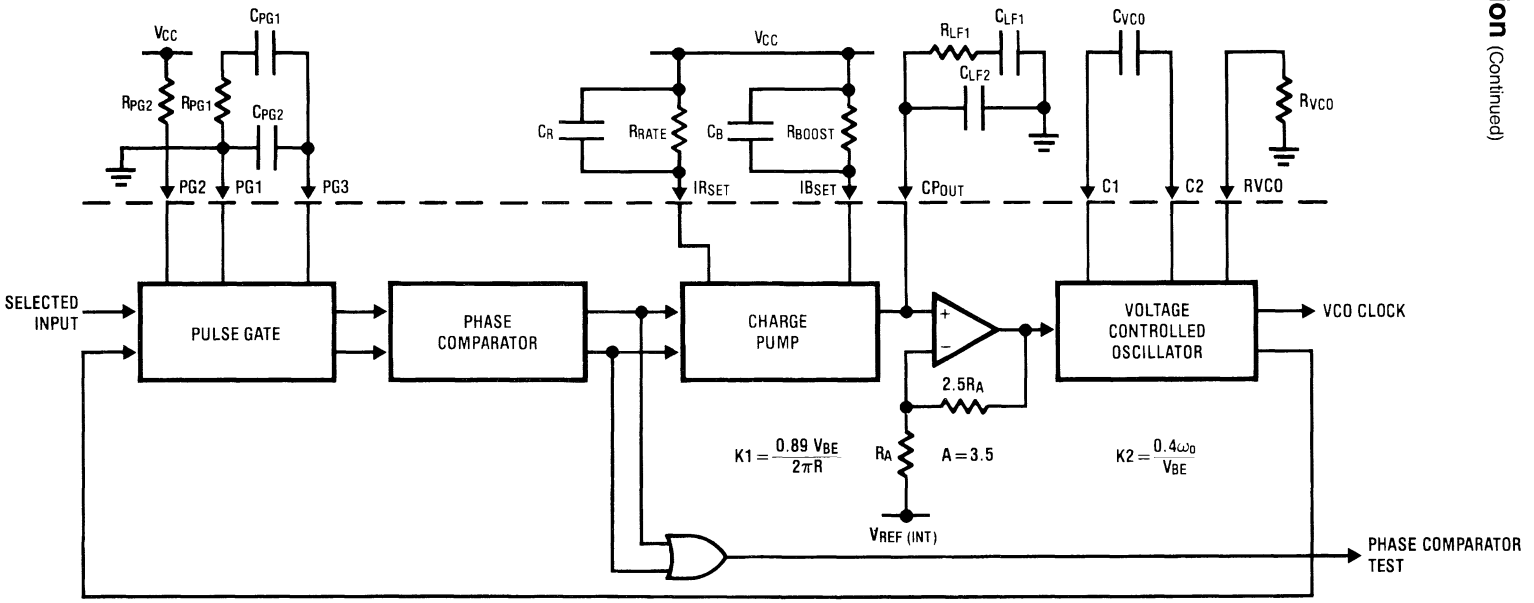


FIGURE 5. Phase-Locked-Loop Section

Circuit Operation (Continued)

Pulse Gate

There are four external components connected to the Pulse Gate as shown in *Figure 6* with the associated internal components. The values of R_{PG1}, R_{PG2}, C_{PG1}, and C_{PG2} are dependent on the data rate. C_{PG1} and C_{PG2} are proportional to the data rate, while R_{PG1} and R_{PG2} are inversely proportional. Table I shows component values for the data rates given. Component values are calculated by selecting R_{PG2} from Table I. Next calculate

$$C_{PG1} = \left(\frac{2.12 \times 10^5}{890 + R_{PG2}}\right)\left(\frac{1}{100 \times R_S}\right)^2$$

$$C_{PG2} = \frac{1}{10} C_{PG1}, \text{ and } R_{PG1} = \left(\frac{890 + R_{PG2}}{2.38 \times 10^5}\right)(100 \times R_S).$$

In the above equations R_S is the rotational speed and, for 3600 RPM, R_S = 60 Hz. A rotational speed of 3600 RPM was assumed for the calculations in Table I. For data rates not listed, R_{PG2} may be approximated as (30 kΩ/f_{DATA}) − 1.20 kΩ = R_{PG2} where f_{DATA} is the data rate in Megabits/second.

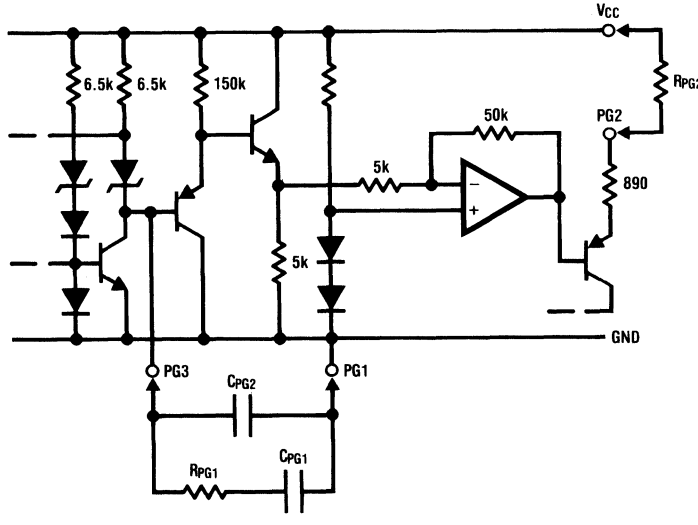
TABLE I. Pulse Gate Component Selection Chart
Components with 10% tolerance will suffice

Data Rate	R_{PG2}	R_{PG1}	C_{PG1}	C_{PG2}
2 Mbit/sec	15 kΩ	430Ω	.39 μF	.039 μF
5 Mbit/sec	4.7 kΩ	150Ω	1 μF	0.1 μF
10 Mbit/sec	1.8 kΩ	68Ω	2.2 μF	.22 μF
15 Mbit/sec	750Ω	39Ω	3.9 μF	.39 μF

Charge Pump

Resistors R_{RATE} and R_{BOOST} determine the charge pump current. The Charge Pump bidirectional output current is approximately 1.9 × the input current (See DC Electrical Characteristics for exact relationship). In the high tracking rate with $\overline{\text{SET PLL LOCK}}$ high, the input current is I_{BSET} + I_{RSET}, i.e., the sum of the currents through R_{BOOST} and R_{RATE} from V_{CC}. In the low tracking rate, with $\overline{\text{SET PLL LOCK}}$ low, this input current is I_{RSET} only.

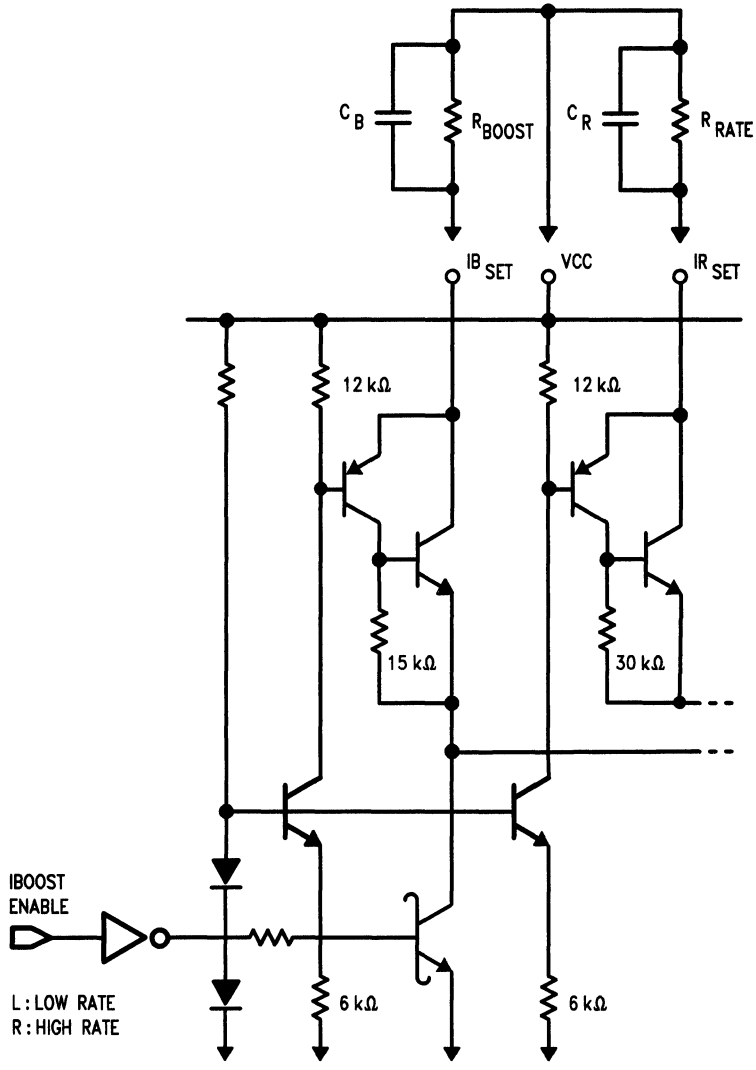
A recommended approach for selecting values for R_{RATE} and R_{BOOST} is described in the design example in the Loop Filter Section. A typical loop gain change of 2:1 for high to low tracking rate would require R_{BOOST} = R_{RATE}. Selecting R_{RATE} to be 820Ω would then result in R_{BOOST} equaling 820Ω. Referring to *Figure 7,* the input current is effectively V_{BE}/R_{RATE} in the low tracking rate, where V_{BE} is an internal voltage. This means that the current into or out of the loop filter is approximately (1.95 × V_{BE}/820) − 70 μA = 1.72 mA. Note that although it would seem the overall gain is dependant on V_{BE}, this is not the case. The VCO gain is altered internally by an amount inversely proportional to V_{BE}, as detailed in the section on the Loop Filter. This means that as V_{BE} varies with temperature or device spread, the gain will remain constant for a particular fixed set of values of R_{RATE} and R_{BOOST}. This alleviates the need for potentiometers to select values for each device. The tolerance required for these two resistors will depend on the total loop gain tolerance allowed, but 5% would be typical. Also V_{CC} bypass capacitors are required for these two resistors. A value of .01 μF is suitable for each.



TL/F/8445–10

FIGURE 6. Pulse Gate Controls

Circuit Operation (Continued)



TL/F/8445–11

FIGURE 7. I_{RATE} Set and I_{BOOST} Set

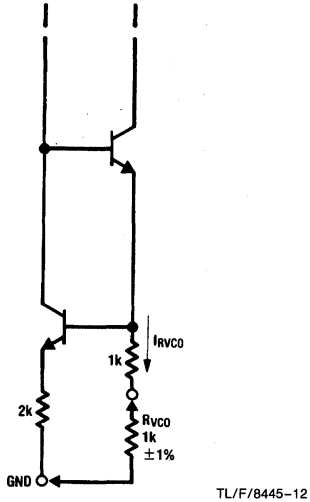
Circuit Operation (Continued)

VCO

The value of R_{VCO} is fixed at 1 kΩ ±1% in the External Component Limits table. *Figure 8* shows how R_{VCO} is connected to the internal components of the chip. This value was fixed at 1 kΩ to set the VCO operating current such that optimum performance of the VCO is obtained for production device spreads. This means fixed value components will be adequate to set the VCO center frequency for production runs. The value of C_{VCO} can therefore be determined from the VCO frequency f_{VCO}, using the equation: $C_{VCO} = [1/(R_{VCO})(f_{VCO})] - 5$ pF where f_{VCO} is twice the input data rate. As an example, for a 5 Mbit/sec data rate, $f_{VCO} = 10$ MHz, requiring that $C_{VCO} = 95$ pF. This does not take into account any inter-lead capacitance on the printed circuit board; the user **must** account for this. The amount of toler ance a particular design can afford on the center frequency will determine the capacitor tolerance. The capacitor is connected to internal circuitry of the chip as shown in *Figure 9.*

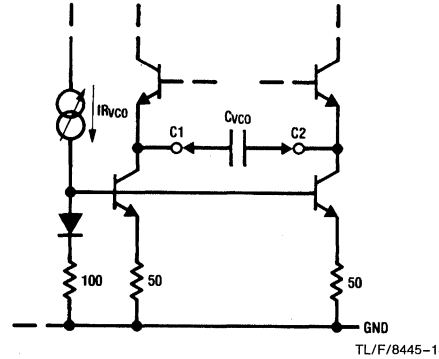
As the data rate increases and C_{VCO} gets smaller, the effects of unwanted internal parasitic capacitances influence the frequency. As a guide the graph of *Figure 10* shows approximately the value of C_{VCO} for a given data rate.

The VCO control input operational range (pin 4) lies at approximately 1.4 volts with a control swing of ±100 millivolts. The VCO itself is constrained to swing a maximum of approximately ±20% of its center frequency, and will remain clamped if the voltage at pin 4 exceeds its operational limit. The VCO center frequency may then be determined by: 1) holding pin 4 at ground potential and measuring the VCO frequency (−20% value); 2) holding pin 4 at approximately 3 volts and measuring the VCO frequency (+20% value); 3) averaging the two measured frequencies for the equivalent center frequency.



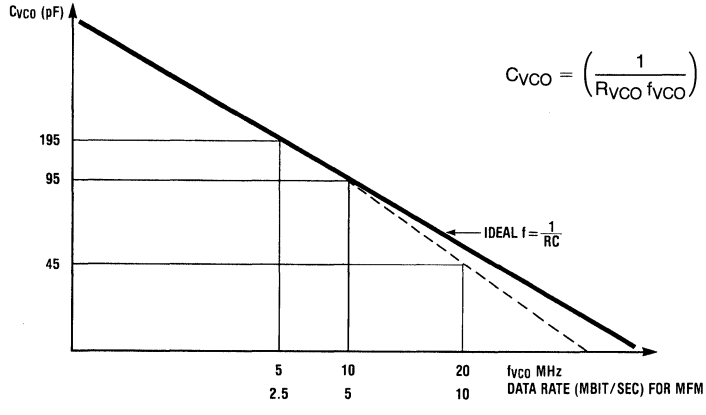
TL/F/8445–12

FIGURE 8. VCO Current Setting Resistor



TL/F/8445–13

FIGURE 9. VCO Capacitor



TL/F/8445–14

FIGURE 10. VCO Capacitor Value for Disk Data Rates

Circuit Operation (Continued)

Loop Filter

The input current into the Buffer Amplifier is offset by a matched current out of the Charge Pump, and even so is much less than the switching current in or out of the Charge Pump. It can therefore be assumed that all the Charge Pump switching current goes into the Loop Filter components R_1 and C_1 and C_2. The tolerance of these components should be the same as R_{RATE} and R_{BOOST}, and will determine the overall loop gain variation. The three components connected to the Charge Pump output are shown in *Figure 11*. Note the return current goes to analog GND, which should be electrically very close to the GND pin itself.

The value of capacitor C_1 determines loop bandwidth ... the larger the value the longer the loop takes to respond to an input change. If C_1 is too small, the loop will track any jitter on the ENCODED DATA input and the VCO output will follow this jitter, which is undesirable. The value of C_1 should therefore be large enough so that the PLL is fairly immune to phase jitter but not large enough that the loop won't respond to longer term data rate changes that occur on the disk drive.

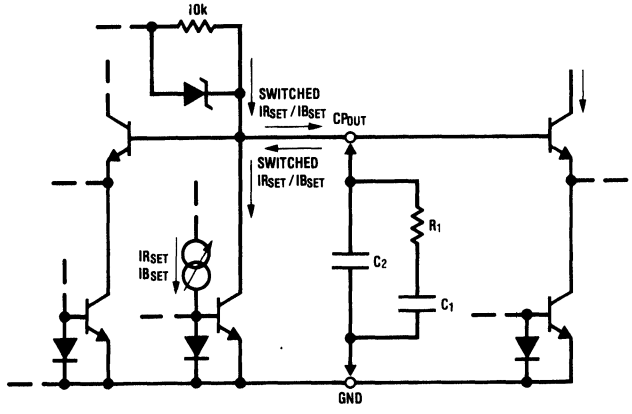
The damping resistor R_1 is required to regulate the second-order behavior of the closed-loop system (overshoot). A value of R_1 that would give a phase margin of around 45 degrees would be a reasonable starting point.

The main function of the capacitor C_2 is to smooth the action of the charge pump at the VCO input. Typically its value will be less than one tenth of C_1. Further effects of C_2 will be discussed later.

Figure 12 shows the relevant phase-locked-loop blocks that determine system response, namely the Phase Detector, Filter/Buffer Amplifier, and VCO. The Phase Detector (Phase Comparator and Charge Pump) produces an aggregate output current i which is proportional to the phase difference between the input signal and the VCO signal. The constant (K_1) is

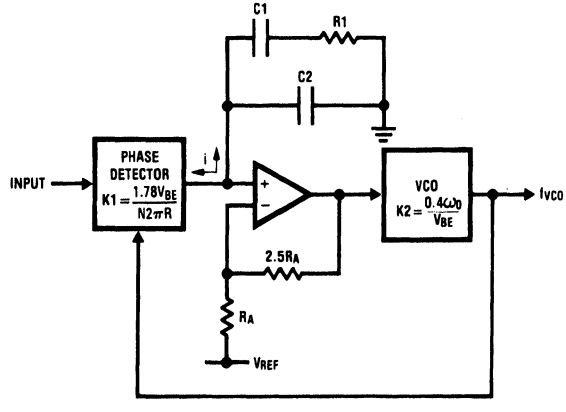
$$\frac{1.78\ V_{BE}}{N2\pi R} \text{ amps per radian, where } N = \frac{f_{VCO}}{f_{DATA}}.$$

R is either R_{RATE} or $R_{RATE} \parallel R_{BOOST}$. The amplified aggregate current feeds into or out of the filter impedance (Z), producing a voltage to the VCO that regulates the VCO frequency. The VCO gain constant is 0.4 ω_{VCO}/V_{BE} radians per second per volt. Under steady state conditions, i will be zero and there will be no phase difference between the input signal and the VCO. Any change of input signal will pro-



TL/F/8445–15

FIGURE 11. Charge Pump Out



TL/F/8445–16

FIGURE 12. Loop Response Components

Circuit Operation (Continued)

duce a change in VCO frequency that is determined by the loop gain equation. This equation is determined from the gain constants K_1, A and K_2 and the filter v/i response.

The impedance Z of the filter is:

$$\frac{1}{sC_2} \parallel \left(\frac{1}{sC_1} + R_1\right) = \frac{1 + sC_1R_1}{sC_1\left(1 + \frac{C_2}{C_1} + sC_2R_1\right)}$$

If $C_2 \ll C_1$ then the impedance Z approximates to:

$$\frac{1 + sC_1R_1}{sC_1(1 + sC_2R_1)}$$

The overall loop gain is then

$$G(s) = \frac{K_1AK_2}{s} \times \frac{1 + sC_1R_1}{sC_1(1 + sC_2R_1)}$$

Let $G_{(K)} = K_1 A K_2$

$$F(s) = \frac{1 + SC_1R_1}{SC_1(1 + SC_2R_1)}$$

The Overall Closed Loop Gain is:

$$\frac{\phi OUT}{\phi IN} = \frac{G_{(K)} F(s)}{s + G_{(K)} F(s)}$$

Substituting, We Get

$$\frac{\phi OUT}{\phi IN} = \frac{G_{(K)}(SC_1R_1 + 1)}{S^3 R_1 C_1 C_2 + S^2 C_1 + GK(SC_1R_1 + 1)}$$

$$= \frac{(G_{(K)}/C_1)(SR_1C_1 + 1)}{S^3R_1C_2 + S^2 + SG_{(K)}R_1 + G_{(K)}/C_1}$$

If $C_2 \ll C_1$, we can ignore the 3rd Order Component introduced by C_2 then:

$$\frac{\phi OUT}{\phi IN} = \frac{(G_{(K)}/C_1)(SR_1C_1 + 1)}{S^2 + SG_{(K)}R_1 + G_{(K)}/C_1}$$

This is a second Order Loop and can be solved as follows:

$$S^2 + SG_{(K)}R_1 + G_{(K)}/C_1 = S^2 + 2\zeta\omega_nS + \omega_n^2$$

$$\therefore C_1 = \frac{G_{(K)}}{\omega_n^2}$$

$$R_1 = \frac{2\zeta\omega_n}{G_{(K)}}$$

$\zeta = 1.0$ For Critically Damped Response

From the above equations:

$$\omega = \sqrt{\frac{G_{(K)}}{C_1}}$$

$$G_{(K)} = K_1 A K_2 = \frac{0.89 \times V_{BE}}{2\pi R} \times \frac{0.4 \times \omega VCO}{V_{BE}} \times 3.5$$

MFM encoded data has a two to one frequency range within the data field. The expression $K = (0.89 \times V_{BE} / 2\pi R)$ is valid when the MFM data pattern is at its maximum frequency. In order to make this equation more general, it may be written as follows: $K = (1.78 \times V_{BE} / 2\pi RN)$ where N is defined as the V_{CO} frequency divided by the encoded data frequency, or, N is equal to F_{VCO}/F_{DATA} (N = 2 for maximum data rate i.e., MFM = 101010 ... and N = 4 for minimum data rate) i.e., MFM = 100010001 .... Now $G_{(K)}$ can be written as follows:

$$G_{(K)} = \frac{1.78 \times V_{BE}}{2\pi RN} \times \frac{0.4 \times \omega VCO}{V_{BE}} \times 3.5$$

$$= \frac{2.5 \times F_{VCO}}{RN}$$

$$\omega_n = \sqrt{\frac{2.5 \times F_{VCO}}{C_1RN}}$$

R = R_{RATE} in the low track rate

R = R_{RATE} // R_{BOOST} in the high track rate

From the above equations:

$$\omega_n = \frac{R_1 G_{(K)}}{2\zeta}$$

$$G_{(K)} = C_1 \omega_n^2$$

$$\zeta = \text{(damping factor)} = \frac{R_1 \omega_n C_1}{2}$$

The damping factor should approach, but not fall below, 0.5 when ω_n is minimum. Response to bit shift is minimized when the damping factor is small; however, if the damping factor drops much below 0.5, the system tends to be oscillatory (underdamped).

Additionally, loop performance is poor (excessive phase acquisition times) if the damping factor becomes significantly greater than 1.0. Any increase in loop bandwidth (due to R decreasing in the high track rate) produces a proportional increase in the damping factor, and this should be limited to the point where the maximum damping factor does not significantly exceed 1.0. With the damping factor range established, loop design can now proceed. The following design example is for a 5 Mbit/sec MFM system.

A 1550 Krads/sec bandwidth in the non read mode results in a wide capture range; a 4% frequency difference between the crystal and recorded data would not cause an acquisition problem. (This bandwidth may seem excessive to some and if the user does not think it is necessary, he may design his filter with a more desirable bandwidth. For an in-depth discussion of this point, it is suggested that the reader refer to the Disk Interface Design Guide and User's Manual, chapter 1, sections 1.3 through 1.7.

This design example assumes that the $\overline{\text{SET PLL LOCK}}$ pin is tied to the $\overline{\text{PLL LOCK DETECTED}}$ pin. This results in the track rate being switched from high to low after two bytes of preamble are detected. As an alternative, the $\overline{\text{SET PLL LOCK}}$ pin may be tied to an inverted READ GATE signal, resulting in the track rate switching immediately to low when READ GATE is asserted. This is discussed further in the above mentioned reference material.

TABLE II.

Data Rate (NRZ)	Non-Read		Read		Charge Pump		Loop Filter		
	$\omega_{n(MAX)}$ rads/sec	ζ	$\omega_{n(MIN)}$ Rads/sec	ζ	R_{RATE} Ω	R_{BOOST} Ω	R_1 Ω	C_1 μF	C_2 pF
5 Mbit/sec	1550K	1.12	797K	0.55	820	820	120	0.012	300
5 Mbit/sec	903K	0.99	435K	0.48	1500	1300	100	0.022	390
5 Mbit/sec	659K	1.55	248K	0.52	1500	590	69	0.068	1500

Circuit Operation (Continued)

In the non read mode or high track rate.

$$\omega_n = \sqrt{\frac{2.5 \times F_{VCO}}{C_1 RN}}$$

Choose $R = R_{RATE} // R_{BOOST} = 410$

In the non-read mode $N = 2$

$$1550 \text{ Krads/sec} = \sqrt{\frac{2.5 \times 10^7}{C_1 \times 410 \times 2}}$$

$C_1 = 0.012\ \mu F$

In the preamble, after two bytes are detected and $\overline{\text{PLL LOCK DETECT}}$ goes low

$$\omega_n = \sqrt{\frac{2.5 \times F_{VCO}}{C_1 RN}}$$

$R = R_{RATE} = 820$

$N = 2$

$\omega_n = 1127$ Krads/sec

Again, in the data field, the minimum data frequency is equal to one half the preamble frequency. This means that $N = 4$ in the bandwidth equation. This reduces the bandwidth to:

$$\omega_{n(min)} = \frac{1}{\sqrt{2}} \times 1107 \text{ Krads/sec} = 797 \text{ Krads/sec}$$

Before, we stated that the minimum value of ζ should be 0.5; knowing $\omega_{n(min)}$ we can now solve for R_1

$$\zeta = \frac{\omega_n R_1 C_1}{2}$$

Choose $\zeta_{(min)} = 0.55$

$$R_1 = \frac{2\zeta}{\omega_n C_1}$$

$R_1 = 115\Omega$ (choose 120Ω)

The maximum damping value occurs in the high track rate;

$$\zeta_{(max)} = \omega_{n(max)} R_2 C_1/2$$
$$= 1550 \text{ Krads/sec} \times 120 \times 0.012\ \mu F/2$$

$\zeta_{(max)} = 1.12$

The maximum damping value in the read mode is as follows:

$\zeta_{(max-read)} = 1127 \text{ Krads/sec} \times 120 \times 0.012\ \mu F/2$

$\zeta_{(max-read)} = 0.81$

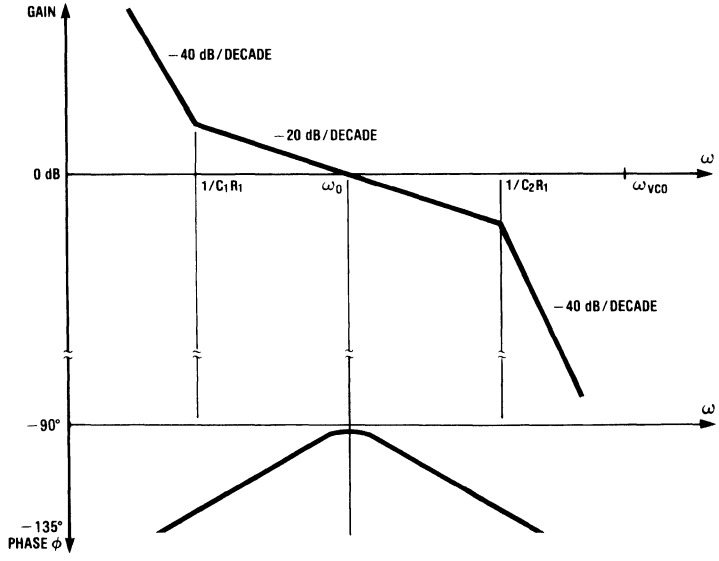
The continuous behavior (non-quantized) approximation used to predict loop performance assumes that the phase detector output is constantly proportional to the input phase difference. In reality, the phase detector output is a pulse applied for a period of time equal to the phase difference. The function of C_2 is to smooth the phase detector output (VCO control voltage) over each cycle. C_2 also adds a second pole to the filter transfer function. This pole should be far enough outside the loop bandwidth (at least one order of magnitude) that its phase and amplitude contribution is negligible in the loop bandwidth. If:

$C_2 = C_1/50 = 240$ pF (choose 300 pF)

The final loop component is R_{BOOST}. Since R_{RATE} and the parallel combination of R_{RATE} and R_{BOOST} are known, we can calculate R_{BOOST}.

$R_{BOOST} = (R_P)(R_{RATE}) / (R_{RATE} - R_P) = 820\Omega$

The above filter values and those for other bandwidths are listed on preceding page.



TL/F/8445–17

FIGURE 13. Bode Plot of Loop Response

Circuit Operation (Continued)

The calculated values are only a guide, the user should then empirically test the loop and determine stability, lock-on time, jitter tolerance, etc.

The desired Bode plot of gain and phase is shown in *Figure 13*, with 20 dB/decade slope at ω_o for stability at unity gain.

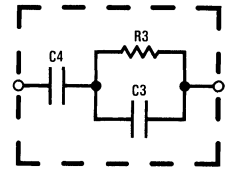
Capacitor C_2 governs the PLL's ability to reject instantaneous bit jitter. As C_2 increases in value, the effective jitter rejection will also increase. However, as the frequency of the pole R_1 and C_2 produce (while increasing C_2) decreases, loop stability will decrease, and the second-order approximation used to analyze the circuit becomes inaccurate. Thus, it is recommended that C_2 remain one tenth (or less) the value of C_1.

The value of resistor R_1 inversely effects the break frequencies on the Bode plot, and directly effects the loop's damping ratio (overshoot response). The capacitor C_1 governs the bandwith of the loop. Too high a value will slow down the response time, but make the PLL less prone to jitter or frequency shift whereas too low a value will improve response time while tending to increase the PLL's reaction to jitter.

Other filter combinations may be used, other than R_1 in series with C_1, all in parallel with C_2. For example the filter shown in *Figure 14* will also perform similarly, and in fact for some systems it will yield superior performance.

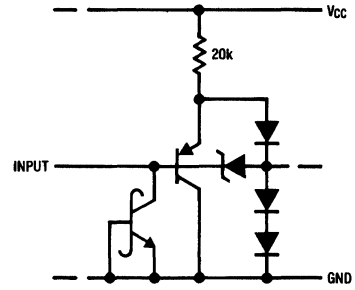
DIGITAL CONNECTIONS TO THE DP8461/65

Figure 17 shows a connection diagram for the DP8461/65 in a typical application. All logic inputs and outputs are TTL compatible as shown in *Figure 15* and *16*. The VCO CLOCK output is 74AS compatible. All other outputs are 74ALS compatible. All inputs are 74ALS compatible and therefore can be driven easily from any 74 series devices. The raw MFM from the pulse detector in the drive is connected to the ENCODED DATA input. The DELAY DISABLE input determines whether attempting lock-on will begin immediately after READ GATE is set or after 2 bytes. Typically in a hard-sectored drive, READ GATE is set active as the sector pulse appears, meaning a new sector is about to pass under the head. Normally the preamble pattern does not begin immediately, because gap bytes from the preceding sector usually extend just beyond the sector pulse. Allowing 2 bytes to pass after the sector pulse helps ensure that the PLL will begin locking on to preamble, and will not be chasing non-symmetrical gap bits. Thus DELAY DISABLE should be set low for this kind of disk drive.



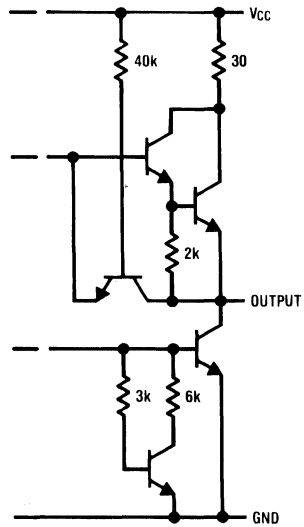
TL/F/8445–18

FIGURE 14. Alternate Loop Filter Configuration



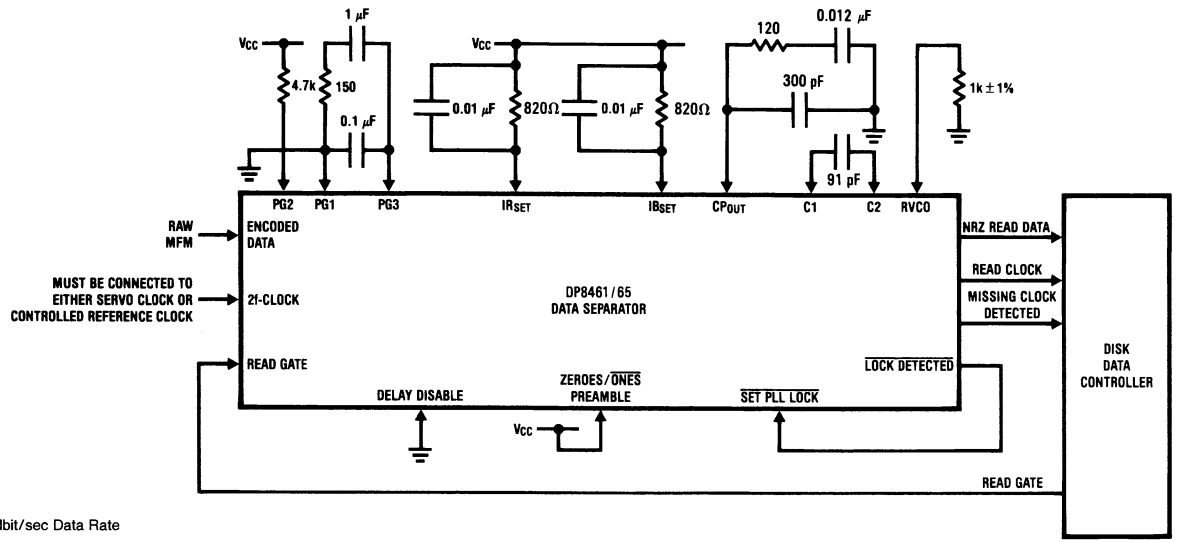
TL/F/8445–19

FIGURE 15. Logic Inputs



TL/F/8445–20

FIGURE 16. Logic Outputs

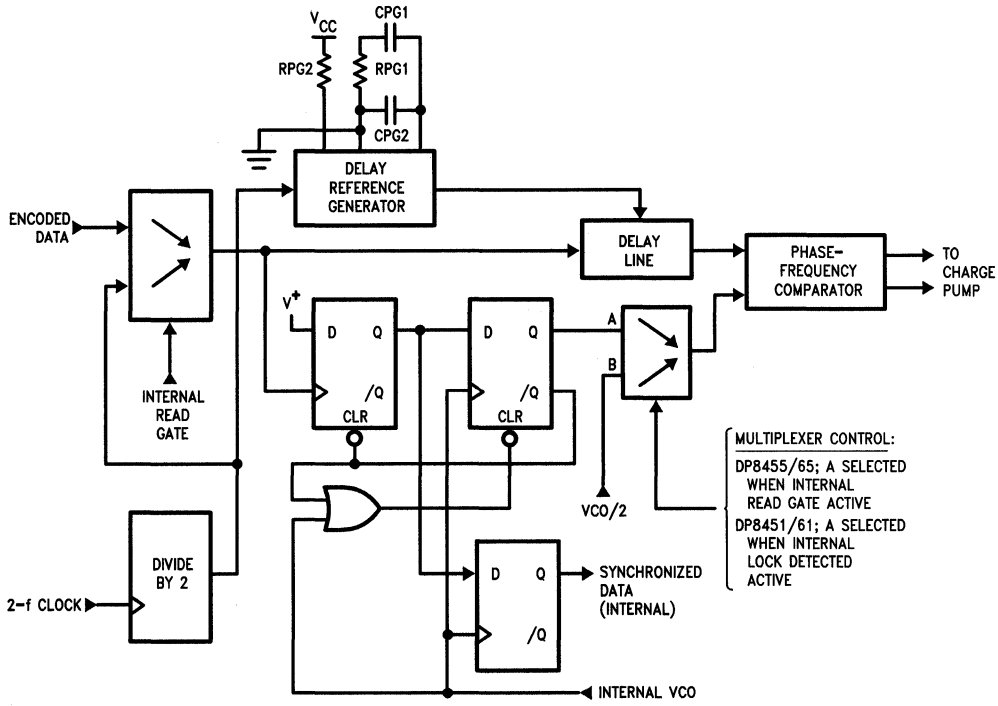


TL/F/8445–21

1) MFM Data Input, 5 Mbit/sec Data Rate
2) 32 Bit Delay to Enable
3) All Zeroes (NRZ) Preamble

FIGURE 17. Typical Connection to DP8461/65

Block Diagram



TL/F/8445–25

Circuit Operation (Continued)

For soft sectored drives, the controller normally will not wait for the index pulse before it attempts lock-on, so that READ GATE may go active at any time. Chances are the head will not be over a preamble field and therefore there is no need to wait 2 bytes before attempting lock-on. DELAY DISABLE can therefore be set high. If a non-preamble field is passing by as READ GATE goes active, the DP8461/65 will not indicate lock, and no data decoding will occur nor will MISSING CLOCK DETECTED go active. Normally, if lock-on has not been achieved after a certain time limit, the controller will de-activate READ GATE and then try again.

For MFM encoded disk drives, the $\overline{\text{LOCK DETECTED}}$ output will be connected back to the $\overline{\text{SET PLL LOCK}}$ input. As the PLL achieves lock-on, the DP8461/65 will automatically switch to the lower tracking rate and decoded data will appear at the NRZ READ DATA output. Also the READ CLOCK output will switch from half the 2F-CLOCK frequency to the disk data rate frequency. If a delay is required before the changeover occurs, a time delay may be inserted between the two pins.

Some drives have an all-ONES data preamble instead of all-ZEROES and the DP8461/65 must be set to the type being used before it can properly decode data. The ZEROES/$\overline{\text{ONES}}$ PREAMBLE input selects which preamble type the chip is to base its decoding phase on.

USE WITH RUN-LENGTH-LIMITED CODES (RLL)

If the drive uses a Run-Length-Limited Code (RLL) such as 1,7 or 1,8 instead of MFM, the user might choose to use the DP8451/55. These circuits contain the PLL portion of the DP8461/65 and thus perform the data synchronization function. RAW DATA is input to pin 16 and the 2F-CLOCK is applied to pin 17. Instead of supplying NRZ DATA, SYNCHRONIZED DATA OUTPUT is issued at pin 12. The VCO CLOCK, pin 8, is used to clock this data into external decoding circuitry. As long as the high frequency pattern of . . . 1010 . . . is used for the preamble, the user may choose the DP8451 if he desires to have the circuit perform phase and frequency comparisons until two bytes of preamble are detected by the on chip preamble pattern detector.

If a 2,7 code is being used the DP8465/55 may be used. Again, since the DP8465 MFM decoding function will not be used, the user may choose to use the DP8455. However, the National Semiconductor DP8462 is designed specifically for the 2,7 code. It is recommended that the user reviews the DP8462 specification for the added advantages the circuit offers with the 2,7 format.

Applications of the DP8461/65 Data Separator

The DP8461/65 are the first integrated circuits to place on one chip a PLL with features that offer the improved speed and reliability required by the disk industry. Not only does each chip simplify disk system design, but also provides fast lock-on to the incoming preamble. Once locked on, the loop is set into a more stable mode. This inherent loop stability allows for a sizeable amount of jitter on the data stream, such as is encountered in many disk systems. Once in the stable tracking rate, the SYNCHRONIZED DATA output represents the incoming ENCODED DATA and is synchronous with VCO CLOCK. If the disk is MFM encoded, then the chip can decode the synchronized data into NRZ READ DATA and READ CLOCK. These are available as outputs from the chip allowing the NRZ READ DATA to be deserialized using the READ CLOCK.

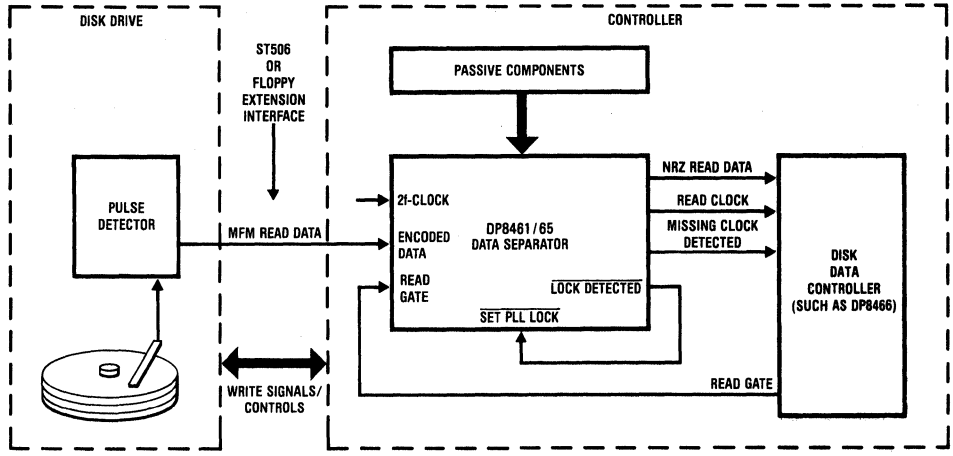
The DP8461/65 are capable of operating at up to 20 Mbits/sec data rates and so are compatible with a wide assortment of disk drives. The faster data rates of the 8-inch and 14-inch disk drives will mandate the selection of the DP8461/65-3 parts with their narrower window margins on the incoming data stream. This will also be the case when 5¼-inch drives achieve higher data rates. Some 8-inch and 14-inch disk drives incorporate the functions of the DP8461/65, but use many discrete ICs. In these cases, replacing these components with the DP8461/65 will offer reduced P.C. board area, lower cost, and improved performance while simplifying circuit testing.

Most 5¼-inch and many 8-inch and 14-inch disk drives manufactured at present do not incorporate any of the functions of the DP8461/65. This is so primarily because the PLL function is difficult to design and implement and requires circuitry which covers a large area of the printed circuit card. This is undesirable both from the drive size aspect and from the cost aspect (the cost includes soldering, testing, and adjusting the components). Consequently, most smaller disk drives output MFM encoded data so that the phase-locked-loop and data separation have to be performed by the controller. The DP8461/65 will therefore replace these functions in controller designs, as shown in *Figure 18*.

System design criteria has become more flexible because the DP8461/65 provide a one-chip solution, requiring only a few external passive components with fixed values. Each operates from a +5V supply, typically consumes about 0.3W, and is housed in a narrow 24-pin package. The circuitry has been designed so that the external resistors and capacitors need not be adjustable; the user chooses the values according to the disk drive requirements. Once selected, they will be fixed for that particular drive type. These features make it possible to transfer these functions to the disk drive, as shown in *Figure 19*. Apart from a slight increase in board area, the advantages outweigh the disadvantages. First, the components selected are fixed for each type of drive and this facilitates the problem of interchangeability of drives. At present, controllers are adjusted to function with each specific drive; with the DP8461/65 in the drive, component adjustment will no longer be required. Second there is often a problem of reliability of data transfer. The data returning from the disk drive is susceptible to noise, bit shift, etc. Soft errors will occur when the incoming disk data bit position is outside the Pulse Gate window as it is being synchronized to the VCO clock in the phase-locked-loop. Obviously, the nearer the PLL is to the data source, the less chance there is that extraneous noise or transmission line imbalances will cause errors to occur. Thus placing the DP8461/65 in the drive will increase the reliability of data transfer within the system.

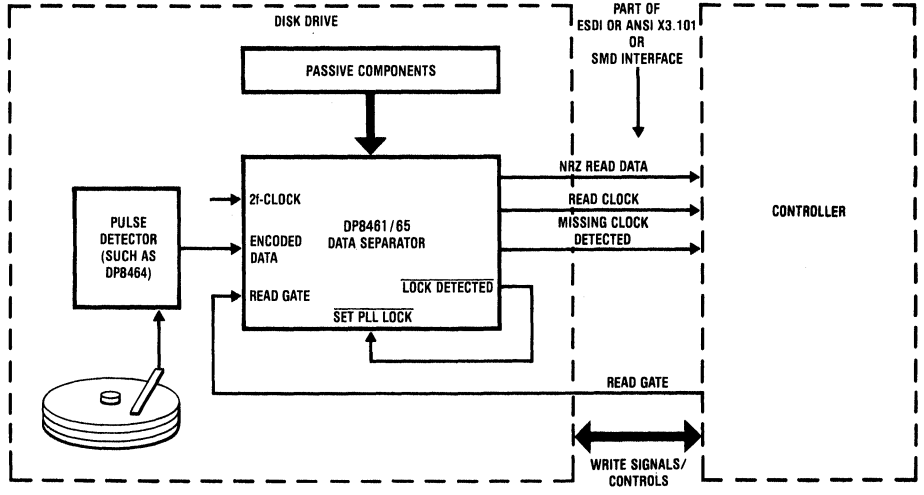
A third advantage is data rate upgrading. Most 5¼-inch drives have 5 Mbit/sec data rate because the early drives were made with this data rate. This meant the controllers had to be designed with PLLs which operate at this data rate. It is therefore difficult for drive manufacturers to introduce new drives that are not compatible with existing controllers. Since no new standard data rate has emerged, they must continue to produce drives at this data rate to be compatible with the controllers on the market. With the DP8461/65 in the drive, and associated components set for the drive's data rate, it no longer becomes a problem to increase the data rate, assuming the controllers digital circuitry can accommodate the change. This will allow the manufactures to increase the bit density and therefore the capacity of their drives.

Applications of the DP8461/65 Data Separator (Continued)



TL/F/8445–22

FIGURE 18. DP8461/65 in the Controller



TL/F/8445–23

FIGURE 19. DP8461/65 in the Disk Drive

PRECAUTIONS IN BREADBOARDING AND PCB LAYOUT

The DP8461/65 contains a high performance analog PLL and certain precautions must be taken when breadboarding or designing a PCB layout. The following guidelines should be adhered to when working with the DP8461/65:

1) Do not wire wrap.
2) Keep component lead lengths short, place components as close to pins as possible.This applies to R1, C1, R2, C_{VCO}, R_{RATE}, R_{BOOST}, C_{RATE}, C_{BOOST}, RPG1, RPG2, and CPG1.
3) Provide a good ground plane and use a liberal amount of supply bypassing. The quieter a PLL's environment, the happier it is.
4) Avoid routing any digital leads within the vicinity of the analog leads and components.
5) Keep inter-pin capacitance to a minimum; i.e., avoid running traces or planes between pins.
6) Minimize digital output pin capacitive loading to reduce current transients.

NSC has used a PC board approach to breadboarding the DP8461/65 that gives an excellent ground plane and keeps component lead lengths very short. With this setup very stable and reliable operation has been observed. Illustration of component layout is shown in *Figure 20*.

Applications of the DP8461/65 Data Separator (Continued)

ADDITIONAL NOTES

1. PG1 should be grounded to improve noise immunity.
2. 2F clock must be applied at all times; without the 2F clock, the pulse gate circuitry will not operate properly making it impossible to lock onto the incoming data stream.
3. The programming capacitor for the V_{CO} can be calculated as:

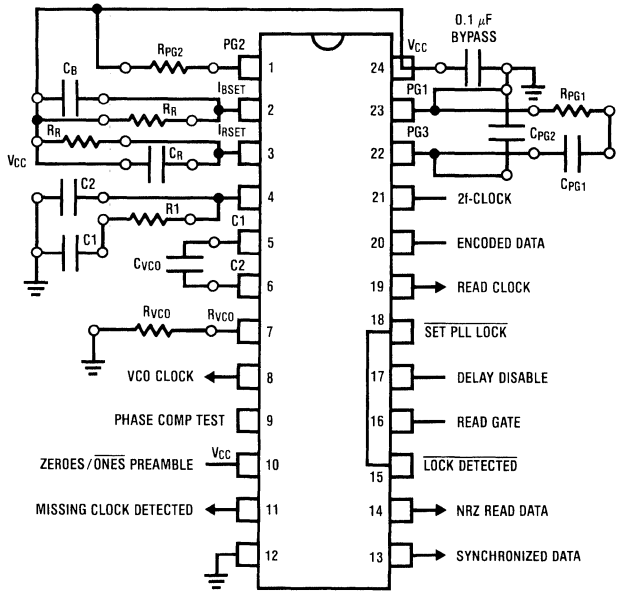
$$C_{VCO} = 1/(f_{VCO} \times R_{VCO}) - 5\ pF$$

The 5 pF value is due to parasitic and pin to pin capacitance. An additional accomodation must also be made for PC board capacitance.

4. Care must be taken in final PC board layout to minimize pin to pin capacitance, particularly in multi-layer printed circuit boards.
5. Please refer also to Precautions for Disk Data Separator Designs, NSC Application Note AN-414.

Connection Diagrams

DP8461/65



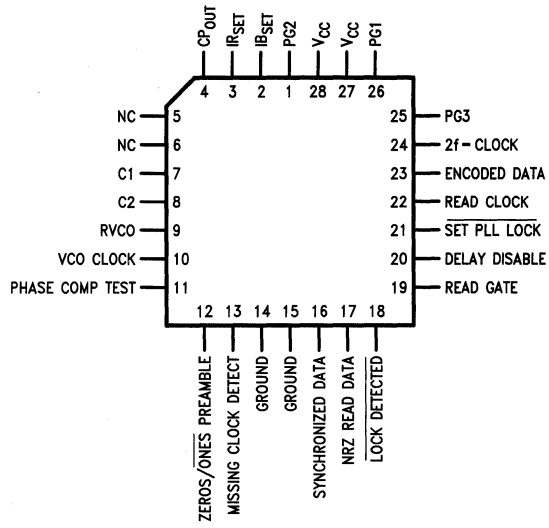
TL/F/8445–24

FIGURE 20. Recommended Component Layout

6

Connection Diagrams (Continued)

Plastic Chip Carrier



TL/F/8445–26

Order Number DP8461V or DP8465V
See NS Package Number V28A

National Semiconductor Corporation

DP8462 2,7 Code Data Synchronizer

General Description

The DP8462 Data Synchronizer is designed for application in disk drive memory systems employing Run Length Limited Codes using 1-0-0 or 1-0-0-0 preamble patterns, and depending on system requirements, may be located either in the drive or in the controller. It receives digital pulses from a pulse detector circuit (such as the DP8464 Disk Pulse Detector) if the DP8462 is situated in the drive, or from an interface if it is situated in the controller. In the read mode, the circuit locks onto and detects either a 100 or 1000 preamble pattern depending on the state of the pattern select input pin. The synchronized data and clock are then available for decoding and deserialization by a decoder circuit. All of the digital input and output signals are TTL compatible and only a single +5V supply is required. Although separate Analog and Digital V_{CC} and Ground pins are provided, they are expected to be tied together by the user. The chip is housed in a standard narrow 24-pin dual-in-line package (DIP) and is fabricated using Advanced Schottky bipolar analog and digital circuitry. This high speed I.C. process allows the chip to work with data rates up to 20 Mbits/sec. There are two versions of the chip, each having a different decode window error specification. These two versions (-3 and -4) will operate from 4 to 20 Mbits/sec, with respectively increasing window errors, as specified in the Electrical Characteristics Table.

The DP8462 features a phase-locked-loop (PLL) consisting of a pulse gate, phase comparator, charge pump, buffering amplifier, and voltage-controlled-oscillator (VCO). Pins are provided for the user to select the values of the external filtering components required for the pulse gate and PLL, the frequency setting components required for the VCO, two current setting resistors for the charge pump, and current setting resistors for the pulse gate that control the delay line.

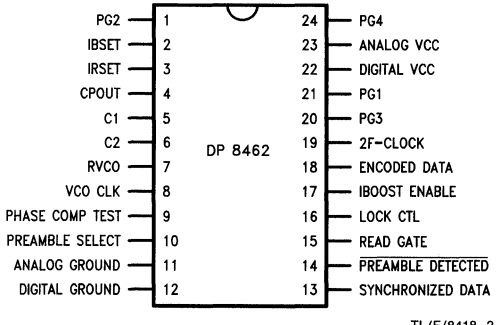
The on-board PLL's phase comparator has two modes of operation: phase and frequency comparison or phase only comparison. In the non-read mode, the comparator performs phase and frequency comparison, but once in the read mode, it switches to phase only comparison. The user selects whether this mode change occurs as soon as read mode is entered or after the preamble pattern is detected. The charge pump also has two modes of operation: high track rate—intended to be used in the non-read mode and in the read mode while acquiring lock, and low track rate—intended to be used in the read mode to retain lock. Both track rates are selected by the user with external components; the user is given control over when the track rate switch takes place.

Features

- Phase-Frequency PLL in non-read mode and during preamble if desired
- Operates at data rates up to 20 Mbit/sec
- Detects either 1-0-0 or 1-0-0-0 preamble patterns
- User determined PLL loop filter network
- PLL charge pump has two user-determined tracking rates
- External control of track-rate switchover
- External control of phase comparator switchover
- Delay line may be externally adjusted if desired
- ORed phase comparator outputs for monitoring bit-shift
- Standard narrow 24-pin DIP or 28-pin Plastic Chip Carrier package
- Less than ½W power consumption
- Single +5V supply

Connection Diagrams

Dual-In-Line Package

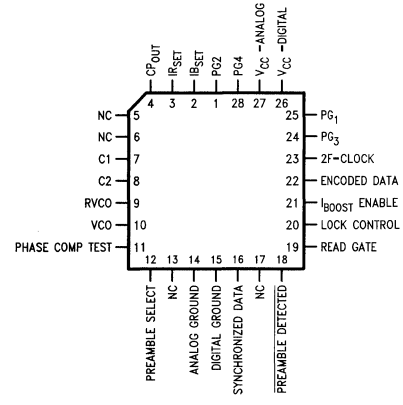


TL/F/8418-2

Top View

Order Number DP8462N
See NS Package N24C

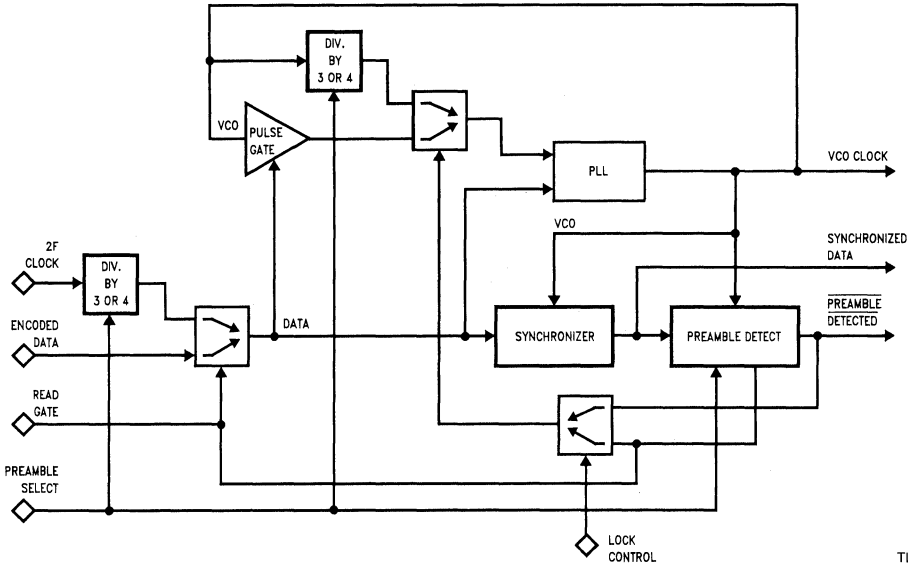
Plastic Chip Carrier



TL/F/8418-24

Order Number DP8462V
See NS Package V28A

Block Diagram



TL/F/8418–1

Pin Descriptions*

POWER SUPPLY

22,23 Digital and Analog $V_{CC} = +5V \pm 5\%$
Should be tied together and bypassed by user.

11,12 Analog and Digital Ground
Should be tied together by user.

TTL Level Logic Inputs

15 READ GATE: When asserted, this signal sets the DP8462 into the Read Mode. The PLL then begins to lock onto the encoded data.

10 PREAMBLE SELECT: A high level on this input enables the circuit to recognize a 1-0-0-0 pattern while a low level results in the recognition of a 1-0-0 pattern. Also, in the non-read mode, if 1-0-0 is selected VCO/3 will lock onto 2F/3 while if 1-0-0-0 is selected VCO/4 will lock onto 2F/4.

16 LOCK CTL: This input allows the user to determine when the circuit will switch from Phase-Frequency comparison to Phase only comparison once in the Read Mode. A low level on this pin causes the circuit to switch from the phase-frequency comparison mode as soon as READ GATE is asserted while a high level means that the circuit will switch after 4 bytes of preamble have been detected and $\overline{\text{PREAMBLE DETECTED}}$ output has been asserted. (See the Truth Table at the end of this Section.)

17 IBOOST ENABLE: This input allows the user to control the PLL's track rate by turning Iboost current on and off. A high level at this input causes Iboost to be added to Irate—placing the PLL in the high track rate. In a typical system IBOOST ENABLE may be tied to $\overline{\text{READ GATE}}$ or $\overline{\text{PREAMBLE DETECTED}}$.

18 ENCODED DATA: This input is for the incoming encoded data from the output of the head amplifier/pulse-detecting network located on the disk drive. Each positive edge of the ENCODED DATA waveform identifies a flux reversal on the disk.

19 2F-CLOCK: This is a system clock input, which is either a signal generated from the servo track, or a signal buffered from a crystal. It operates at twice the NRZ DATA rate. 2F-CLOCK MUST ALWAYS BE APPLIED TO THIS INPUT FOR PROPER OPERATION.

TTL Level Logic Outputs

8 VCO CLOCK: This is the output of the on-chip VCO, transmitted from an Advanced Schottky TTL buffer. It is synchronized to the SYNCHRONIZED DATA output so that it can be used by the encoder/decoder circuitry.

13 SYNCHRONIZED DATA: This is the same encoded data that is input to the chip, but is synchronous with the VCO CLOCK.

14 $\overline{\text{PREAMBLE DETECTED}}$: After READ GATE is asserted, this output goes low after detecting approximately 4 bytes of preamble and remains low until READ GATE goes inactive.

9 PHASE COMP TEST: This output is the logical "OR" of the Phase Comparator outputs, and may be used for the testing of the disk media.

Analog Signals

21,20 PG1, PG3: The external capacitors and resistor of the Pulse Gate filter are connected to these pins. PG1 should be tied directly to ground.

1 PG2: This is the Pulse Gate delay reference pin. The delay reference generator establishes a voltage at this pin; thereby producing the bias current for the Pulse Gate delay section in the resistor tied between this pin and V_{CC}.

*Pin Number Designations apply for the 24 Pin DIP. See Connection Diagram for the Plastic Chip Carrier Pin Designations.

Pin Descriptions (Continued)

24 PG4: This is the Pulse Gate delay control pin. This pin can be tied to the PG2 pin if the user desires to adhere to the chip's standard synchronization window specification; otherwise, it can be tied to PG2 through a "current splitting" network (see *Figure 6*) —thereby shifting the synchronization window early or late.

3 IRSET: The current into the rate set pin (Vbe/Rrate) is approximately half the charge pump output current for the low tracking rate.

2 IBSET: The current into the boost set pin (Vbe/Rboost) is approximately half the amount by which the charge pump current is increased for the high tracking rate.
(I_{hirate} = I_{rate} Set + I_{boost} Set).

4 CPOUT: This pin is the output node of the charge pump and also the noninverting input of the Buffer Amplifier. It is made available for connection of external filter components for the phase-locked-loop.

5,6 VCO C1, C2: An external capacitor connected across these pins sets the nominal frequency.

7 RVCO: The current into this pin determines the operating currents within the VCO.

Note: ANALOG and DIGITAL V_{CC} pins must be tied together by the user.
ANALOG and DIGITAL GND pins must also be tied together by the user.

Truth Table of Pulse-Gate's Modes

LOCK CTL (Pin 16)	READ GATE (Pin 15)	$\overline{\text{PREAMBLE}}$ DETECTED (Pin 14)	Pulse-Gate Comparison Mode	Comments
LO	LO	LO	N/A	N/A
LO	LO	HI	Phase and Frequency	Non-Read Mode
LO	HI	HI	Phase only	Read Mode
LO	HI	LO	Phase only	Read Mode
HI	LO	LO	N/A	N/A
HI	LO	HI	Phase and Frequency	Non-Read Mode
HI	HI	HI	Phase and Frequency	Read Mode
HI	HI	LO	Phase Only	Read Mode

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage	7V
TTL Inputs	7V
Output Voltages	7V
Input Current (CPOUT, IRSET, IBSET, RVCO)	2 mA
Storage Temperature	−65°C to +150°C
Operating Temperature Range	0°C to +70°C

Operating Conditions

Symbol	Parameter		Conditions	Min	Typ	Max	Units
V_{CC}	Supply Voltage			4.75	5.00	5.25	V
T_A	Ambient Temperature			0	25	70	°C
I_{OH}	High Logic Level Output Current		V_{CO} Clock Others			−2000 −400	μA
I_{OL}	Low Logic Level Output Current		V_{CO} Clock Others			20 8	mA
f_{DATA}	Input Data Rate			4.0		20	Mbit/sec
t_{WCK}	Width of 2f-CLOCK, High or Low			10			ns
t_{WPD}	Width of ENCODED DATA Pulse (Note 1)	High		18			ns
		Low		0.4t			ns
V_{IH}	High Logic Level Input Voltage			2			V
V_{IL}	Low Logic Level Input Voltage					0.8	V
t_{SU} Read Gate	Min Time Required for a Positive Edge of Read Gate to Occur Before a Negative Edge of VCO			20			ns
t_{HOLD} Read Gate	Min Time Required for a High Level on Read Gate to be Held After a Negative Edge of VCO			10			ns

Note 1: t is defined as the period of the NRZ data ($t = 2/F_{VCO}$).

DC Electrical Characteristics Over Recommended Operating Temperature Range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IC}	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max	V_{CC} −2V	V_{CC} −1.6V		V
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max			0.5	V
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V			20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V			−200	μA
I_O	Output Drive Current (Note 1)	V_{CC} = Max, V_O = 2.125V	−12		−110	mA
I_{CC}	Supply Current	V_{CC} = Max			85	mA
I_{OUT}	Charge Pump Output Current	$200 \le I_{RATE} + I_{BOOST} \le 2000$	0.9 I_{TYP} −25	2.0 (I_{RATE} + I_{BOOST})	1.1 I_{TYP} +25	μA

Note 1: This value has been chosen to produce a current that closely approximates one-half of the true short-circuit output current, I_{OS}.

AC Electrical Characteristics Over Recommended V_{CC} and Operating Temperature Range

(All Parts unless stated otherwise) ($t_R = t_F$ = 2.0 ns, V_{IH} = 3.0V, V_{IL} = 0V) (Note 1)

Symbol	Parameter	Min	Typ	Max	Units
t_{READ}	Positive VCO CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE high)		1	1	—
$t_{TRANSMIT}$	Positive VCO CLOCK transitions required to transmit input encoded data to output	1	2	3	—
$t_{READ\ ABORT}$	Number of VCO CLOCK cycles after READ GATE set low to read operation abort (Note 3)			2	T-clock
t_{WINDOW}	Variance of center of decode window from nominal (Note 6) DP8462-3 / DP8462-4			6 / 10	ns
ϕLINEARITY	Phase range for charge pump output linearity (Note 2)	$-\pi$		$+\pi$	Radians
K_1	Phase comparator—Charge Pump gain constant (N = f_{VCO}/f input data) (Note 4)		$\frac{1.78\ V_{BE}}{N2\pi R}$		Amps/rad
$V_{CONTROL}$	Charge pump output voltage swing from nominal		±100		mV
K_{VCO} (= A × K_2)	VCO gain constant (ω_{VCO} = VCO center frequency in rad/s) (Note 5)	$\frac{1.20\ \omega_C}{V_{BE}}$	$\frac{1.40\ \omega_C}{V_{BE}}$	$\frac{1.60\ \omega_C}{V_{BE}}$	rad/sec V
f_{VCO}	VCO center frequency variation over temperature and V_{CC}	−2		+2	%
$f_{MAX\ VCO}$	VCO maximum frequency		60		MHz
t_{PHL}	Propagation delay from VCO negative edge to synchronous DATA negative edge	2		18	ns
t_{PLH}	Propagation delay from VCO negative edge to synchronous DATA positive edge	4		20	ns

Note 1: A sample calculation of frequency variation vs. control voltage: V_{IN} = ±0.1V;

$$K_{VCO} = \frac{\omega OUT}{V_{IN}} = \frac{0.4\ \omega_C}{0.2V} = \frac{2.0\ \omega_C}{V}\ \frac{(rad/sec)}{(volt)}.$$

Note 2: $-\pi$ to $+\pi$ with respect to 2f VCO CLOCK.

Note 3: T-clock is defined as the time required for one period of the VCO CLOCK to occur.

Note 4: With respect to VCO CLOCK; $I_{PUMP\ OUT} = 1.9\ I_{SET}$

$$I_{SET} = \frac{V_{BE}}{R_{SET}}$$

Note 5: Although specified as the VCO gain constant, this is the gain from the Buffer Amplifier input to the VCO output.

Note 6: This specification is guaranteed only for the conditions under which the parts were tested. However, significant variation from formula is not expected for other data rates and filters. This specification is for the condition when PG2 and PG4 are tied together. External adjustment can be used to optimize t_{WINDOW} as described in the pulse gate section. The filter values below were chosen for operation in an automatic test system (static window) environment. Different criteria may apply for choosing filter values in a disk system. See Loop Filter Section for sample calculations of other filter values.

Part Type	Data Rate Tested	Filter				
		C_1	C_2	R_1	R_{RATE}	R_{BOOST}
DP8462-4	5 Mbit/sec	0.03 μF	600 pF	100Ω	820Ω	1.5 kΩ
DP8462-3	10 Mbit/sec	0.022 μF	510 pF	81Ω	800 kΩ	1.8 kΩ

Note: For further informtion refer to Application Note AN-414

External Component Selection (All Parts) (Note 1)

Symbol	Component	Min	Typ	Max	Units
R_{VCO}	VCO Frequency Setting Resistor (Note 2)	990		1010	Ω
C_{VCO}	VCO Frequency Setting Capacitor (Notes 3,4)	20		120	pF
R_{RATE}	Charge Pump I_{RATE} Set Resistor (Note 6)	0.4		4.0	kΩ
R_{BOOST}	Charge Pump (High Rate) I_{BOOST} Resistor (Note 6)	0.5		∞	kΩ
C_R	I_{RATE} Bypass Capacitor (Note 5)	0.01			μF
C_B	I_{BOOST} Bypass Capacitor (Note 5)	0.01			μF

Note 1: External component values for the Loop Filter and Pulse Gate are given in Table II and Table I respectively.

Note 2: A 1% Component Tolerance is Required.

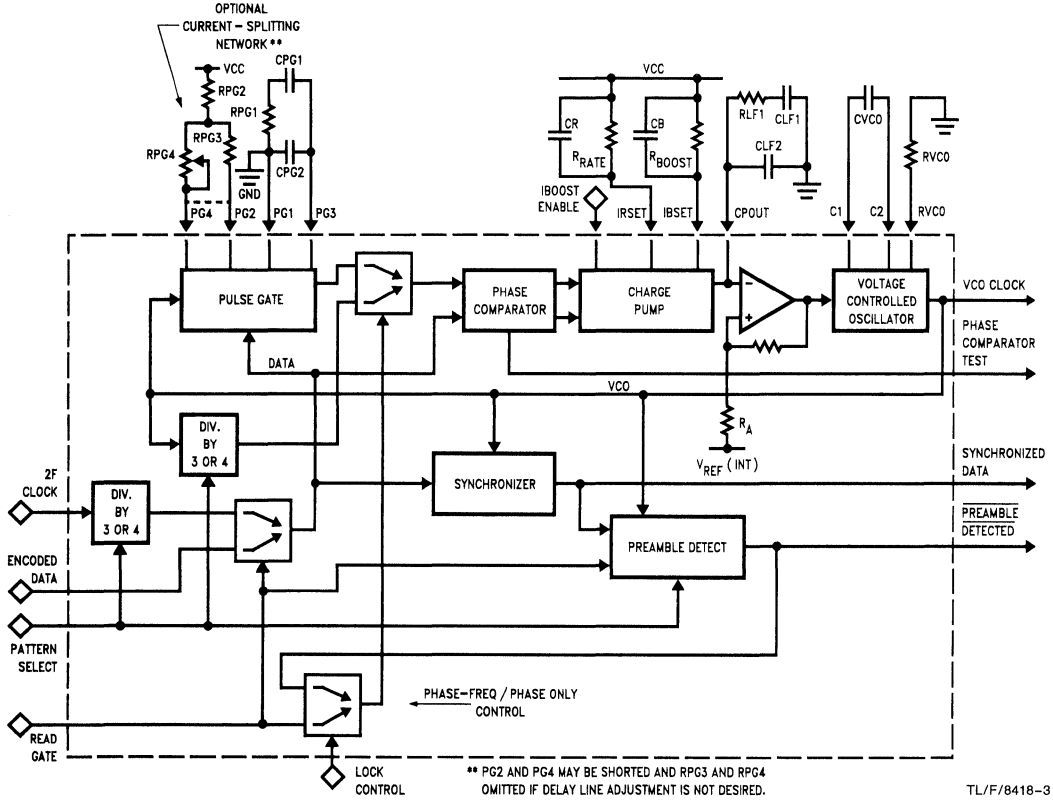
Note 3: These MIN and MAX values correspond to the MAX and MIN data rates respectively.

Note 4: The Component Tolerance is system dependent on how much center frequency deviation can be tolerated.

Note 5: Component Tolerance 15%.

Note 6: The minimum value of the parallel combination of R_{RATE} and R_{BOOST} is 350Ω.

Detailed Block Diagram



Circuit Operation

In the non-read mode, the DP8462 Data Separator remains locked to the 2f CLOCK signal divided by 3 or 4 (depending upon the preamble used) in anticipation of a preamble when read mode is entered. When the READ GATE input goes high, the DP8462 enters the read mode after 1 VCO CLOCK cycle. Referring to *Figure 1*, once in the read mode, the PLL reference signal is switched from the 2f-divided-by-3-or-4 signal to the ENCODED DATA input. The PLL is at this point in the high-tracking rate mode and also in the Phase and Frequency Comparison mode. The PLL then attempts to

Circuit Operation (Continued)

quickly lock onto the incoming ENCODED DATA stream and starts looking for 16 consecutive preamble pulses—chosen by the user to be either 100 (PATTERN SELECT: LO) or 1000 (PATTERN SELECT: HI). If the user has chosen to switch to Phase Only Comparison as soon as read operation begins (LOCK CTL: LO), then the Phase Comparator will start to compare ENCODED DATA with VCO-gated-by-DATA immediately (see *Figure 2*); otherwise, it will keep comparing ENCODED DATA with VCO divided by 3 or 4—i.e., remain in Phase and Frequency Comparison mode until after 16 consecutive preamble pulses have been detected. At this time, $\overline{\text{PREAMBLE DETECTED}}$ output goes low and the circuit now starts to compare ENCODED DATA with VCO-gated-by-DATA (see *Figure 1*).

The user is given control over when to switch the charge-pump current rate through the use of the IBOOST ENABLE input. One way the user can accomplish this is by tying $\overline{\text{PREAMBLE DETECTED}}$ output to the IBOOST ENABLE input directly. Thus, once $\overline{\text{PREAMBLE DETECTED}}$ is asserted, the circuit will go into low track rate and Phase Only Comparison mode (if LOCK CTL: HI) so that a more stable lock can be retained. The incoming ENCODED DATA stream is now synchronized with the VCO CLOCK and appears at the SYNCHRONIZED DATA output (see *Figure 3*). (If the user wishes to switch to low track rate as soon as the circuit enters the read mode, then the READ GATE signal should be inverted and applied to the IBOOST ENABLE input).

Figure 4 shows the sequence when READ GATE goes low, signifying the end of a read operation. The PLL reference signal is switched back to 2f divided by 3 or 4 input and the $\overline{\text{PREAMBLE DETECTED}}$ output goes high (causing the charge-pump to go to the high tracking rate, if $\overline{\text{PREAMBLE DETECTED}}$ is tied to the IBOOST ENABLE input). Also, the Phase Comparator goes back to Phase and Frequency Comparison mode and the circuit attempts to lock onto the 2f divided by 3 or 4 signal, thus returning to the initial conditions.

CIRCUIT DESCRIPTION

1. Divide by 3 or 4: Depending on the preamble pattern being used, these circuits divide 2f CLOCK and internal VCO CLOCK signals by 3 or 4. During the non-read mode, the VCO remains phase and frequency locked to these divided signals so that when read mode is entered, the PLL can quickly acquire lock because the data stream that consists of the preamble pattern is very close in frequency to the VCO divided by 3 or 4.

2. Pulse Gate: Once in the read mode, the PLL has to lock the VCO CLOCK to the ENCODED DATA stream; outside of the preamble, however, the data signal is not cyclic like the VCO CLOCK and therefore cannot be frequency compared to the VCO. It is for this reason that the Pulse Gate is used to allow a reference signal from the VCO into the Phase Comparator only when an ENCODED DATA bit is valid. The Pulse Gate also utilizes a scheme which delays the incoming data by one-half the period of the 2f-CLOCK. This optimizes the position of the decode window and allows input jitter up to ± half the 2f-CLOCK period, assuming no error in the decode window position. The decode window error can be determined from the specification in the Electrical Characteristics Table.

3. Multiplexers at the Phase Comparator's inputs: These multiplexers are used to determine which signals the Phase Comparator will compare during different modes of operation. Either 2F divided by 3 or 4 or ENCODED DATA is compared with either VCO divided by 3 or 4 (Phase and Frequency Lock) or with VCO gated by DATA (Phase Only Lock).

4. Phase Comparator: The Phase Comparator receives its inputs from the Multiplexers mentioned above, and is edge-triggered from these inputs to provide charge-up and charge-down outputs.

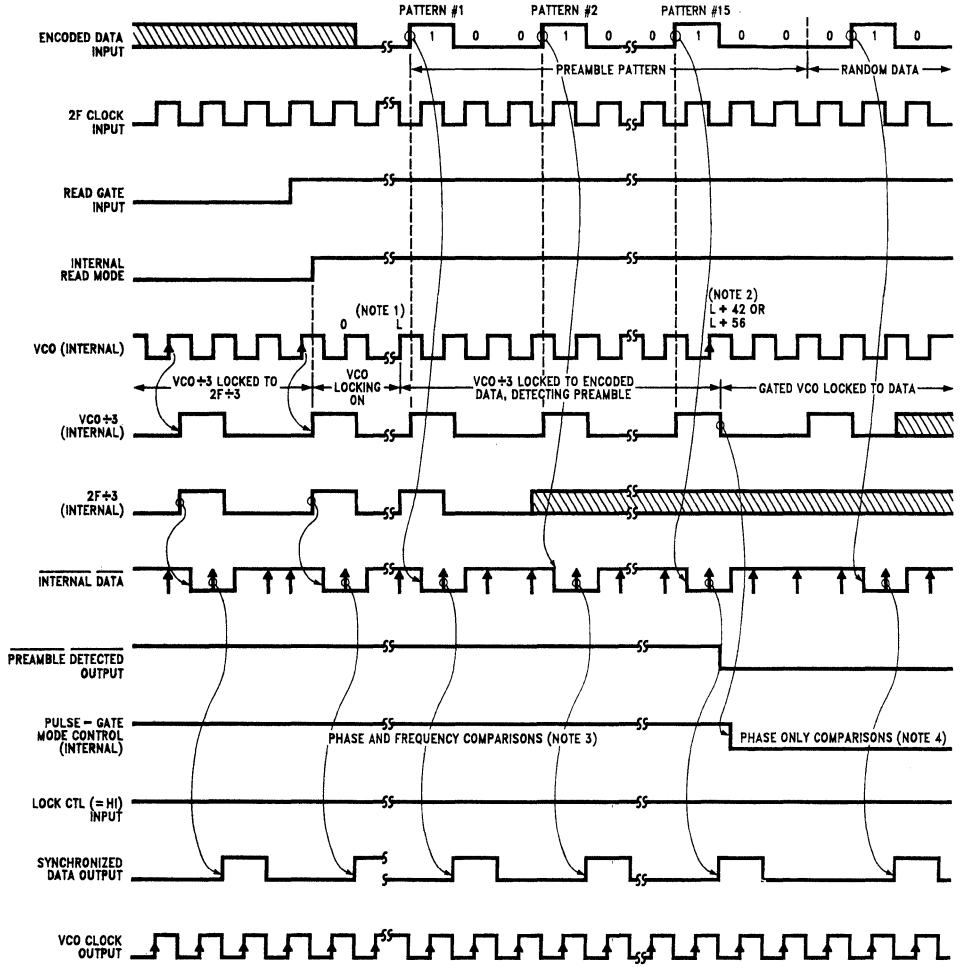
5. Charge Pump: The high speed charge pump consists of a switchable constant current source and sink. The charge pump constant current is set by connecting external resistors to V_{CC} from IRSET and IBSET pins. With IBOOST ENABLE HIGH, the PLL is in the high tracking rate and both resistors determine the current. With IBOOST ENABLE LOW, the PLL is in the low tracking rate and only the IRSET resistor determines the charge pump current. The output of the charge pump feeds into external filter components and the Buffer Amplifier. Thus, through the use of the IBOOST ENABLE pin, the user can determine when the circuit switches track rates.

6. Buffer Amplifier: The Buffer Amplifier is configured as a high input impedance amplifier which is inserted between the charge pump and the VCO, thus allowing connection of external PLL filter components to the charge pump output pin CPOUT. The output of the Buffer Amplifier is internally connected to the VCO control input.

7. VCO: The Voltage-Controlled-Oscillator requires a resistor from the RVCO pin to ground and a capacitor between pins C1 and C2, to set the center frequency. The VCO frequency can be varied from nominal by approximately ±20%, as determined by its control input voltage (CPOUT).

8. Preamble Pattern Detector: Two types of preamble patterns are commonly used in RLL 2,7 code disk systems—1-0-0 and 1-0-0-0. The user selects the preamble pattern to be used by setting PREAMBLE SELECT input either HI for the 1-0-0-0 pattern or LO for the 1-0-0 pattern. The DP8462 divides 2F Clock and VCO Clock signals by 3 or 4 depending upon whether 1-0-0 or 1-0-0-0 pattern is selected, respectively, and remains locked to this divided pattern in anticipation of a preamble. Once the chip is in the read mode, the VCO proceeds to lock onto the incoming data stream. The Preamble Pattern Detector then searches for 16 consecutive patterns (i.e., 100100100... or 100010001000...) to indicate lock has been achieved. The $\overline{\text{PREAMBLE DETECTED}}$ output then goes low. Any deviation from the above-mentioned continuous stream of patterns before 16 of these are detected will reset the Pattern Detector and the procedure will then start over again.

Circuit Operations (Continued)

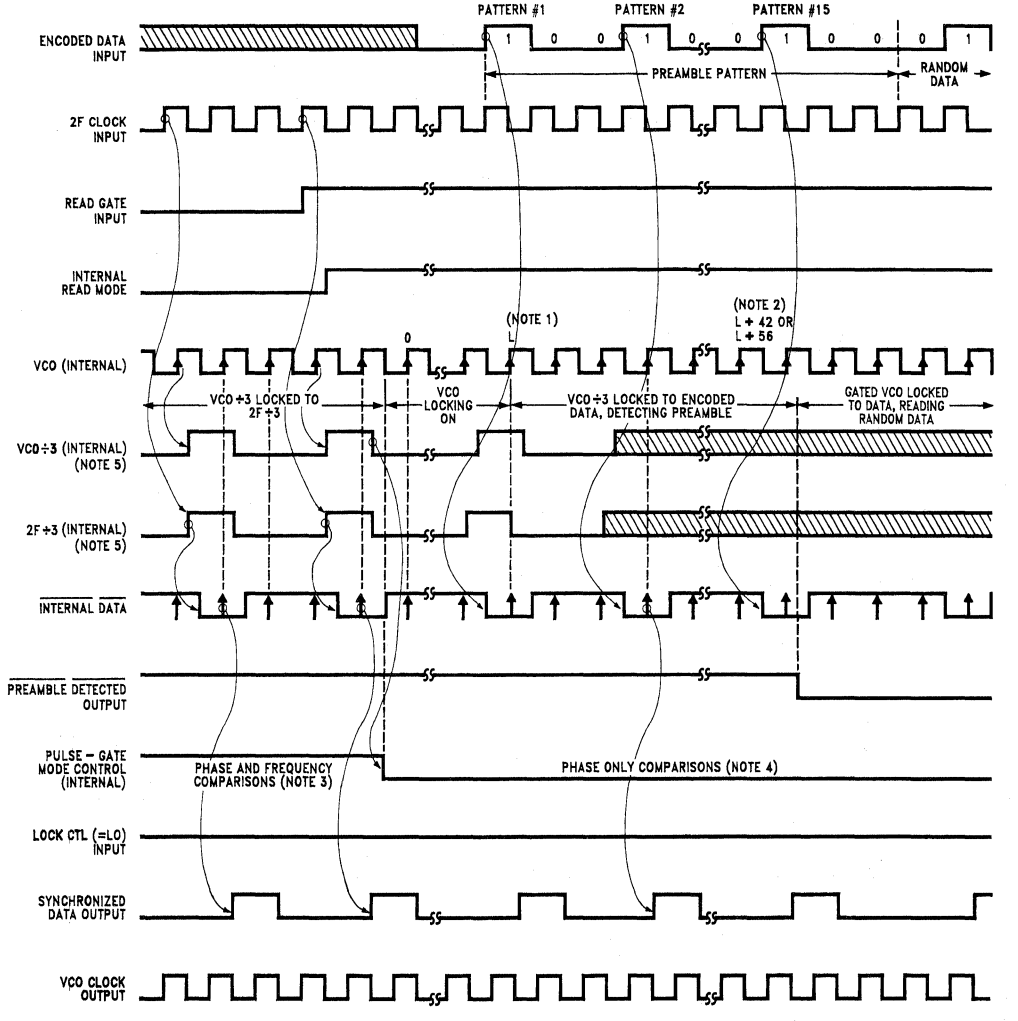


TL/F/8418–4

Note 1: L = Number of VCO cycles required for VCO to lock—typically 20 but determined by external component value.
Note 2: At L + 42 (Pattern = 1-0-0) or L + 56 (Pattern = 1-0-0-0), 15 patterns have been detected.
Note 3: VCO ÷3 (or 4) being compared with 2F ÷3 (or 4) in the non-read mode and Preamble in the Read Mode.
Note 4: VCO GATED BY DATA being compared with ENCODED DATA.
Note 5: PREAMBLE SELECT = LO; 100 pattern selected—so 2F & VCO are being divided by 3.

FIGURE 1. Lock-On Sequence Waveform Diagram—Pulse Gate Mode Switches after Preamble Detection

Circuit Operation (Continued)

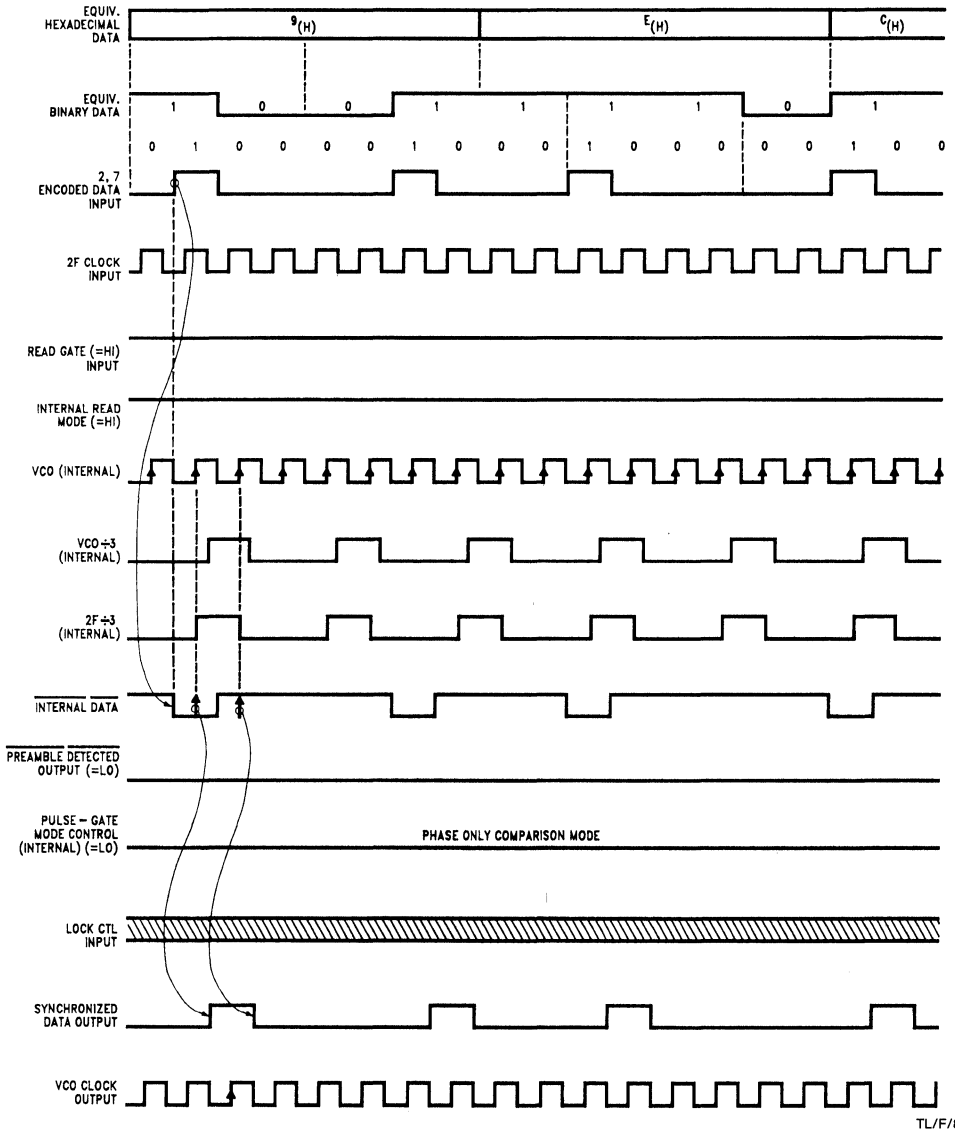


TL/F/8418–5

Note 1: L = Number of VCO cycles required for VCO to lock—typically 20 but determined by external component value.
Note 2: At L + 42 (Pattern = 1-0-0) or L + 56 (Pattern = 1-0-0-0), 15 patterns have been detected.
Note 3: VCO ÷ 3 (or 4) being compared with 2F ÷ 3 (or 4) in the non-read mode.
Note 4: VCO gated by DATA being compared with ENCODED DATA.
Note 5: PREAMBLE SELECT = LO; 1-0-0 pattern selected—so 2F & VCO are being divided by 3.

FIGURE 2. Lock-On Sequence Waveform Diagram—Pulse Gate Mode Switches Immediately After READ GATE is Asserted

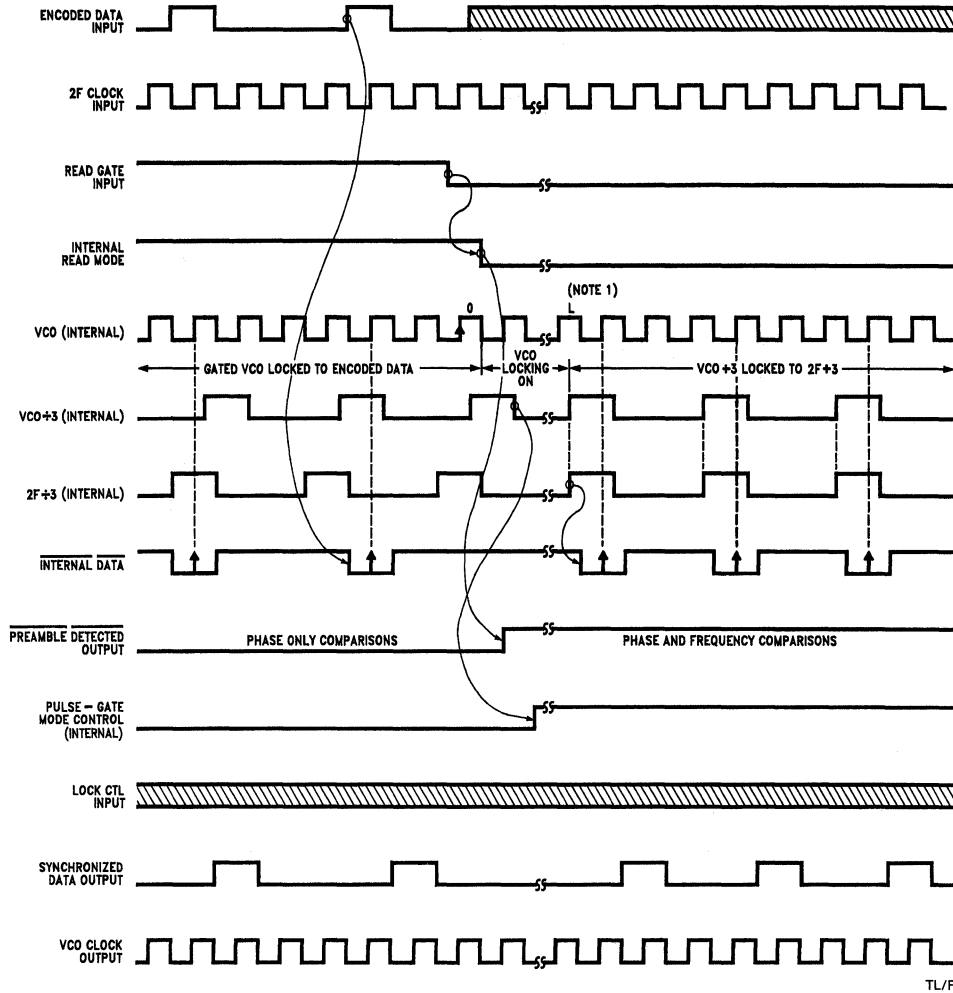
Circuit Operation (Continued)



TL/F/8418–6

FIGURE 3. Locked-On Waveform Diagram

Circuit Operation (Continued)



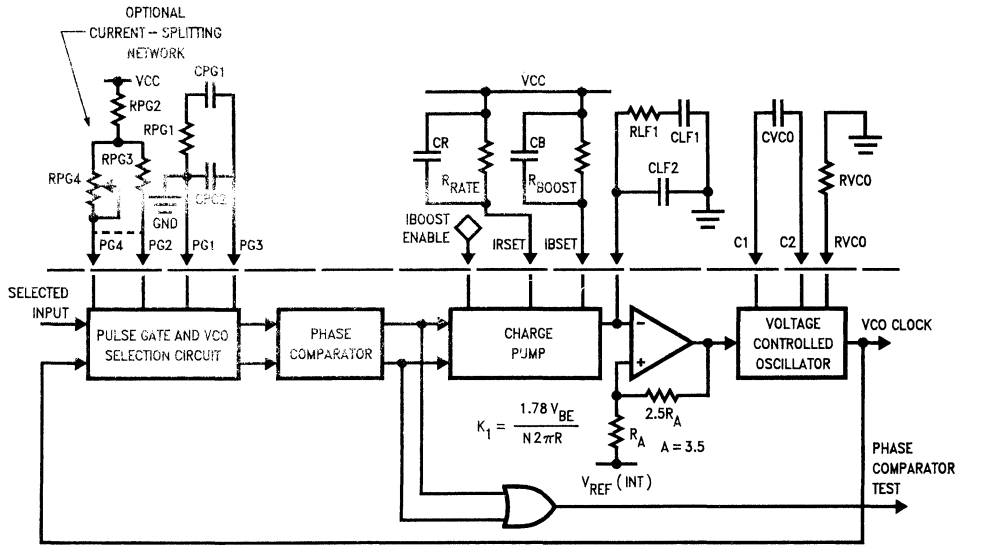
TL/F/8418–7

Note 1: L indicates the number of cycles required for the VCO to lock to the 2f-Clock.

Note 2: PREAMBLE SELECT = LO; 1-0-0 Pattern selected—so 2F & VCO being divided by 3.

FIGURE 4. Lock-Ending Sequence Waveform Diagram

Circuit Operation (Continued)



TL/F/8418–8

FIGURE 5. Phase-Locked-Loop Section

BIT JITTER TOLERANCE

The two options of the DP8462, the -4 and -3 offer decreasing window errors (respectively) so that the parts may be selected for different data rates (up to 20 Mbit/sec). The -4 part will be used in most low data rate applications. As an example, at the 5 Mbit/sec data rate of most 5¼ inch drives, T = 200 ns so that from the Electrical Characteristics Table, t_{WINDOW} = 10 ns. The chip therefore contributes up to 10 ns of window error, out of the total allowable error of 50 ns (half the 2f-clock period of 100 ns). This allows the disk drive to have a margin of 40 ns of jitter on the transition position before an error will occur. The bit jitter tolerance can be improved by adjusting the window center using PG2 and PG4. A current splitting network consisting of RPG3 and RPG4 can be used to adjust the delay line. This adjustment is internally compensated for V_{CC} and temperature variation.

ANALOG CONNECTIONS TO THE DP8462

External passive components are required for the Pulse Gate, Charge Pump, Loop Filter and VCO as shown in *Figure 5*. The information provided here is for guidelines only. The user should select values according to his own system requirements. Phase-Locked Loops are complex circuits that require detailed knowledge of the specific system. Factors such as loop gain, stability, response to change of signal, lock-on time, etc. are all determined by the external components. In many disk systems these factors are critical, and National Semiconductor recommends the designer be knowledgeable of phase-locked-loops, or seek the advice of an expert. Inaccurate design will probably result in excessive disk error rates. The phase-locked-loop in the DP8462 has many advantages over all but the most sophisticated discrete designs, and if the component values are selected correctly, it will offer significant performance advantages. This should result in a reduction of disk error rates over equivalent discrete designs.

PULSE GATE

There are 6 external components connected to the Pulse Gate as shown in *Figure 6* with the associated internal components. Of these, RPG3 and RPG4 are optional and may be omitted if adjustment of the delay line is not desired. The values of RPG1, RPG2, RPG3, RPG4, CPG1, and CPG2 are dependent on the data rate. RPG1 and RPG2 are inversely proportional to the data rate, while CPG1 and CPG2 are proportional. Table I shows component values for the data rates given. Component values are calculated by selecting RPG2 from Table I [RPG2' = RPG2 + (RPG3//RPG4)]. Next calculate

$$CPG1 = \left(\frac{2.12 \times 10^5}{890 + RPG2'}\right)\left(\frac{1}{100 \times Rs}\right)^2$$

$$CPG2 = \frac{1}{10} CPG1, \text{ and}$$

$$RPG1 = \left(\frac{890 + RPG2'}{2.38 \times 10^5}\right)(100 \times Rs).$$

In the above equation Rs is the rotational speed and, for 3600 RPM, Rs = 60 Hz. A rotational speed of 3600 RPM was assumed for the calculations in Table I. For data rates not listed, RPG2 may be approximated as (30 kΩ/fDATA) − 1.2 kΩ = RPG2 where fDATA is the data rate in Megabits per second. RPG3 and RPG4, in conjuction with RPG2, form a "current-splitting-network" that can be used to adjust the delay line; thus adjusting the decode window early or late. RPG2 should be made large with respect to RPG3 and RPG4 and a potentiometer can be used for RPG4—with its value centered around that of RPG3. For example, at Data Rate = 5 Mbits/sec., Table I dictates that RPG2' should be 4.7k. If the delay line is to be made adjustable, then one could pick RPG2 = 4.3k and RPG3 = 800Ω. Now, using a 1.6 kΩ potentiometer for RPG4, RPG4 = 800Ω would give RPG2' = 4.7 kΩ and would provide standard window synchronization; varying RPG4 high or low, however, would

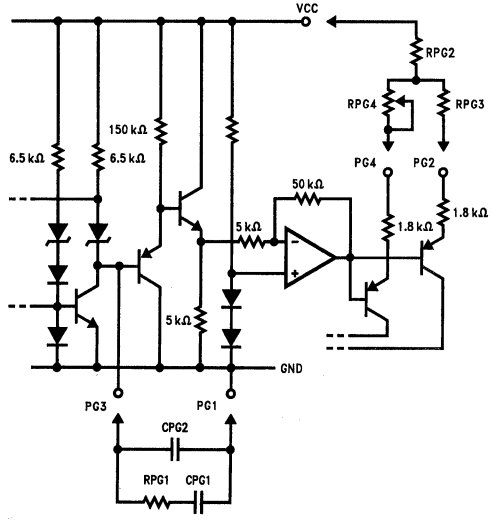
Circuit Operation (Continued)

shift the window late or early, respectively. If no adjustment is desired, then PG2 and PG4 should be tied together and only RPG2 should be used. Components with 5% tolerance will suffice.

TABLE I. Pulse Gate Component Selection Chart

Data Rate	RPG2'	RPG1	CPG1	CPG2
5 Mbit/sec	4.7 kΩ	150Ω	1 μF	0.1 μF
10 Mbit/sec	1.8 kΩ	68Ω	2.2 μF	0.22 μF
15 Mbit/sec	750Ω	39Ω	3.9 μF	0.39 μF

Where [RPG2' = RPG2 + RPG3//RPG4]



TL/F/8418–9

FIGURE 6. Pulse-Gate Controls

CHARGE PUMP

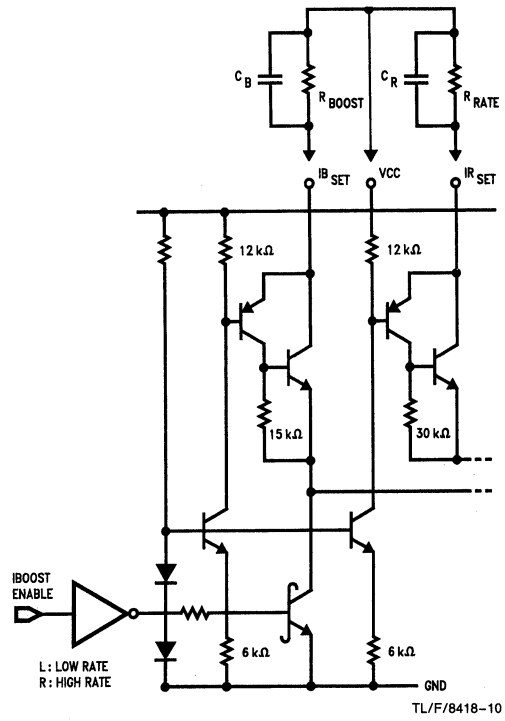
Resistors R_{RATE} and R_{BOOST} determine the charge pump current. The Charge Pump bidirectional output current is related to the input current according to the relationship specified in the DC Electrical Characteristics Table. In the high tracking rate with IBOOST ENABLE high, the input current is $I_{BSET} + I_{RSET}$, i.e., the sum of the currents through R_{BOOST} and R_{RATE} from V_{CC}. In the low tracking rate, with IBOOST ENABLE low, this input current is I_{RSET} only.

A recommended approach would be to select R_{RATE} first. The External Component Limits table allows R_{RATE} to be 0.4 kΩ to 4.0 kΩ, so for simplicity select $R_{RATE} = 820\Omega$. A typical loop gain change of 2:1 for high to low tracking rate would require $R_{BOOST} = R_{RATE}$ or 820Ω. Referring to *Figure 7* the input current is effectively V_{BE}/R_{RATE} in the low tracking rate, where V_{BE} is an internal voltage. This means that the current into or out of the loop filter is approximately 2.0 V_{BE}/R_{RATE}, or in this example approximately 1.8 mA. Note that although it would seem the overall gain is dependent on V_{BE}, this is not the case. The VCO gain is altered internally by an amount inversely proportional to V_{BE}, as detailed in the section on the Loop Filter. This means that as V_{BE} varies with temperature or device spread, the gain will remain constant for a particular fixed set of values of R_{RATE} and R_{BOOST}. This alleviates the need for potentiometers to select values for each device. The tolerance required for these two resistors will depend on the total loop gain tolerance allowed, but 5% would be typical. Also V_{CC} by-pass capacitors are required for these two resistors. A value of 0.01 μF is suitable for each.

VCO

The value of R_{VCO} is fixed at 1 kΩ ± 1% in the External Component Limits table. *Figure 8* shows how R_{VCO} is connected to the internal components of the chip. This value was fixed at 1 kΩ to set the VCO operating current such that optimum performance of the VCO is obtained for production device spreads. This means fixed value components will be adequate to set the VCO center frequency for production runs. The value of C_{VCO} can therefore be determined from the VCO frequency f_{VCO}, using the equation: $C_{VCO} = [1/(R_{VCO})(f_{VCO})] - 5$ pF where f_{VCO} is twice the input data rate. As an example, for a 5 Mbit/sec data rate, $f_{VCO} = 10$ MHz, requiring that $C_{VCO} = 95$ pF. This does not take into account any lead capacitance on the printed circuit board; the user must account for this. The amount of tolerance a particular design can afford on the center frequency will determine the capacitor tolerance. The capacitor is connected to the internal circuitry of the chip as shown in *Figure 9*.

As the data rate increases and C_{VCO} gets smaller, the effects of unwanted parasitic capacitances influence the fre-



TL/F/8418–10

FIGURE 7. I_{RATE} Set and I_{BOOST} Set

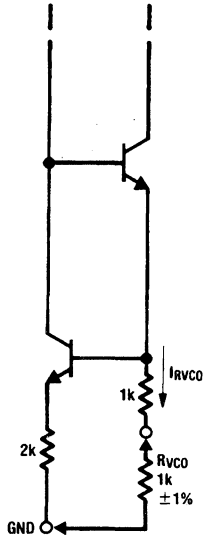
Circuit Operation (Continued)

quency. As a guide the graph of *Figure 10* shows approximately the value of C_{VCO} for a given data rate.

The VCO center frequency may be determined by: 1) holding pin 4 at ground potential and measuring the VCO frequency (−20% value); 2) holding pin 4 at approximately 3 volts and measuring the VCO frequency (+20% value); 3) averaging the two measured frequencies for the equivalent center frequency.

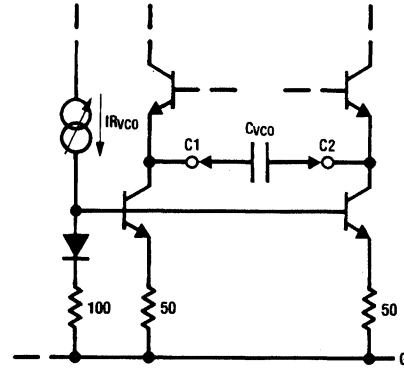
LOOP FILTER

The input current into the Buffer Amplifier is offset by a matched current out of the Charge Pump, and even so is much less than the switching current in or out of the Charge Pump. It can therefore be assumed that all the Charge Pump switching current goes into the Loop Filter components R_1 and C_1 and C_2. The tolerance of these compo-



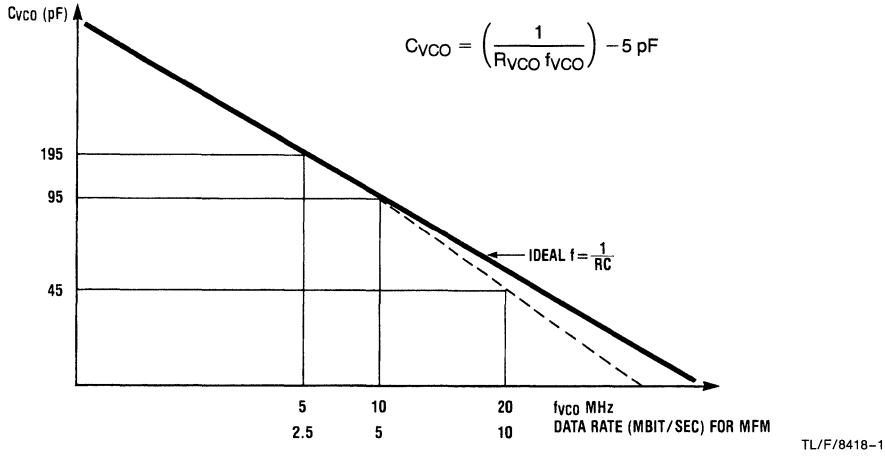
TL/F/8418–11

FIGURE 8. VCO Current Setting Resistor



TL/F/8418–12

FIGURE 9. VCO Capacitor



TL/F/8418–13

FIGURE 10. VCO Capacitor Value for Disk Data Rates

Circuit Operation (Continued)

nents should be the same as R_{RATE} and R_{BOOST}, and will determine the overall loop gain variation. The three components connected to the Charge Pump output are shown in *Figure 11*. Note the return current goes to analog GND, which should be electrically very close to the GND pin itself.

The value of capacitor C_1 determines loop bandwidth—the larger the value the longer the loop takes to respond to an input change. If C_1 is too small, the loop will track any jitter on the ENCODED DATA input and the VCO output will follow this jitter, which is undesirable. The value of C_1 should therefore be large enough so that the PLL is fairly immune to phase jitter but not large enough that the loop won't respond to longer term data rate changes that occur on the disk drive.

The damping resistor R_1 is required to damp any oscillation on the VCO input that would otherwise occur due to step function changes on the input. A value of R_1 that would give a phase margin of around 45 degrees would be a reasonable starting point.

The main function of the capacitor C_2 is to "smooth" the VCO input voltage. Typically its value will be less than one tenth of C_1.

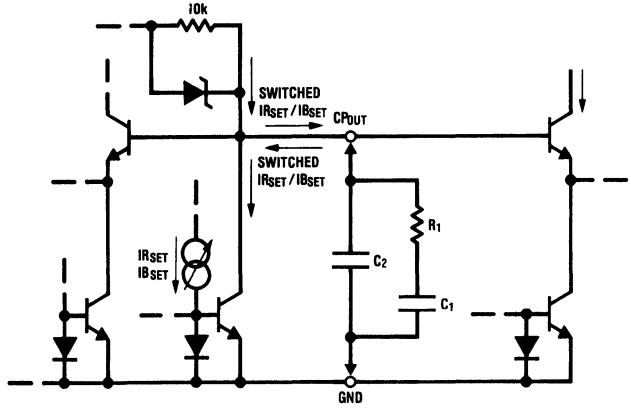
Figure 12 shows the relevant phase-locked-loop blocks that determine system response, namely the Phase Detector, Filter/Buffer Amplifier, and VCO. The Phase Detector (Phase Comparator and Charge Pump) produces an aggregate output current i which is proportional to the phase difference between the input signal and the VCO signal. The constant (K_1) is $\frac{1.78\ V_{BE}}{2\pi RN}$ amps per radian where $N = \frac{f_{VCO}}{f_{DATA}}$. R is either R_{RATE} or $R_{RATE} \parallel R_{BOOST}$. This aggregate current feeds into or out of the filter impedance (Z), producing a voltage to the VCO that regulates the VCO frequency. The VCO gain constant is 0.4 ω_{VCO}/V_{BE} radians per second per volt. Under steady state conditions, i will be zero and there will be no phase difference between the input signal and the VCO. Any change of input signal will produce a change in VCO frequency that is determined by the loop gain equation. This equation is determined from the gain constants K_1, A and K_2 and the filter v/i response.

The impedance Z of the filter is:

$$\frac{1}{sC_2} \parallel \left(\frac{1}{sC_1} + R_1\right) = \frac{1 + sC_1R_1}{sC_1\left(1 + \frac{C_2}{C_1} + sC_2R_1\right)}$$

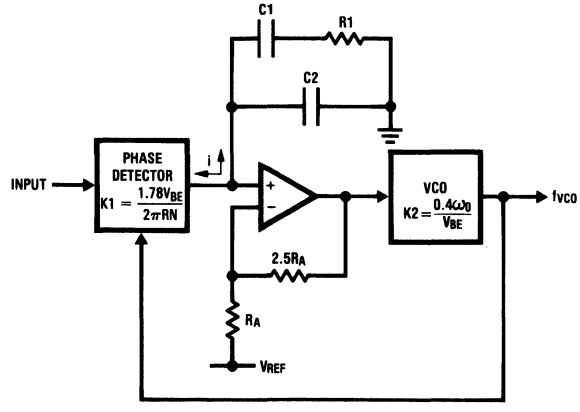
If $C_2 < C_1$ then the impedance Z approximates to:

$$\frac{1 + sC_1R_1}{sC_1\,(1 + sC_2R_1)}$$



TL/F/8418–14

FIGURE 11. Charge Pump Out



TL/F/8418–15

FIGURE 12. Loop Response Components

Circuit Operation (Continued)

The overall loop gain is then

$$G(s) = \frac{K_1 A K_2}{s} \times \frac{1 + sC_1R_1}{sC_1(1 + sC_2R_1)}$$

Let $G_{(K)} = K_1 A K_2$

$$F(s) = \frac{1 + SC_1R_1}{SC_1(1 + SC_2R_1)}$$

The Overall Closed Loop Gain is:

$$\frac{\phi OUT}{\phi IN} = \frac{G_{(K)} F(s)}{s + G_{(K)} F(s)}$$

Substituting, We Get

$$\frac{\phi OUT}{\phi IN} = \frac{G_{(K)}(SC_1R_1 + 1)}{S^3 R_1 C_1 C_2 + S^2 C_1 + GK(SC_1R_1 + 1)}$$

$$= \frac{(G_{(K)}/C_1)(SR_1C_1 + 1)}{S^3 R_1C_2 + S^2 + SG_{(K)}R_1 + G_{(K)}/C_1}$$

If $C_2 \ll C_1$, we can ignore the 3rd Order Component introduced by C_2 then:

$$\frac{\phi OUT}{\phi IN} = \frac{(G_{(K)}/C_1)(SR_1C_1 + 1)}{S^2 + SG_{(K)}R_1 + G_{(K)}/C_1}$$

This is a second Order Loop and can be solved as follows:

$$S^2 + SG_{(K)}R_1 + G_{(K)}/C_1 = S^2 + 2\zeta\omega_n S + \omega_n^2$$

$$\therefore C_1 = \frac{G_{(K)}}{\omega_n^2}$$

$$R_1 = \frac{2\zeta\omega_n}{G_{(K)}}$$

From the above equations:

$$\omega = (G(K)/C1)^{1/2}$$

$$G(K) = K1 \times A \times K2 =$$

$$[(0.89 \times V_{BE})/(2 \times Pi \times R)] \times [(0.4 \times W_{VCO})/V_{BE}] \times [3.5]$$

2,7 coded data has a 2.67 to 1.0 frequency range within the data field. The expression $K = (0.89 \times V_{BE}/2 \times Pi \times R)$ is valid when the VCO frequency is twice the ENCODED DATA frequency. In order to make this equation more general, it may be written as follows: $K = (1.78 \times V_{BE})/(2 \times Pi \times R \times N)$ where N is defined as the VCO frequency divided by the encoded data pulse frequency, or $N = F_{VCO}/F_{DATA}$ ($N = 3$ for maximum data rate and $N = 8$ for minimum data rate). Now G(K) can be written as follows:

$$G(K) = [(1.78 \times V_{BE})/(2 \times Pi \times R \times N)] \times [(0.4 \times \omega_{VCO})/V_{BE}] \times [3.5]$$

$$= (2.5 \times F_{VCO})/(R \times N)$$

$$\omega n = [(2.5 \times F_{VCO})/(C1 \times R \times N)]^{1/2}$$

where,

$R = R_{RATE}$ in the low track rate;

$R = (R_{RATE}//R_{BOOST})$ in the high track rate.

From the above equations:

$$\omega_N = (R1 \times G(K))/(2\zeta)$$

$$G(K) = C1 \times (\omega_N^2)$$

$$\zeta = \text{(damping factor)} = (R1 \times \omega_N \times C1)/2$$

The damping factor should be approximately 0.5 when ω_N is minimum. Response to bit shift is minimized when the damping factor is small; however, if the damping factor drops much below 0.5, the system tends to be oscillatory (underdamped). Additionally, loop performance is poor (excessive phase-acquisition times) if the damping factor becomes much larger than 1.0. Any increase in loop bandwidth (due to R decreasing in the high track rate) produces a proportional increase in the damping factor, and should be limited to the point where the maximum damping factor is 1.0. With the damping factor range established, loop design can proceed.

From the Disk Interface Design Guide And User's Manual Chapter 1, Section 1.3–1.7, it is shown that a 946 krads/sec loop bandwidth during acquisition results in a 7 byte crystal reference clock acquisition and data frequency acquisition (VCO settled to within 2 ns of window center). We recommend that these design guide sections be reviewed in conjunction with the DP8462 data sheet in order to obtain a more detailed explanation of the loop bandwidth selection used here, as well as for disk system PLLs in general.

This design example is for a 10 MBit/sec data rate and assumes that the IBOOST ENABLE pin is tied to the $\overline{\text{PREAMBLE DETECTED}}$ pin. This results in the track rate being switched from high to low after four bytes of preamble are detected. As an alternative, the IBOOST ENABLE pin may be tied to an inverted READ GATE signal, resulting in the track rate switching immediately to low when READ GATE is asserted. This is discussed further in the Design Guide.

We will assume a 1-0-0-0 . . . preamble. During acquisition we are in the high track rate and thus ω_N is at maximum value. In the read mode the highest frequency pattern we can encounter is 1-0-0 . . . ; however, ω_N will be lower since we will be in the low track rate.

$$\omega_N = [(2.5 \times F_{VCO})/(C1 \times R \times N)]^{1/2}$$

Choose $R_P = R_{RATE} // R_{BOOST} = 575\Omega$*

$$946 \times 10^3 = [(2.5 \times 20 \times 10^6)/(C1 \times 575 \times 4)]^{1/2}$$

$C1 = 0.028\ \mu F$ Choose $C1 = 0.022\ \mu F$

We don't want ζ to exceed 1.0. Therefore,

$$\zeta = \frac{\omega_N \times R1 \times C1}{2}$$

$$1.0 = \frac{(946 \times 10^3 \times R1) \times 0.022 \times 10^{-6}}{2}$$

$R1 = 96\Omega$

Choose $R1 = 100\Omega$

***Note:** Designing a PLL is an iterative procedure. For the DP8462, design values for R_{RATE} and R_{BOOST} typically range from 700Ω to 1.5 kΩ. The application note provides a more thorough discussion for choosing these values.

The continuous-behavior (non-quantized) approximation used to predict loop performance assumes that the phase detector output is constantly proportional to the input phase difference. In reality, the phase detector output is a pulse applied for a period of time equal to the phase difference. The function of C2 is to "smooth" the phase detector output (VCO control voltage) over each cycle. C2 also adds a second pole to the filter transfer function. This pole should be far enough outside the loop bandwidth (at least one order of magnitude) that its phase and amplitude contribution is negligable in the loop bandwidth. If

$$C2 = C1/50 = 390\ pF$$

the acquisition performance and the margin loss are not significantly changed from the predictions. If a larger C2 is used, the margin loss can be reduced at the expense of the acquisition time. This may be desirable for some systems. Please see the Disk Interface Design Guide And User's Manual Chapter 1, Sections 1.3–1.7 for a discussion of the function of C2.

Circuit Operation (Continued)

As soon as the $\overline{\text{PREAMBLE DETECTED}}$ output goes low we switch to the low track rate. To maintain stability we must ensure that $\zeta_{min} \geq 0.5$.

ζ_{min} occurs when ω_N is minimum; i.e., when we have seven consecutive zeroes (N = 8).

$$\zeta_{min} = \frac{(\omega_{Nmin} \times R1 \times C1)}{2}$$

$$0.5 = \frac{(\omega_{Nmin} \times 100 \times 0.022 \times 10^{-6})}{2}$$

$$\omega_{Nmin} = 454.5 \text{ krads/sec}$$

We can now calculate R_{RATE}

$$\omega_{Nmin} = [(2.5 \times F_{VCO})/(C1 \times R_{RATE} \times N)]^{1/2}$$

$$454.5 \times 10^3 = [(2.5 \times 20 \times 10^6)/(0.022 \times 10^{-6} \times R_{RATE} \times 8)]^{1/2}$$

Therefore, $R_{RATE} = 1.375\ k\Omega$

Choose, $R_{RATE} = 1.2\ k\Omega$

Now we calculate ω_{Nmax} and ζ_{max} in the low track rate

$$\omega_{Nmax} = [(2.5 \times 20 \times 10^6)/(0.022 \times 10^{-6} \times R_{RATE} \times 3)]^{1/2}$$

$$\omega_{Nmax} = 794 \text{ krads/sec}$$

$$\zeta_{max} = \frac{(\omega_{Nmax} \times R1 \times C1)}{2}$$

$$\zeta_{max} = 0.87$$

The final component to be determined is R_{BOOST}

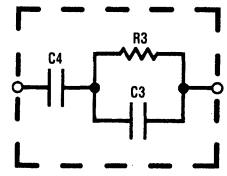
$$\text{Since, } Rp = \frac{R_{BOOST} \times R_{RATE}}{R_{BOOST} + R_{RATE}}$$

$$575 = \frac{R_{BOOST} \times 1.2 \times 10^3}{R_{BOOST} + 1.2 \times 10^3}$$

Therefore, $R_{BOOST} = 1.1\ k\Omega$

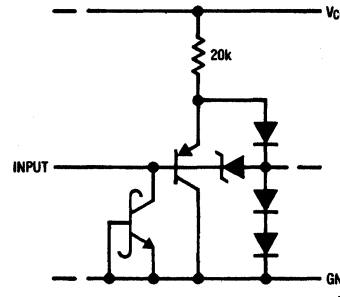
DIGITAL CONNECTIONS TO THE DP8462

Figure 16 shows a connection diagram for the DP8462 in a typical application. All logic inputs and outputs are TTL compatible as shown in *Figure 14* and *15*. The VCO Clock output is 74AS compatible. All other outputs are 74ALS compatible. All inputs are 74ALS compatible and therefore can be driven easily from any 74 series devices.



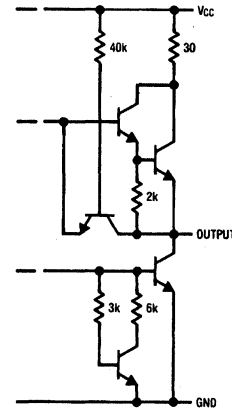
TL/F/8418–17

FIGURE 13. Alternate Loop Filter Configuration



TL/F/8418–16

FIGURE 14. Logic Inputs



TL/F/8418–18

FIGURE 15. Logic Outputs

TABLE II. Loop Filter External Component Values

Data Rate (NRZ)	Pulse Gate Components (Note 3)				Charge Pump (Note 1)		Loop Filter (Note 2)		
	R_{PG2}	R_{PG1}	C_{PG1}	C_{PG2}	R_{RATE}	R_{BOOST}	R_1	C_1	C_2
5 Mbit/sec	4.7k	150Ω	1.0 μF	0.1 μF	820Ω	1.5 kΩ	100Ω	0.03 μF	600 pF
10 Mbit/sec	1.8k	68Ω	2.2 μF	0.22 μF	1.2 kΩ	1.1 kΩ	100Ω	0.022 μF	390 pF
15 Mbit/sec	0.75k	39Ω	3.9 μF	0.39 μF	820Ω	2.7 kΩ	33Ω	0.082 μF	1600 pF

Note 1: Component tolerances are system dependent, they depend on how much loop gain deviation can be tolerated.

Note 2: Component tolerances are typically 5% but they depend on the amount of Loop Bandwidth tolerance that can be accepted.

These values have been altered from calculated values based on empirical tests of the loop.

Note 3: Component tolerances typically 10%, not critical.

Circuit Operation (Continued)

The incoming data from the pulse detector in the drive is connected to the ENCODED DATA input. PREAMBLE SELECT input is tied high or low depending on whether the user's system is employing 1000 or 100 preamble pattern. The LOCK CTL input is to be tied high or low depending on whether or not it is desired to keep the PLL in Phase and Frequency comparison mode while detecting preamble. Phase and Frequency comparison lock while detecting preamble will eliminate the chances of the PLL locking onto a harmonic of the preamble frequency when Read mode is first entered. (Susceptibility to a harmonic lock is increased when using the 1000 preamble). Since a high level on IBOOST ENABLE input puts the PLL in high track rate, it should be held high during Non-Read (standby) mode so that a quick lock is achieved upon entering Read mode. Once the PLL is locked onto the incoming data, however, this input should be taken low. Although the user is free to do this anytime, one possible method is to tie this input to the $\overline{\text{PREAMBLE DETECTED}}$ output of the chip—as shown in *Figure 16*. The READ GATE input is used to place the chip in and out of Read mode and therefore should be tied to the controller and/or a 2, 7 code Encoder/Decoder).

As for the outputs, SYNCHRONIZED DATA and VCO CLOCK may be tied to the Encoder/Decoder—which in turn would deseriallize and decode the data before sending it to the controller. $\overline{\text{PREAMBLE DETECTED}}$ output can be tied to the controller and/or the Encoder/Decoder to provide an indication when 4 consecutive bytes of preamble pattern have been detected. The only output that is not shown in *Figure 16* is the PHASE COMPARATOR TEST output. This output is the logical OR of the Phase Comparator's outputs (Charge-Up and Charge-Down inputs of the Charge-Pump). As such, pulses generated at this output provide information about the loop filter's behavior in that the envelope of the pulses generated at this output is a waveform that represents the loop filter's response to any phase difference detected by the Phase Comparator.

Finally, to improve noise immunity, Digital and Analog VCC pins should be tied together and also the Digital and Analog Ground pins should be tied together. PG1 pin should also be grounded.

Applications of the DP8462 Data Synchronizer

The DP8462 is part of National Semiconductor's DP8460 Series Disk Chip Set and therefore, is designed to work in conjunction with other members of this family; such as DP8464—the pulse detector, and DP8466—the disk data controller. A typical system application employing these components is shown in *Figure 17*. The DP8462 is based upon the proven circuitry of the DP8465 (Data synchronizer and separator for the MFM code)—the first integrated circuit to place on one chip a PLL with features that offer the improved speed and reliability required by the disk industry. Not only does the chip simplify disk system design, but also provides fast lock-on to the incoming preamble. Once locked on, the loop is set into a lower bandwidth mode. This inherent loop stability allows for a sizable amount of jitter on the data stream, such as is encountered in many disk systems. Once in the stable tracking rate, the SYNCHRONIZED DATA output represents the incoming ENCODED DATA and is synchronous with VCO CLOCK. This synchronized data is then deserialized by the ENDEC using the VCO CLOCK.

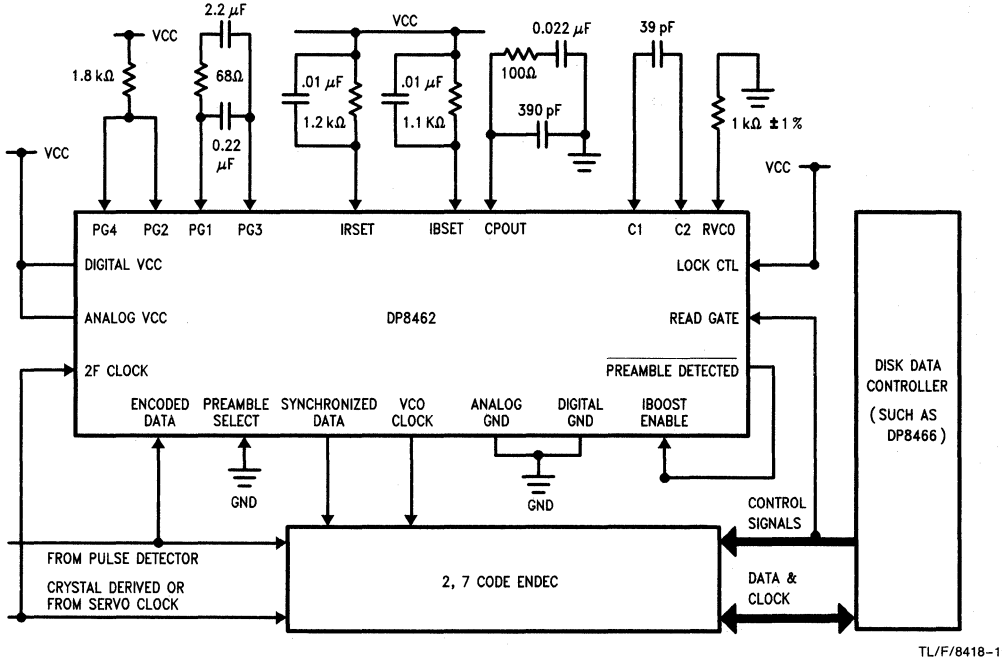
The DP8462 is capable of operating at up to a 20 Mbits/sec data rate and so is compatible with a wide assortment of disk drives. The faster data rates of the 8-inch and 14-inch disk drives will mandate the selection of the DP8462-3 parts with narrower window margin on the incoming data stream. This will also be the case when 5¼-inch drives achieve higher data rates. Some 8-inch and 14-inch disk drives incorporate the functions of the DP8462, but use many discrete ICs. In these cases, replacing these components with the DP8462 will offer reduced P.C. board area, lower cost, and improved performance while simplifying circuit testing.

Most 5¼-inch and many 8-inch and 14-inch disk drives manufactured at present do not incorporate any of the functions of the DP8462. This is so primarily because the PLL function is difficult to design and implement and requires circuitry which covers a large area of the printed circuit card. This is undesirable both from the drive size aspect and from the cost aspect (the cost includes soldering, testing, and adjusting the components). Consequently, most smaller disk drives output RLL encoded data so that the phase-locked-loop and data separation have to be performed by the controller. The DP8462 will therefore replace these functions in controller designs, as shown in *Figure 18a*.

System design criteria may now change because the DP8462 is a one-chip solution, requiring only a few external passive components with fixed values. It operates from a +5V supply, consumes about 0.5W, and is housed in a narrow 24-pin package. The circuitry has been designed so that the external resistors and capacitors need not be adjustable; the user chooses the values according to the disk drive requirements. Once selected, they will be fixed for that particular drive type. These features make it possible to transfer these functions to the disk drive, as shown in *Figure 18b*. Apart from a slight increase in board area, the advantages outweigh the disadvantages. First, the components selected are fixed for each type of drive and this facilitates the problem of interchangeability of drives. At present, components in the controller are adjusted to function with each specific drive; with the DP8462 in the drive, component adjustment will no longer be required. Second, there is often a problem of reliability of data transfer. The incoming data signal is susceptible to noise, bit shift, etc. Soft errors will occur when the incoming disk data bit position is outside the Pulse Gate window as it is being synchronized to the VCO clock in the phase-locked-loop. Obviously, the nearer the PLL is to the data source, the less chance there is that errors will occur. Thus placing the DP8462 in the drive will increase the reliability of data transfer within the system.

Applications of the DP8462 Data Synchronizer (Continued)

A third advantage is data rate upgrading. Most 5¼-inch drives have 5 Mbit/sec data rate because the early drives were made with this data rate. This meant the controllers had to be designed with PLLs which operate at this data rate. It is therefore difficult for drive manufacturers to introduce new drives that are not compatible with existing controllers. Since no new standard data rate has emerged, they must continue to produce drives at this data rate to be compatible with the controllers on the market. With the DP8462 in the drive, and its associated components set for the drive's data rate, it no longer becomes a problem to increase the data rate, assuming the controller's digital circuitry can accommodate the change. This will allow the manufacturers to increase the bit density and therefore the capacity of their drives.

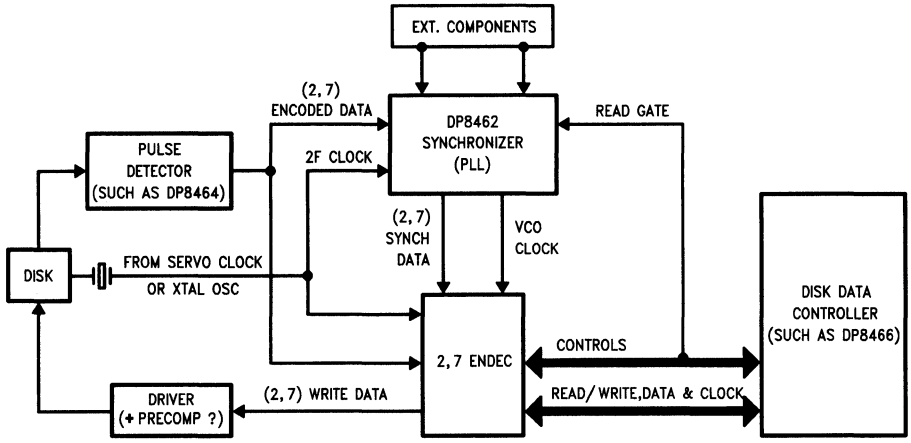


TL/F/8418–19

FIGURE 16. Typical Connection to DP8462 For:

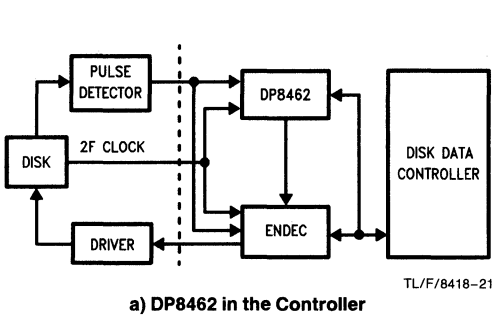
1) RLL (2,7 Code) Data Input, 10 Mbit/sec Data Rate
2) 1-0-0 Preamble Pattern
3) PLL to stay in Phase-Frequency Comparison mode until 4 bytes of Preamble Detected
4) PLL to stay in high Track Rate until $\overline{\text{PREAMBLE DETECTED}}$ asserted
5) Delay line left unadjusted (PG2 & PG4 shorted together)

Applications of the DP8462 Data Synchronizer (Continued)



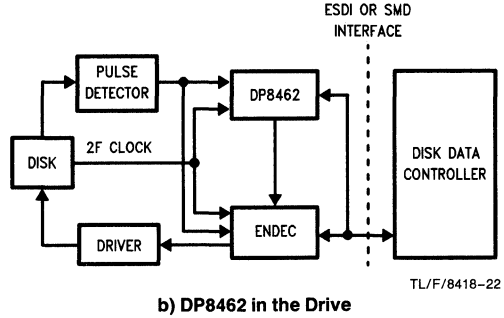
TL/F/8418–20

FIGURE 17. Typical Application of DP8462 in a System Employing RLL (2,7) Code



TL/F/8418–21

a) DP8462 in the Controller



TL/F/8418–22

b) DP8462 in the Drive

FIGURE 18. Two Different Methods of Utilizing DP8462

PRECAUTIONS IN BREADBOARDING AND PCB LAYOUT

The DP8462 contains a high performance analog PLL and certain precautions must be taken when breadboarding or designing a PCB layout. The following guidelines should be adhered to when working with the DP8462:

1) Do not wire wrap.
2) Keep component lead lengths short, place components as close to pins as possible. This applies to R1, C1, R2, C_{VCO}, R_{RATE}, R_{BOOST}, C_{RATE}, C_{BOOST}, RPG1, RPG2, and CPG1.
3) Provide a good ground plane and use a liberal amount of supply bypassing. The quieter a PLL's environment, the happier it is.
4) Avoid routing any digital leads within the vicinity of the analog leads and components.

We have used a PC board approach to breadboarding the DP8462 that gives us an excellent ground plane and keeps component lead lengths very short. With this setup we have found very stable and reliable operation. Illustrations of component layout is shown in *Figure 19*. Note that the board layout is a recommendation not a requirement.

ADDITIONAL NOTES

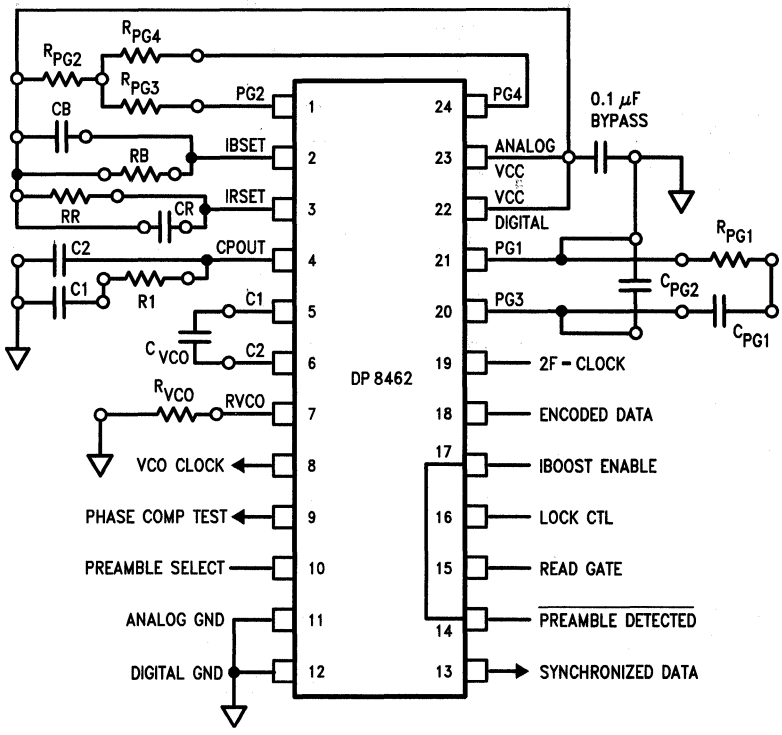
1) PG1 should be grounded to improve noise immunity.
2) 2F clock must be applied at all times; without the 2F clock, the pulse gate circuitry will not operate properly making it impossible to lock onto the incoming data stream.
3) The programming capacitor for the V_{CO} can be calculated as:

$$C_{VCO} = 1/(f_{VCO} * R_{VCO}) - 5\ pF$$

The 5 pF value is due to parasitic internal device capacitance.

4) Care must be taken in final PC board layout to minimize pin to pin capacitance, particularly in multi-layer printed circuit boards.
5) Please refer also to Precaution For Disk Data Separator Designs, NSC Application Note AN-414.

Connection Diagram



TL/F/8418–23

FIGURE 19. Recommended Component Layout

National Semiconductor Corporation

PRELIMINARY

DP8463B (2, 7) ENDEC

General Description

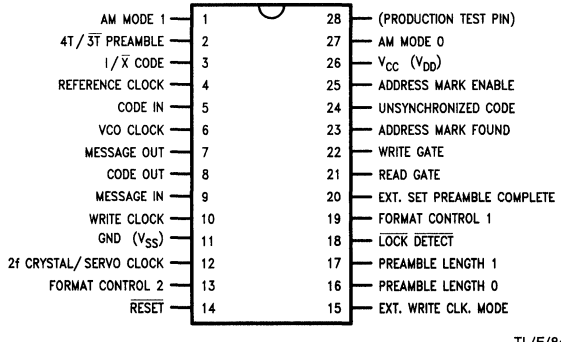
The DP8463B (2, 7) ENDEC performs the encoding and decoding necessary to use either of two different Run-Length Limited (RLL) Codes for disk drive memory systems. Either code gives a disk system the ability to record up to 50% more message data in the same media space without any increase in the Flux Changes per Inch (FCI), when compared to a system using Modified Frequency Modulation (MFM) coding. The DP8463B also performs several other functions to enable many disk controllers, designed for MFM or other codes, to use RLL codes. These added functions are: the writing and reading of an Address Mark for soft-sectored disk formats, and the writing and reading of a Preamble pattern (Phase Locked Loop (PLL) Sync Field) that is compatible with the (2, 7) RLL Code. There are three different Address Marks and two Preamble patterns that may be programmed along with two possible encode/decode tables in seven possible formats for soft sectored disks or three possible formats for hard sectored disks. The user can also select two possible lengths of Preamble to count before issuing a "Lock Detect" signal or an external, user provided, signal may be accepted so any Preamble length may be counted before issuing a Lock Detect signal. The term "Message" is used to designate unencoded data, and the term "Code" is used to designate the encoded data.

Features

- Up to 50% increase in recorded data density over MFM
- Encodes and decodes IBM (2, 7) Code
- Encodes and decodes Xerox (2, 7) Code
- Soft-sector Address Mark generation and detection
- Preamble generation and detection (maximum frequency "3T" = 100 . . . or "4T" = 1000 . . .)
- Programmable formats:
  - Hard sector
  - Soft sector with Address Mark preceding Preamble
  - Soft sector with Address Mark following Preamble
- Programmable Address Mark:
  - SMD 3-Byte gap with no transitions, preceding Preamble
  - IBM 2-Byte gap with two transitions, preceding Preamble
  - Address Mark not violating (2, 7) constraints following Preamble
- Programmable Preamble length counted before "lock-detect" issued:
  - Externally determined
  - 6 Bytes
  - 8 Bytes
- Code output is synchronized to 2f crystal/servo clock
- Glitchless multiplexer switching between VCO clock for reading and 2f clock for writing
- Message Data Rate to 25M bits per second (Code Rate = 50 Mbps)
- TTL Compatible Inputs and Outputs
- Compatible with Phase-Locked-Loop of DP8462 Data Separator
- Compatible with DP8466 Disk Data Controller
- 28-Pin wide Dual-In-Line Package & 28 pin Plastic Chip Carrier
- 2-Micron, Dual Metal CMOS
- Single +5V Supply

Connection Diagrams

Dual-In-Line Package

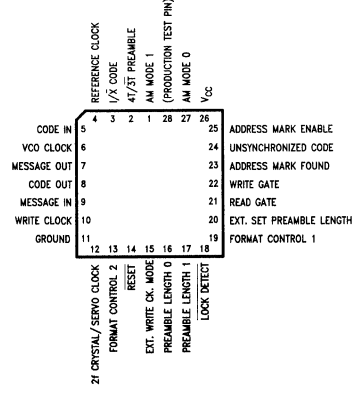


TL/F/8433-1

Top View

28 Pin Package
Order Number DP8463BN
See NS Package N28B

Plastic Chip Carrier



TL/F/8433-2

Top View

28 Pin Package
Order Number DP8463BV
See NS Package V28A

Absolute Maximum Ratings

Supply Voltage	−0.5 to 7V
Input or Output Voltage	−0.5 to V_{CC} + 0.5V
Storage Temperature	−65°C to +150°C
Lead Temperature	300°C

Recommended Operating Conditions

Symbol	Parameter	Min	Max	Units
V_{CC}	DC Supply Voltage	3	6	V
V_I, V_O	Input or Output Voltage	0	V_{CC}	V
I_O	High or Low Level Output Current	0	±25	mA
I_{CC}	V_{CC} or GND per pad	0	±50	mA
T_A	Operating Temperature Range	−40	+85	°C

DC Electrical Characteristics

V_{CC} = 5V ±10%, Min./Max. limits apply across temperature unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
V_{IH}	Min. High Level Input Voltage	V_O = 0.5V or V_{CC}−1V, $\|I_O\|$ = 1μA	2		V
V_{IL}	Max. Low Level Input Voltage	V_O = 0.5V or V_{CC}−1V, $\|I_O\|$ = 1μA		0.8	V
V_{OH}	High Level Output Voltage	V_I = V_{CC} or GND, $\|I_O\|$ = 1 μA	V_{CC}−0.05		V
V_{OL}	Low Level Output Voltage	V_I = V_{CC} or GND, $\|I_O\|$ = 1 μA		0.05	V
I_{OH}	High Level Output Current	V_I = V_{CC} or GND, V_O = V_{CC}−0.8V	−6		mA
I_{OL}	Low Level Output Current	V_I = V_{CC} or GND, V_O = 0.4V	6		mA
I_{IH}	High Level Input Current	V_{CC} = Max., V_I = 2.7V		+1	μA
I_{IL}	Low Level Input Current	V_{CC} = Max., V_I = 0.4V		−45	μA
I_{CC}	Supply Current	V_I = V_{CC} or GND T_A = 25°C V_{CO} = 50 Mb/s		100	mA

AC Electrical Characteristics

V_{CC} = 5V ±10%, Min./Max. limits apply across temperature unless otherwise specified. Input load = 1 pF, Output Load = 15 pF.

Symbol	Parameter		Part No	Min	Typ	Max	Units
f_{DATA}	Maximum message data frequency		DP8463-10	10			Mb/s
			DP8463-20	20	25		Mb/s
f_{VCO}	Maximum VCO frequency		DP8463-10	20			Mb/s
			DP8463-20	40	50		Mb/s
$t_{MI\ SET-UP}$	Set-up time of MESSAGE IN before WRITE CLOCK positive edge for IBM Encoder			10			ns
$t_{MI\ HOLD}$	Hold Time of MESSAGE IN for IBM Encoder			0			ns
$t_{CK-DATA}$	Propagation delay allowed external controller between negative edge of REFERENCE CLOCK and start and of MESSAGE IN for Xerox Encoder	DP8463-10	Freq. 10 Mb/s	V10		60	ns
		DP8463-20	15	10		50	ns
			20			40	ns
			25	10		30	ns
$t_{CI\ SET-UP}$	Set-up time of CODE IN before VCO CLOCK positive edge			7			ns
$t_{CI\ HOLD}$	Hold time of CODE IN after VCO CLOCK positive edge			3			ns
$t_{pd\ IE}$	Propagation delay of IBM Encoder from positive edge of WRITE CLOCK to start of CODE OUT (5 = logical delay, 1 = circuit delay)			5	6	6	code bits
$t_{pd\ ID}$	Propagation delay of IBM Decoder from VCO CLOCK to MESSAGE OUT			4		6	code bits

AC Electrical Characteristics V_{CC} = 5V ±10%, Min./Max. limits apply across temperature unless otherwise specified. Input load = 1 pF, Output Load = 15 pF. (Continued)

Symbol	Parameter	Min	Typ	Max	Units
$t_{pd\ XE}$	Propagation delay of Xerox Encoder from internal strobe of MESSAGE IN to CODE OUT	3		4	code bits
$t_{pd\ XD}$	Propagation delay of Xerox Decoder from VCO CLOCK to MESSAGE OUT	8		9	code bits

Note: Mb/s = Megabits/second

DP8463 Pin Definitions

POWER

11. GND (V_{SS})

26. V_{CC} (V_{DD})

INPUT SIGNALS

14. $\overline{\text{RESET}}$

Active low input resets various flip-flops when the next VCO CLOCK positive pulse is received. The DP8463 should be reset after each time power is applied.

6. VCO CLOCK

This is the clock output of the data separator circuit (e.g., DP8462). During the read mode (i.e., when READ GATE is high) it is the clock derived from the code recorded on the disk. During the write mode, it is the 2f frequency output of a crystal oscillator or a disk's servo track.

5. CODE IN

This is the encoded data output of the data separator circuit (e.g., DP8462). It is synchronous with the VCO CLOCK. The positive edge of the VCO CLOCK signal is used to strobe the CODE IN signal into the (2, 7) ENDEC. A "one" is a high level for 7 or more nanoseconds before the positive VCO CLOCK edge.

24. UNSYNCHRONIZED CODE

This is the encoded data output of the pulse detector circuit (e.g., DP8464). It is used by the DP8463 for detecting soft-sector Address Marks (gaps) that precede the Preamble (=PLL=PLO Sync Field). A "one" is a high level for 7 or more nanoseconds. A "one" resets the Address Mark Counter which counts VCO CLOCK pulses.

9. MESSAGE IN

This is the unencoded "write data" from the disk data controller (e.g., DP8466). When IBM codes are used (i.e., I/$\overline{\text{X}}$ Code input is high) and a high level is on the EXTERNAL WRITE CLOCK MODE input, the WRITE CLOCK strobes the MESSAGE IN data into the DP8463. When Xerox codes are used (I/$\overline{\text{X}}$ Code input is low), the EXTERNAL WRITE CLOCK MODE input must be low. The MESSAGE IN data is strobed into the DP8463 by an internal clock signal that is a 1f clock derived from the VCO CLOCK. The "$t_{CK\text{-}DATA}$" specifies the constraints on the time between the REFERENCE CLOCK and the start of the data MESSAGE IN signal.

10. WRITE CLOCK

This clock strobes the MESSAGE IN (write data) data into the IBM encoder section and is the clock for the IBM encoder.

21. READ GATE

A high level signal places the (2, 7) ENDEC in the mode to read code from the disk and decode it, plus detect and count a number of Preamble patterns. Address Marks that follow the Preamble and do not violate the (2, 7) RLL constraints are read while READ GATE is high.

22. WRITE GATE

A high level signal places the (2, 7) ENDEC in the mode to encode data and write code on the disk, and where appropriate, to write an Address Mark, Preamble pattern and Phase Sync pattern. A low level signal puts the (2, 7) ENDEC in a mode to detect gap-type Address Marks that precede the Preamble.

25. ADDRESS MARK ENABLE

This pin must be held high for the full time an Address Mark is written on a soft sectored disk. WRITE GATE must also be high during the writing of the Address Mark.

PROGRAMMING INPUTS

2. 4T/$\overline{\text{3T}}$ PREAMBLE MODE

A high level places the (2, 7) ENDEC in the mode to generate and detect "4T" Preamble patterns (i.e., 1000 in code which is 4 Time periods). The MESSAGE IN signal is inverted in the (2, 7) ENDEC before being encoded so the 4T code pattern is generated from an all zeros MESSAGE IN data stream. The output of the decoder is also inverted in this mode so the inversion is transparent to the user. The double inversion is performed during all reading/writing, not just during the Preamble. A low level input places the (2, 7) ENDEC in the "3T" Preamble Mode (i.e., 100 . . . code pattern). The I/O is not inverted in this mode.

3. IBM/$\overline{\text{XEROX}}$ CODE

A high level places the (2, 7) ENDEC in a mode to encode and decode according to the IBM Message Data/Code Table. A low level has the (2, 7) ENDEC encode and decode according to the Xerox Message Data/Code Table.

15. EXTERNAL WRITE CLOCK MODE

A high level on this pin makes the DP8463 responsive to WRITE CLOCK inputs. When the DP8463 is being used for Xerox codes, this pin must be low. When IBM codes are used, this pin must be high (except for special modes discussed in a separate application note).

DP 8463 Pin Definitions (24 pin version) (Continued)

1 & 27. ADDRESS MARK MODE

AMM1	AMM0	MODE
0	0	Hard Sector
0	1	Soft Sector, Generate & Detect SMD Address Mark
1	0	Soft Sector, Generate & Detect IBM 8/20 Address Mark
1	1	Soft Sector, Generate & Detect N7V Address Mark

16 & 17. LENGTH OF PREAMBLE READ

PL1	PL0	LENGTH OF PREAMBLE
0	0	Set by EXTPREC Pulse
0	1	6 Message (Data) Bytes
1	0	8 Message (Data) Bytes

These programmable inputs determine the length of Preamble that is read until an active low $\overline{\text{LOCK DETECTED}}$ signal is output.

20. EXTERNALLY SET PREAMBLE READ COMPLETE (EXTPREC)

A positive level ≥ 10 ns from an external counter determines the length of the Preamble read before issuing an active low. $\overline{\text{LOCK DETECT}}$ signal and switching the (2, 7) ENDEC out of its Preamble Mode into its normal Decode Mode. For example, the short 15 ones count of the DP8462 (3 or 4 message bytes length) could be used. The DP8463 must receive at least one message byte of the preamble pattern before the EXTPREC is received.

OUTPUT SIGNALS

4. REFERENCE CLOCK

This is a divide-by-two derivation of the VCO CLOCK received from the Data Separator Circuit. The REFERENCE CLOCK is fed to the "Read Clock" Input of the Disk Data Controller. During the reading of the Preamble, a group of Phase Sync Code Patterns appear so the proper two code bits at the 2f frequency are synchronized with the 1f Reference Clock Flip-Flop. During the read mode, the positive edge of the REFERENCE CLOCK should be used by the Disk Data Controller to strobe the MESSAGE OUT (NRZ) data bit or the ADDRESS MARK FOUND signal. During the write mode, the negative edge of the REFERENCE CLOCK should cause the Digital Data Controller to issue a new message data bit. With WRITE GATE active (high) the REFERENCE CLOCK is as undelayed as possible from its VCO CLOCK source, to maximize the time window of receiving MESSAGE DATA for the Xerox encoder. When WRITE GATE is inactive (low), the REFERENCE CLOCK is delayed so its positive going edge occurs later in the MESSAGE OUT signal period to maximize the set-up time for the controller to receive it.

8. CODE OUT

A high level for a code bit period is output for each "one" in code that is to be written on the disk media as a flux transition by a write amplifier containing a write flip-flop that changes state every time a positive pulse is received. This output can also be strobed by the appropriate phase of a 2f Clock signal.

23. ADDRESS MARK FOUND (AMF)

When the SMD or IBM Address Mark is detected, a positive pulse, the width of two message data bits, is issued on this pin. When the N7V A.M. is detected, the AMF is one message bit wide. When the Address Mark is a gap preceding the Preamble, this "Address Mark Found" (AMF)

Formats

	External Write Clock				Internal Write Clock					
	Soft Sector			Hard Sector	Soft Sector				Hard Sector	
Format Element:										
Address Mark, SMD	1					1	1			
Address Mark, IBM		1								
Preamble, 4T, ENDEC inverts I/O	2	2	1	1	1	2			1	
Preamble, 3T, Natural							2	1		1
Phase Sync, 4T, Natural							3	2		2
Address Mark, Non (2, 7) violation			2		2			3		
Code, IBM	3	3	3	2						
Code, Xerox					3	3	4	4	2	3

Notes:
A. Each table entry is the sequence of appearance of the format element.
B. "Natural" means the DP8463 outputs are from inputs transformed only by the encoder/decoder without inversions in the DP8463.
C. The 4T Preamble inherently provides the Phase Sync function.

DP 8463 Pin Definitions (Continued)

can be used as a "Sector Mark" input to a disk data controller operating in a hard sector mode. When the AM follows the Preamble, the disk data controller would synchronize on byte boundaries when it received the AMF and a pre-programmed byte is detected (e.g. see DP8466 specifications).

7. MESSAGE OUT

This is the "read data" for the disk data controller. A high level signal represents a "one" of decoded data. It is strobed into the disk data controller by the positive edge of the REFERENCE CLOCK.

18. $\overline{\text{LOCK DETECT}}$

A low level signifies that a minimum, uninterrupted length of a Preamble pattern has been read. The length of Preamble is programmed by PL1 and PL0. This output may be used to switch a Data Separator Circuit from a high to a low tracking rate mode.

Message Data/Code Tables

1. IBM (2, 7, 1, 2, 3) Message Data/Code Table

Normal Message Data		Inverted Message Data		Code	
MSB	LSB	MSB	LSB	MSB	LSB
000		111		000100	
10		01		0100	
010		101		100100	
0010		1101		00100100	
11		00		1000	
011		100		001000	
0011		1100		00001000	

Most Significant Bit (MSB) is read/written first

2. XEROX (2, 7, 1, 2, 3) Message Data/Code Table

Normal Message Data		Inverted Message Data		Code	
MSB	LSB	MSB	LSB	MSB	LSB
1		0		X0	
01		10		0001	
001		110		000010	
000		111		001001	

X = 1, if 2 or more zeros immediately precede present code word

X = 0, if either of the 2 preceding bits is a one

MSB is read/written first

NOTE:

Definition of (2, 7, 1, 2, 3):

2 = minimum number of zeros between adjacent ones

7 = maximum number of zeros between adjacent ones

1 = } ratio of message data bits to code bits,

2 = } first number is message data

3 = number of different lengths of message data words

Description of Circuit Characteristics

1. Address Marks

There are three programmable Address Marks for soft sector formats:

A. SMD Address Mark:

This is a gap without any flux transitions on the disk for a length of three message data bytes. It appears at the start of a sector. It is written by the Disk Data Controller (DDC) having its ADDRESS MARK ENABLE active high for a time equal to 24 message data bits while WRITE GATE is also active high. The gap is detected any time an interval of 16 message data bits passes without a flux transition. The detection occurs when WRITE GATE is inactive (low). The state of ADDRESS MARK ENABLE during the detecting of the gap is a "don't care" so the DP8463 is compatible with both Storage Module Device (SMD) and Enhanced Small Disk Interface (ESDI) standards.

B. IBM Address Mark:

This Address Mark is a gap of 32 code bits which has no flux transitions except for two transitions in positions 8 and 20 (and the first seven positions are "don't care" values according to IBM's definition). The DP8463 writes a "1" in positions 1, 8 and 20, and detects the Mark by detecting two gaps of 8 to 11 bits with a "1" between the two gaps. (The "1" in position 1 insures that the first detected gap will be after, not before, position eight.)

C. Non Seven Violation (N7V) Address Mark:

This address mark is a unique word which does not violate the (2,7) RLL constraints but can not be generated by a message data pattern in the Xerox or IBM encoders. This N7V Address Mark (A.M.) should follow the Preamble. The N7V A.M. is two message data bytes wide. The first byte is a "4T" phase sync pattern in code. The second byte is the unique A.M. pattern.

2. Phase Sync Pattern

Since (2, 7) RLL Code has two bits for every one message data bit, the decoder must be able to align itself properly on the first bit of each code word. The maximum frequency code pattern of (2, 7) RLL Code is the "3T" pattern "100." In decoding a series of 100100...code patterns, it is impossible to determine if any individual code "1" is an odd or even bit unless you know exactly where the pattern began, which is impossible in the Preamble.

If the "4T" code pattern is used, i.e., 1000, and if we know what message data pattern was used to encode it, then the decoder knows that every "1" in code is the beginning of a code word, when an all ones message data pattern was input to the encoder during the Preamble. Therefore, a "4T" Preamble is able to attain phase sync naturally while a "3T" Preamble must be followed by a 4T pattern to attain phase sync before any data is read.

3. Byte Sync Pattern

The purpose of a Byte Sync pattern is to tell the Disk Data Controller the location of the message data byte boundary. With hard-sector disks or soft sector formats where the Address Mark precedes the Preamble, the Sync Byte may be user-selected without any DP8463 constraints. If a "3T" Preamble or a N7V Address Mark is used, there are constraints on the Sync Byte. When a "3T" Preamble is used, the Sync Byte message bits must be "all ones" so that a "4T" code pattern is generated which provides the necessary Phase Sync pattern when reading. When a N7V Address Mark is used, there must also be a "4T" phase sync pattern generated by the Byte Sync. An "all ones" in message bits Byte Sync is used with the "3T" Preamble as noted above, but when the

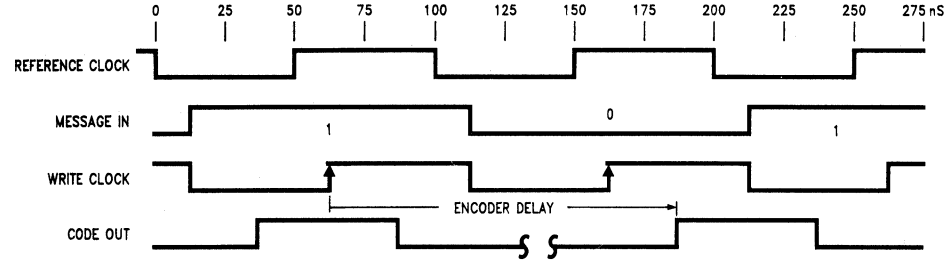
Description of Circuit Characteristics (Continued)

DP8463 is used in its data inverting mode, the Sync Byte must be mostly zeros, i.e., 10000000 in message bits.

4. External Write Clock

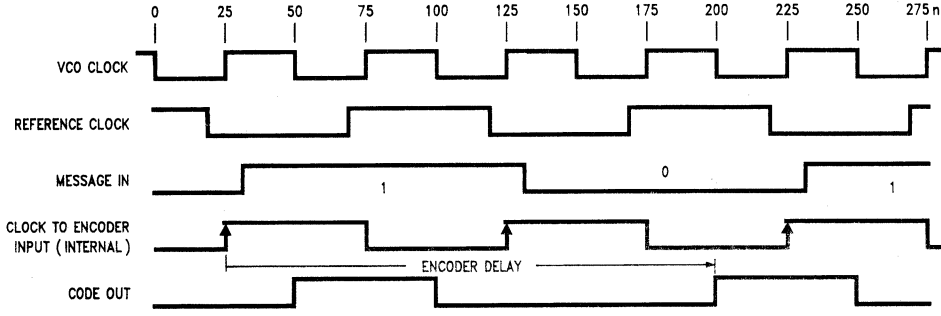
The external WRITE CLOCK input to the DP8463 from the Disk Data Controller (DDC) has its source in the 2f CLOCK, but there are many propagation delays between the two clocks that may vary with different combinations of DP8463s, DDCs, buffers, and cables. When the IBM Encoder is used with the external WRITE CLOCK input, the timing of the CODE OUT signal to the disk's write amplifier is dependent upon the WRITE CLOCK's duty cycle and the different high-to-low versus low-to-high propagation delays in the IBM encoder. This may not be significant at low speeds, but at high speeds an unsymmetrical duty cycle will make a significant contribution to bit jitter of the IBM Encoder output.

IBM ENCODING*
10 Mbits/s



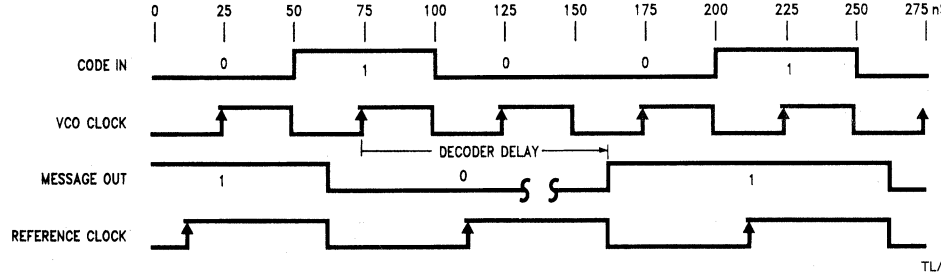
TL/F/8433–3

XEROX ENCODING*
10 Mbits/sec Message data rate



TL/F/8433–4

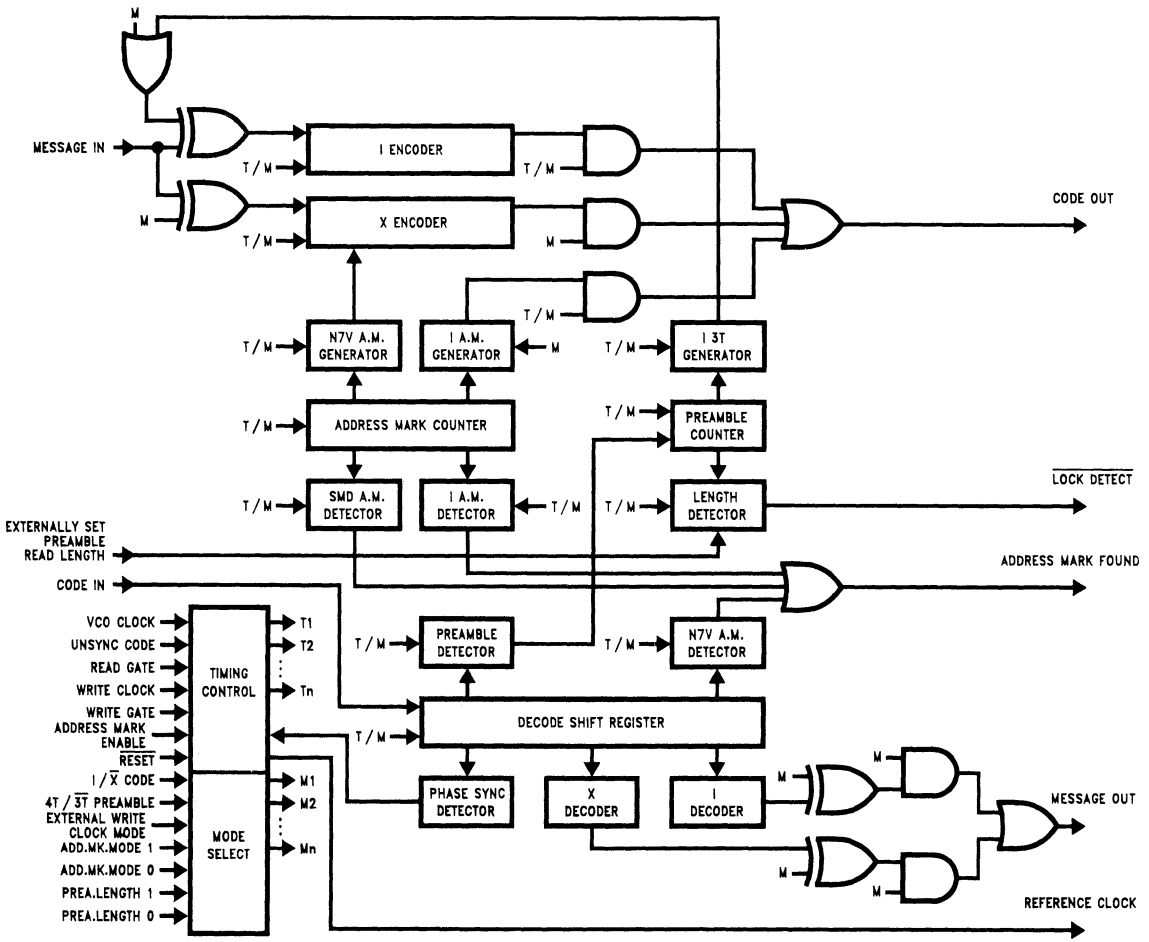
IBM OR XEROX DECODING*
10 Mbits/sec



TL/F/8433–5

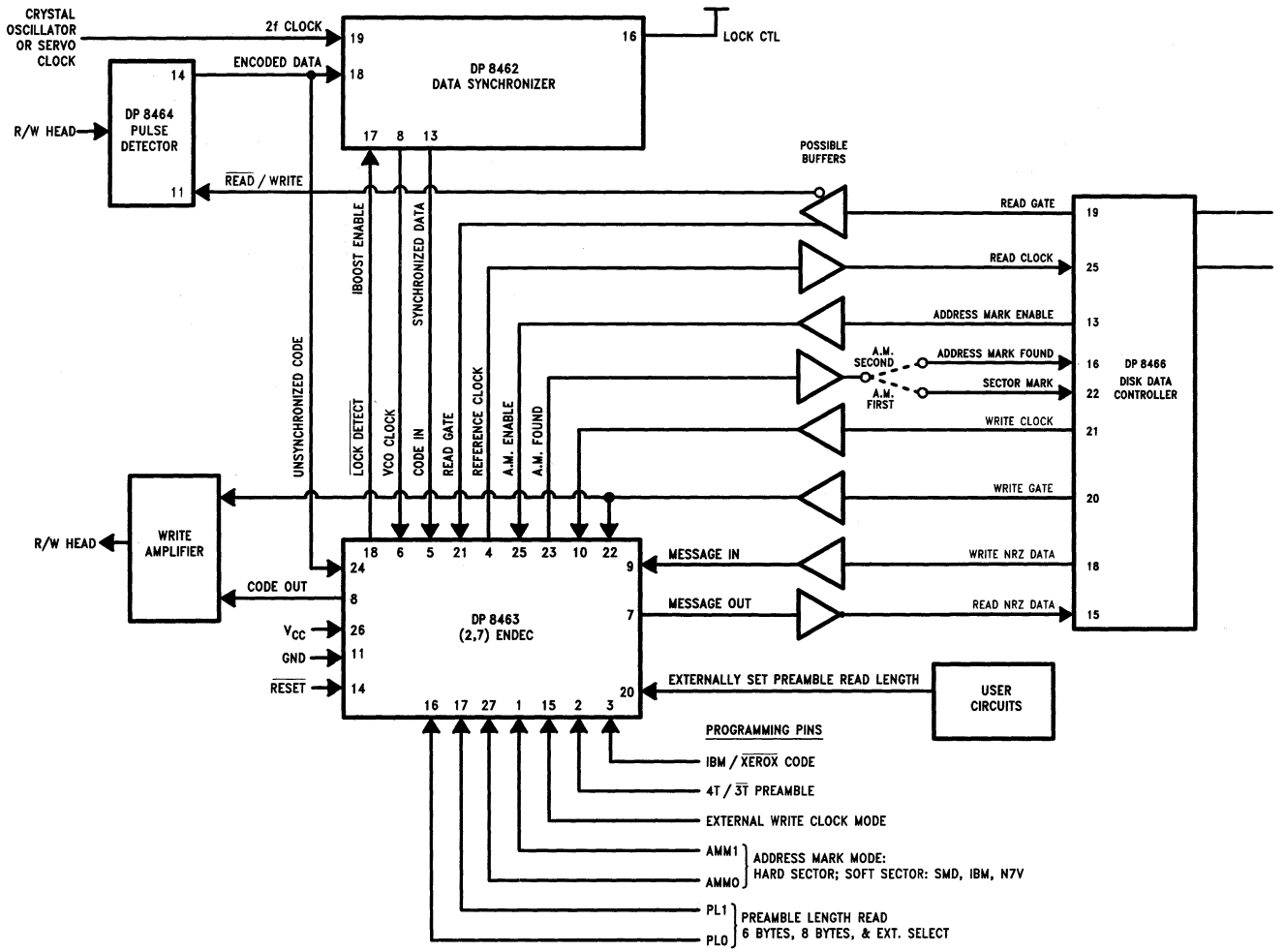
*typical times show

DP8463 (2, 7) Endell Endec Block Diagram



TL/F/8433–6

Hard Disk Chip Set with (2, 7) RLL Codes



TL/F/8433–7

Description of Circuit Characteristics (Continued)

The WRITE CLOCK will always be in sync with the 2f CLOCK but the phase relationship is a variable depending upon the system speed and propagation delays. Therefore, the user must use a flip-flop on the CODE OUT output which is clocked by the correct phase of the 2f CLOCK. With this circuitry, the transition signals to the write amplifier will always occur on the same edge of the 2f CLOCK whose period is very constant. See DP8463 Application Note for suggested circuits to automatically do the resynchronization.

5. Internal Write Clock

When the Xerox Encoder is used, the CODE OUT signal is clocked by a 2f signal that is a derivation of the VCO CLOCK so the output is only switched on the positive edges of the 2f frequency so no jitter error is introduced by the Xerox Encoder. The MESSAGE IN data is also strobed into the Xerox Encoder by a derivation of the VCO CLOCK that is half the CODE OUT frequency. This places a slight constraint on the total propagation delay of the DDC from the time it receives the REFERENCE CLOCK (Read Clock to DDC) until the MESSAGE IN (Write Data from DDC) signal is issued. The time limits are specified in the AC Electrical Characteristics.

6. Error Propagation

Since a single bit shift error in a code word may be decoded as a different message data word, there is error propagation. The longest error burst found for the IBM Code is 5 bits long, and for the Xerox Code, 6 bits. Therefore, the disk system must have Error Checking and Correcting (ECC) circuitry capable of correcting these errors.

National Semiconductor Corporation

DP8464B Disk Pulse Detector

General Description

The DP8464B Disk Pulse Detector utilizes analog and digital circuitry to detect amplitude peaks of the signal received from the read/write amplifier fitted with the heads of disk drives. The DP8464B produces a TTL compatible output which, on the positive leading edge, indicates a signal peak. Electrically, these peaks correspond to flux reversals on the magnetic medium. The signal from the read/write amplifier when reading a disk is therefore a series of pulses with alternating polarity. The Disk Pulse Detector accurately replicates the time position of these peaks.

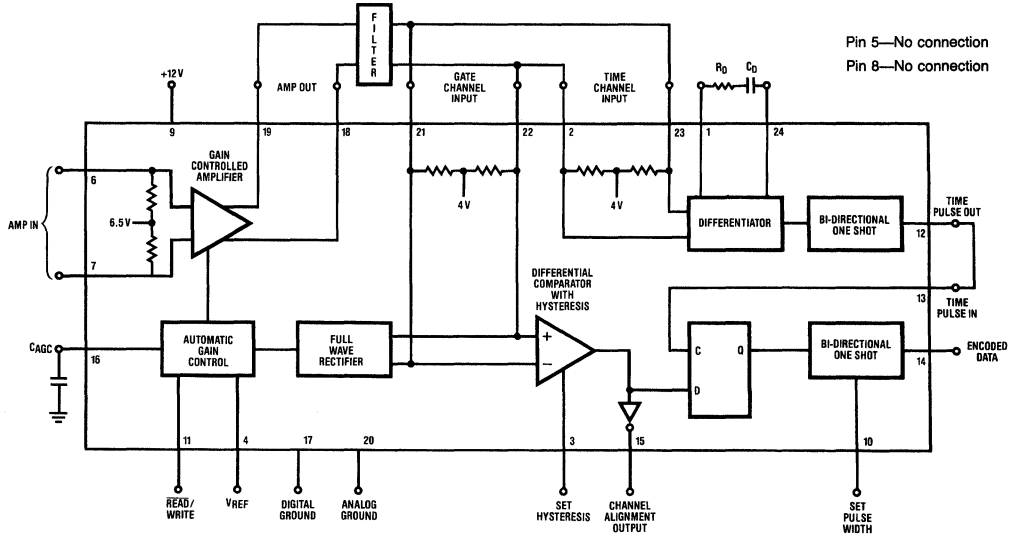
The DP8464B Disk Pulse Detector has three main sections: the Amplifier, the time channel and the gate channel. The Amplifier section consists of a wide bandwidth amplifier, a full wave rectifier and Automatic Gain Control (AGC). The time channel is made from the differentiator and its following bi-directional one shot, while the gate channel is made from the differential comparator with hysteresis, the D flip-flop and its following bi-directional one shot.

The Disk Pulse Detector is fabricated using an advanced oxide isolated Schottky process, and has been designed to function with data rates up to 15 Megabits/second. The DP8464B is available in either a 300 mil wide 24-pin dual-in-line package or a surface mount 28-pin plastic chip carrier package. Normally, it will be fitted in the disk drive, and its output may be directly connected to the DP8461 or the DP8465 Data Separator.

Features

- Wide input signal amplitude range—from 20 mVpp to 660 mVpp differential
- Data rates up to 15 Megabits/sec 2,7 code
- On-chip differential gain controlled amplifier, differentiator, comparator gating circuitry, and output pulse generator
- Input capacitively coupled directly from the disk head read/write amplifier
- Adjustable comparator hysteresis
- AGC and differentiator time constants set by external components
- TTL compatible digital Inputs and Outputs
- Encoded Data Output may connect directly to the DP8461 or DP8465 Data Separator
- Standard drive supply: 12V ± 10%
- Available in 300 mil wide 24-pin dual-in-line package or a surface mount 28-pin plastic chip carrier package

Block Diagram



TL/F/5283–7

Note: All pin numbers in this data sheet refer to the 24-pin dual-in-line package.

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

	Pins	Limit
Supply Voltage	9	14V
TTL Input Voltage	11,13	5.5V
TTL Output Voltage	12,14,15	5.5V
Input Voltage	3,4	5.5V
Minimum Input Voltage	3,4	−0.5V
Differential Input Voltage	6-7, 21-22, 2-23	3V or −3V

Storage Temperature	−65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	300°C
Maximum Power Dissipation at 25°C	
Molded DIP Package (derate 15.6 mW/°C above 25°C)	1950 mW
Plastic Chip Carrier Package (derate 12.5 mW/°C above 25°C)	1560 mW

Operating Conditions

Symbol	Parameter	Min	Typ	Max	Units
V_{CC}	Supply Voltage	10.8	12.0	13.2	V
T_A	Ambient Temperature	0		70	°C

DC Electrical Characteristics

Over Recommended Operating Temperature and Supply Range V_{REF} = 0.5V, Set Hysteresis = 0.3V. $\overline{Read}$/Write = 0.3V unless otherwise noted. All Pin Numbers Refer to 24 Pin Dual-In-Line Package.

Symbol	Pins	Parameter	Conditions	Min	Typ	Max	Units
AMPLIFIER							
$Z_{IN_{AI}}$	6,7	Amp In Impedance	T_A = 25°C (Note 1)	0.8	1.0	1.2	kΩ
A_{VMIN}	18,19	Min Voltage Gain	AC Output 4 Vpp Differential			6.0	V/V
A_{VMAX}	18,19	Max Voltage Gain	AC Output 4 Vpp Differential	200			V/V
$V_{C_{AGC}}$	16	Voltage on C_{AGC}	A_V = 6.0 A_V = 200	 2.8	4.5 3.7	5.5	V V
GATE CHANNEL							
$Z_{IN_{GCI}}$	21,22	Gate Channel Input Impedance	T_A = 25°C (Note 1)	1.75	2.5	3.25	kΩ
$I_{C_{AGC}-}$	16	Pin 16 Current which Charges C_{AGC}	$V_{PIN\ 16}$ = 3.9V $\lvert V_{PIN\ 21} - V_{PIN\ 22} \rvert$ = 2.6V	−1.5	−2.5	−3.5	mA
$I_{C_{AGC}+}$	16	Pin 16 Current which Discharges C_{AGC}	$V_{PIN\ 16}$ = 5V $\lvert V_{PIN\ 21} - V_{PIN\ 22} \rvert$ = 1.4V		1	5	μA
$I_{V_{REF}}$	4	V_{REF} Input Bias Current			−20	−100	μA
$V_{TH_{AGC}}$	22,21 4,16	AGC Threshold	(Note 2) $V_{PIN\ 16}$ = 4.2V	0.88	1.0	1.12	V
I_{SH}	3	Set Hysteresis Input Bias Current			−60	−100	μA
V_{THSH}	22,21 3,15	Set Hysteresis Threshold	(Note 3)	0.48	0.6	0.72	V
TIME CHANNEL							
$Z_{IN_{TC}}$	2,23	Time Channel Input Impedance	T_A = 25°C (Note 1)	3.5	5.0	6.5	kΩ
I_{C_d}	24	Current into Pin 1 and 24 that Discharges C_d		1.4	1.8	2.2	mA

DC Electrical Characteristics Over Recommended Operating Temperature and Supply Range V_{REF} = 0.5V, Set Hysteresis = 0.3V. $\overline{Read}$/Write = 0.3V unless otherwise noted. All pin numbers refer to the 24 pin dual-in-line package. (Continued)

Symbol	Pins	Parameter	Conditions	Min	Typ	Max	Units
WRITE MODE							
$Z_{IN_{AI}}$	6,7	Amp In Impedance in Write Mode	$V_{PIN\ 11}$ = 2.0V	100		500	Ω
$I_{C_{AGC}-}$	16	Pin 16 Current in Write Mode	$V_{PIN\ 11}$ = 2.0V $V_{PIN\ 16}$ = 3.9V $\lvert V_{PIN\ 21} - V_{PIN\ 22} \rvert$ = 1.3V		1	5	μA
DIGITAL PINS							
V_{IH}	11,13	High Level Input Voltage		2			V
V_{IL}	11,13	Low Level Input Voltage				0.8	V
V_I	11,13	Input Clamp Voltage	V_{CC} = Min I_I = −18 mA			−1.5	V
I_{IH}	11,13	High Level Input Current	V_{CC} = Max V_I = 2.7V			20	μA
I_I	11,13	Input Current at Maximum Input Voltage	V_{CC} = Max V_I = 5.5V			1	mA
I_{IL}	11,13	Low Level Input Current	V_{CC} = Max V_I = 0.5V			−200	μA
V_{OH}	12,14, 15	High Level Output Voltage	V_{CC} = Min I_{OH} = −40 μA (Note 4)	2.7			V
V_{OL}	12,14, 15	Low Level Output Voltage	V_{CC} = Min I_{OL} = 800 μA (Note 4)			0.5	V
I_{OS}	12,14, 15	Output Short Circuit Current	V_{CC} = Max V_O = OV			−100	mA
I_{CC}	9	Supply Current	V_{CC} = Max		54	75	mA

AC Electrical Characteristics Over Recommended Operating Temperature and Supply Range

Symbol	Pins	Parameter	Conditions	Typ	Max	Units
DP8464B-2 t_{pp}	14	Pulse Pairing	(See Pulse Pairing Set Up)	±1.5	±3	ns
DP8464B-3 t_{pp}	14	Pulse Pairing	(See Pulse Pairing Set Up)	±2	±5	ns

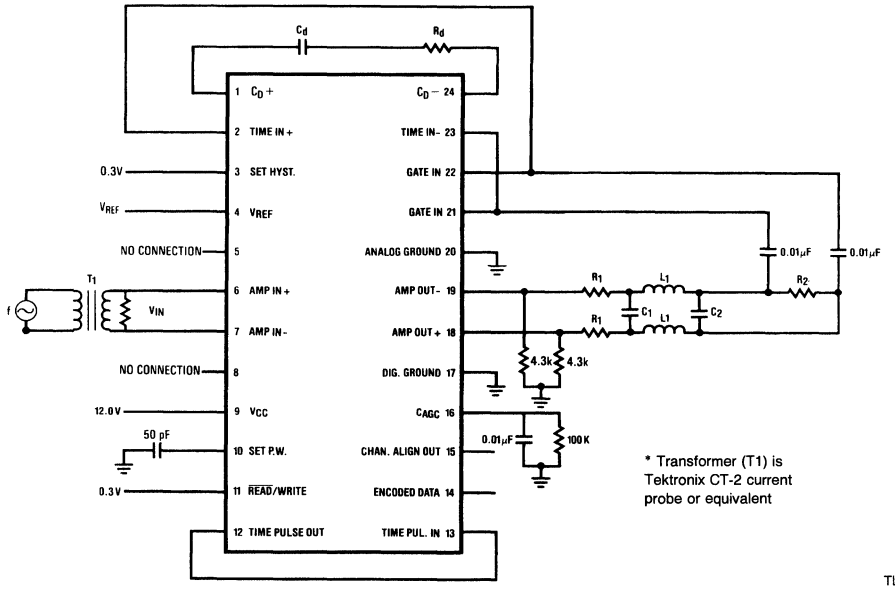
Note 1: The temperature coefficient of the input impedance is typically 0.05% per degree C.

Note 2: The AGC Threshold is defined as the voltage across the Gate Channel Input (pins 21 and 22) when the voltage on C_{AGC} (pin 16) is 4.2V.

Note 3: The Set Hysteresis Threshold is defined as the minimum differential AC signal across the Gate Channel Input (pins 21 and 22) which causes the voltage on the Channel Alignment Output (pin 15) to change state.

Note 4: To prevent inductive coupling from the digital outputs to Amp In, the TTL outputs should not drive more than one ALS TTL load each.

Pulse Pairing Set Up



TL/F/5283–3

DP8464B-2

f = 3.33 MHz and 1.25 MHz
V_{IN} = 40 mV_{pp} Differential
V_{REF} = 0.44V
C_D = 68 pF
R_D = 300Ω

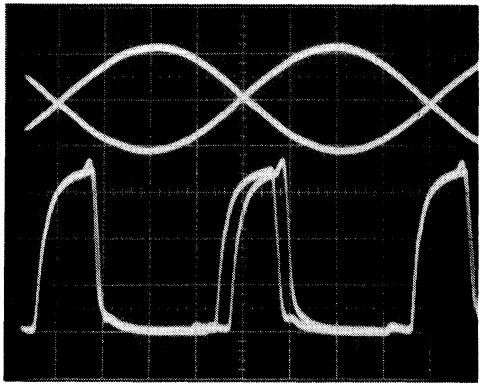
DP8464B-3

f = 2.5 MHz
V_{IN} = 40 mV_{pp} differential
V_{REF} = 0.50V
C_D = 50 pF
R_D = 430Ω

Filter

R1 = 240Ω R2 = 680Ω
C1 = 15 pF C2 = 100 pF
L1 = 4.7 μH

This is a 3 pole Bessel with the corner frequency at 7.5 MHz.



TL/F/5283–4

Pulse Pairing Measurement

Connect a scope probe to pin 14 (Encoded Data Out) and trigger off its positive edge. Adjust the trigger holdoff so the scope first triggers off the pulse associated with the positive peak and then off the pulse associated with the negative peak (as shown in the scope photo below). Pulse pairing is displayed on the second pair of pulses on the display. If the second pulses are separated by 4 ns, then the pulse pairing for this part is ±2 ns.

Circuit Operation

The output from the read/write amplifier is AC coupled to the Amp Input of the DP8464B. The amplifier's output voltage is fed back via an external filter to an internal fullwave rectifier and compared against the external voltage on the V_{REF} pin. The AGC circuit adjusts the gain of the amplifier to make the peak to peak differential voltage on the Gate Channel Input four times the DC voltage on V_{REF}. Typically the signal on Amp Out will be set for 4 Vpp differential. Since the filter usually has a 6 dB loss, the signal on the Gate Channel Input will be 2 Vpp differential. The user should therefore set 0.5V on V_{REF} which can be done with a simple voltage divider from the +12V supply.

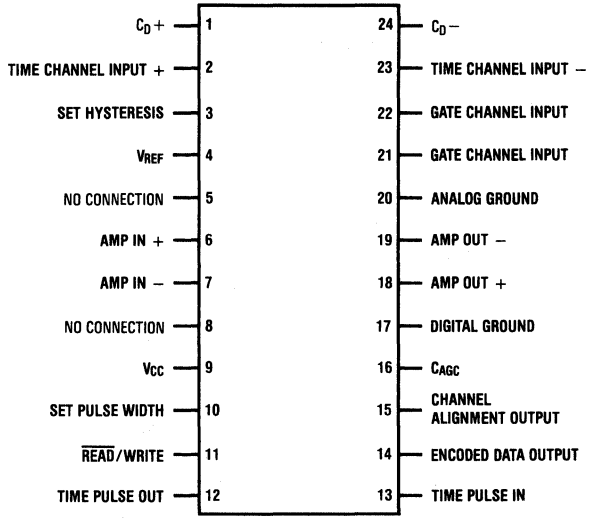
The peak detection is performed by feeding the output of the Amplifier through an external filter to the differentiator. The differentiator output changes state when the input pulse changes direction, generally this will be at the peaks. However, if the signal exhibits shouldering (the tendency to return to the baseline), the differentiator will also respond to noise near the baseline. To avoid this problem, the signal is also fed to a gating channel which is used to define a level either side of the baseline. This gating channel is comprised

Circuit Operation (Continued)

of a differential comparator with hysteresis and a D flip-flop. The hysteresis for this comparator is externally set via the Set Hysteresis pin. In order to have data out, the input amplitude must first cross the hysteresis level which will change the logic level on the D input of the flip-flop. The peak of the input signal will generate a pulse out of the differentiator and bi-directional one shot. This pulse will clock the new data at the D input through to the output. In this way, when the differentiator is responding to noise at the baseline, the output of the D flop is not changing since the logic level into the D input has not changed. The comparator circuitry is therefore a gating channel which prevents any noise near the baseline from contaminating the data. The amount of hysteresis is twice the DC voltage on the Set Hysteresis pin. For instance, if the voltage on the Set Hysteresis pin is 0.3V, the differential AC signal across the Gate Channel Input must be larger than 0.6V before the output of the comparator will change states. In this case, the hysteresis is 30% of a 2V peak to peak differential signal at the gate channel input.

Connection Diagrams

Dual-In-Line (DIP) Package

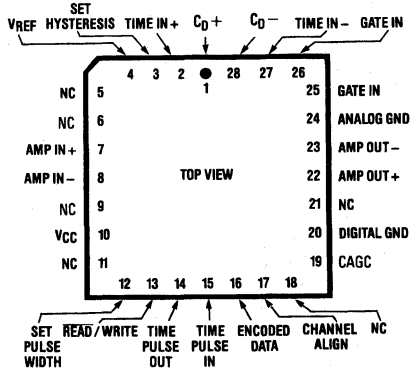


Top View

TL/F/5283–2

Order Number DP8464BN-3 or DP8464BN-4
See NS Package N24C

Plastic Chip Carrier (PCC) Package



TL/F/5283–30

Order Number DP8464BV-3 or DP8464BV-4
See NS Package V28A

Pin Definitions

(All pin numbers refer to the 24 pin dual-in-line package)

Pin #	Name	Function
Power Supply		
9	V_{CC}	The supply is $+12V \pm 10\%$.
17	Digital Ground	Digital signals should be referenced to this pin.
20	Analog Ground	Analog signals should be referenced to this pin.
Analog Signals		
6 7	Amp In+ Amp In−	These are the differential inputs to the Amplifier. The output of the read/write head amplifier should be capacitively coupled to these pins.
18 19	Amp Out+ Amp Out−	These are the differential outputs of the Amplifier. These outputs should be capacitively coupled to the gating channel filter (if required) and to the time channel filter.
22 21	Gate Channel Inputs	These are the differential inputs to the AGC block and the gating channel. These inputs must be capacitively coupled from the Amp Out.
2 23	Time Channel Input+ Time Channel Input−	These are the differential inputs to the differentiator in the time channel. In most applications, a filter between the Amp Out (pins 18 and 19) and these inputs is required to band limit the noise and to correct for any phase distortion introduced by the read circuitry. In all cases this input must be capacitively coupled to prevent disturbing the DC input level.
1 24	C_d+ C_d−	The external differentiator network is connected between these two pins.
3	Set Hysteresis	The DC voltage on this pin sets the amount of hysteresis on the differential comparator. Typically this voltage can be established by a simple resistive divider from the positive supply.
4	V_{REF}	The AGC circuit adjusts the gain of the amplifier to make the differential peak to peak voltage on the Gate Channel Input equal to four times the DC voltage on this pin. This voltage can be established by a simple resistive divider from the positive supply.
5 8	No connection No connection	
16	C_{AGC}	The external capacitor for the AGC is connected between this pin and Analog Ground.

Pin #	Name	Function
Digital Signals		
10	Set Pulse Width	An external capacitor to control the pulse width of the Encoded Data Out is connected between this pin and Digital Ground.
11	$\overline{\text{Read}}$/Write	If this pin is low, the Pulse Detector is in the read mode and the chip is active. When this pin goes high, the pulse detector is forced into a standby mode. This is a standard TTL input.
12	Time Pulse Out	This is the TTL output from the bi-directional one shot following the differentiator. In most applications this can be connected directly to the Time Pulse In.
13	Time Pulse In	This is the TTL input to the clock of the D flip-flop. Usually this is connected directly to the Time Pulse Out pin.
15	Channel Alignment	This is the buffered output of the differential comparator with hysteresis. This is usually used in the initial system design and is not used in production.
14	Encoded Data Out	This is the standard TTL output whose leading edge, indicates the time position of the peaks.

Application Information

GENERAL DESCRIPTION

All pin numbers refer to 24 pin dual-in-line package.

The DP8464B Disk Pulse Detector utilizes analog and digital circuitry to detect amplitude peaks of the signal received from the Read/Write Amplifier. The analog signal from a disk is a series of pulses, the peaks of which correspond to 1's or flux reversals on the magnetic medium. The pulse detector must accurately determine the time position of these peaks. The peaks are indicated by the positive leading edge of a TTL compatible output pulse. This task is complicated by variable pulse amplitudes depending on the media type, head position, head type and read/write amplifier circuit gain. Additionally, as the bit density on the disk increases, the amplitude decreases and significant bit interaction occurs resulting in pulse distortion and shifting of the peaks.

The graph in *Figure 1* shows how the pulse amplitude varies with the number of flux reversals per inch (or recording density) for a given head disk system. The predominant disk applications are associated with the first two regions on this graph, Regions 1 and 2. Typical waveforms received by the pulse detector for these regions are shown next to the graph.

Application Information (Continued)

Region 1 is the high resolution area characterized by a large spread between flux reversals and a definite return to baseline (no signal) between these peaks. Pulses of this type are predominantly found in drives which use either thin film heads or plated media, or in drives which utilize run length limited codes (like the 2,7 code) which spread the distance between flux reversals.

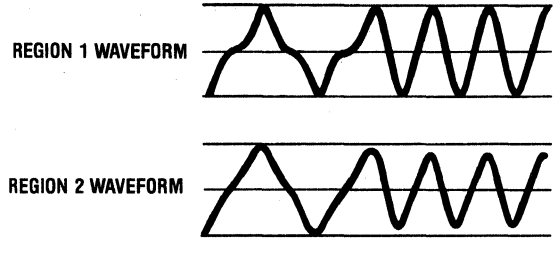
A Region 2 waveform will vary from a tendency to return to the baseline (called shouldering) to almost sinusoidal at the higher frequencies. These pulses come from drives which use limited frequency codes (such as MFM). The pulses may contain shouldering on the outer tracks of the disk and be nearly sinusoidal on the inner tracks since the flux density increases towards the inner track.

Detecting pulse peaks of waveforms of such variable characteristics requires a means of separating both noise and shouldering-caused errors from the true peaks. In the past, mild shoulder-caused errors were blocked by self-gating circuits (such as the "de-snaker"). These circuits fail when shouldering is extensive, hence the need for the DP8464B which includes a peak sensing circuit and an amplitude sensitive gating channel in parallel.

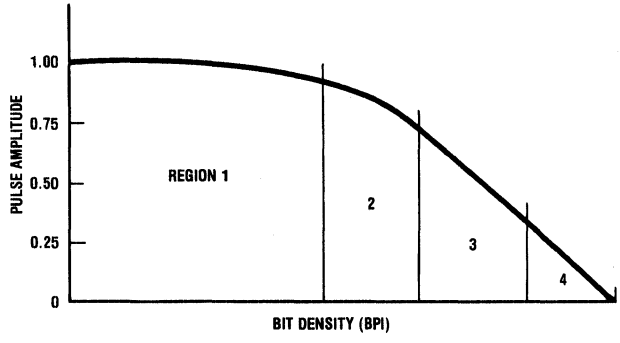
The main circuit blocks of the DP8464B are shown in *Figure 2*. The output from the read/write amplifier is fed directly to the Amp Input of the DP8464B. This is the input of a Gain Controlled Amplifier. The amplifier's output voltage is fed back via an external filter to an internal fullwave rectifier and compared against the external voltage on the V_{REF} pin. The AGC circuit adjusts the gain of the amplifier to make the peak-to-peak differential Gate Channel input voltage four times the DC voltage on V_{REF}.

The peak detection is performed by feeding the output of the Gain Controlled Amplifier through an external filter to the differentiator. The differentiator output changes state when the input pulse changes direction, generally this will be at the peaks. However, if the signal exhibits shouldering (the tendency to return to the baseline) as seen in Region 1 and the upper part of Region 2, the differentiator will also respond to noise near the baseline. To avoid this, the signal is also fed to a gating channel which is used to define a level either side of the baseline. This gating channel comprises a differential comparator with hysteresis and a D flip-flop. The hysteresis for this comparator is externally set via the Set Hysteresis pin. In order to have valid data out, the input amplitude must first cross the hysteresis level. This will change the logic level on the D input of the flip-flop. The peak of the input signal will generate a pulse out of the differentiator and bi-directional one shot. This pulse will clock in the new data on the D input, which will appear at the Q output. In this way, when the differentiator is responding to noise at the baseline, the output of the D flop is not changing since the logic level into the D input has not yet changed. The comparator circuitry is therefore a gating channel to prevent any noise near the baseline from contaminating the data.

The amount of hysteresis is twice the DC voltage on the Set Hysteresis pin. For instance, if the voltage on the Set Hysteresis pin is 0.3V, the differential Gate Channel Input must be larger than 0.6V ($\pm$0.3V) before the output of the comparator will change states. The Time Pulse Out, Encoded Data, and Channel Alignment Output are designed to drive 1 standard TTL gate.

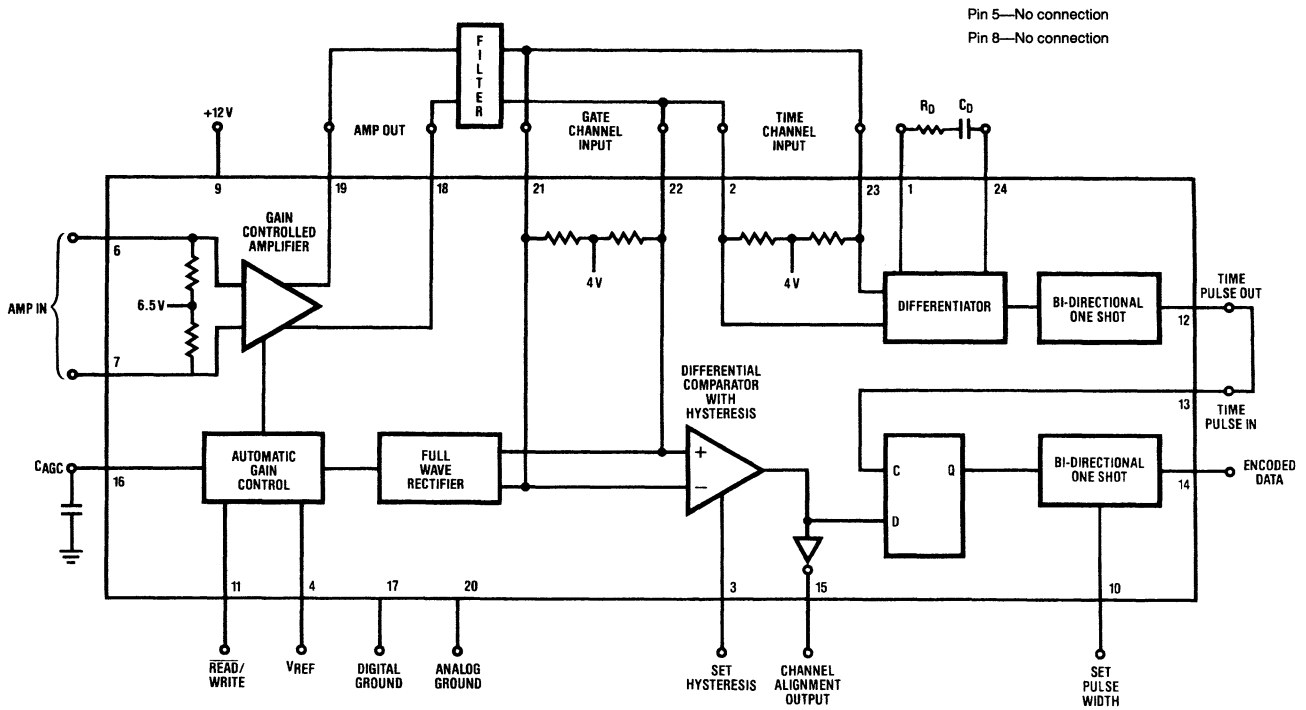


TL/F/5283–5



TL/F/5283–6

FIGURE 1. Pulse Amplitude vs. Bit Density with Typical Waveforms



TL/F/5283–7

FIGURE 2. DP8464B Block Diagram, Region 1 Connection

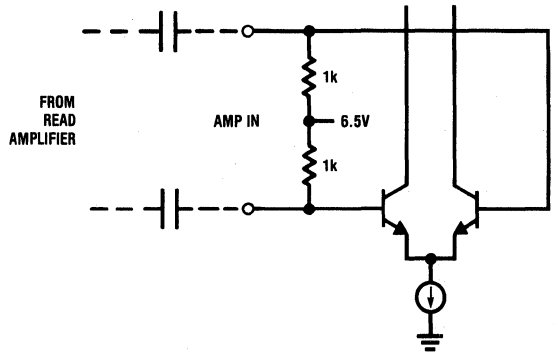
Application Information (Continued)

GAIN CONTROLLED AMPLIFIER

The purpose of the Gain Controlled Amplifier is to increase the differential input signal to a fixed amplitude while maintaining the exact shape of the input waveform. The Gain Controlled Amplifier is designed to accept input signals from 20 mVpp to 660 mVpp differential and amplify that signal to 4 Vpp differential. The gain is therefore from 6 to 200 and is controlled by the automatic gain control (AGC) loop. The amplifier output is actually capable of delivering typically 5 Vpp differential output but the parts are only tested and guaranteed to 4 Vpp.

The input to the Gain Controlled Amplifier is shown in *Figure 3*. The value of the input capacitors should be selected so that the pole formed by the coupling capacitor and the 1k bias resistor is a factor of 10 lower than the lowest signal frequency. These input bias resistors have a ±20% tolerance and a temperature coefficient of 0.05% per degree C. When the pulse detector is in the write mode, these bias resistors are automatically shunted by 425Ω resistors. This allows the input circuit to recover quickly from the large transients encountered during a write to read transition. The input impedance to the amplifier is therefore 1k during read operations and 300Ω during write operations.

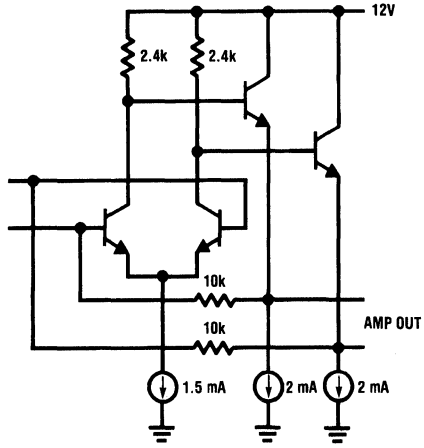
The output of the Gain Controlled Amplifier is shown in *Figure 4*. The outputs are biased at (12V − (0.75 mA × 2.4k) − 0.75V) or 9.5V. Since each output will swing ±1V (4 Vpp differential), each output pin will swing from 8.5V to 10.5V. If the total differential load placed on the output is 1k, (see *Figure 5*) then the circuit must supply 2V/1k or 2 mA. Since the output is class A, external resistors to ground must be used to provide the sink current. In this case, in order to sink 2 mA at the lowest voltage, then (8.5V/2 mA) or an external 4.3k resistor from each output to ground is required. Note that the circuit has additional margin since the internal 2 mA current sources were not included in the calculation. Typically the output impedance of the Gain Controlled Amplifier is 17Ω, and the −3 dB bandwidth is greater than 20 MHz.



TL/F/5283–8

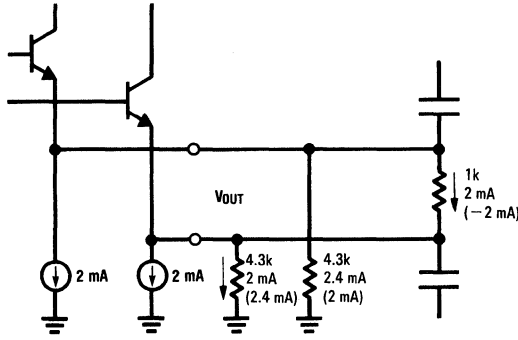
FIGURE 3. Input to Gain Controlled Amplifier

Application Information (Continued)



TL/F/5283–9

FIGURE 4. Output of Gain Controlled Amplifier



TL/F/5283–10

FIGURE 5. Output Stage with 1k Differential Load

Application Information (Continued)

AUTOMATIC GAIN CONTROL (AGC)

The Automatic Gain Control holds the signal level at the Gate Channel Input at a constant level by controlling the gain of the Gain Controlled Amplifier. This is necessary because the amplitude of the input signal will vary with track location, variations in the magnetic film, and differences in the actual recording amplitude. The Gain Controlled Amplifier is designed for a maximum 4 Vpp differential output. To prevent the Gain Controlled Amplifier from saturating, the V_{REF} level must be set so the maximum amplifier output voltage is 4 Vpp. The AGC will force the differential peak-to-peak signal on the Gate Channel Input to be four times the voltage applied to the V_{REF} pin. Normally some kind of filter is connected between the Gain Controlled Amplifier's output and the Gate Channel Input. Typically this filter has a 6 dB insertion loss in its pass band. Since the AGC holds the amplitude at the Gate Channel Input constant, this 6 dB loss through the Gate Channel filter will cause the Gain Controlled Amplifier's output to be 6 dB larger than the Gate Channel Input.

The AGC loop starts out in the high gain mode. When the input signal is larger than expected, the AGC loop will quickly reduce the amplifier gain so the peak-to-peak differential voltage on the Gate Channel Input remains four times the voltage on V_{REF}. If the input amplitude suddenly drops, the AGC loop will slowly increase the amplifier gain until the differential peak-to-peak Gate Channel Input voltage again reaches four times V_{REF}. The AGC loop requires several peaks to react to an increased input signal. In order to recover the exact peak timing during this transition, the V_{OUT} level must be set somewhat lower than the maximum of 4 Vpp. For instance, if the V_{REF} is 0.5V, and if the loss in the gate channel filter is 6 dB, then the Amp Output is 4 Vpp. If the Amp Input suddenly increases 30%, the amplifier may saturate and the timing for a few peaks may be disturbed until the AGC reduces the amplifier gain. If the peak detection is critical during this time, the system may fail. The proper operation, for this example, is to set the V_{REF} at 0.35V so the amplifier will not saturate if the input suddenly increases 30%.

A simplified circuit of the AGC block is shown in *Figure 6.* When the full wave rectified signal from the Gate Channel Input is greater than V_{REF}, the voltage on the collector of transistor T1 will increase and charge up the external capacitor C_{AGC} through T2. The typical available charging current is 2.5 mA. Conversely, if this input is less than V_{REF}, transistor T2 will be off, so the capacitor C_{AGC} will be discharged by the base current going into the Darlington T3 and T4. This discharge current is approximately 1 μA. The voltage across C_{AGC} controls the gain of the Gain Controlled Amplifier. This voltage will vary from typically 3.4V at the highest gain to 4.5V at the lowest gain.

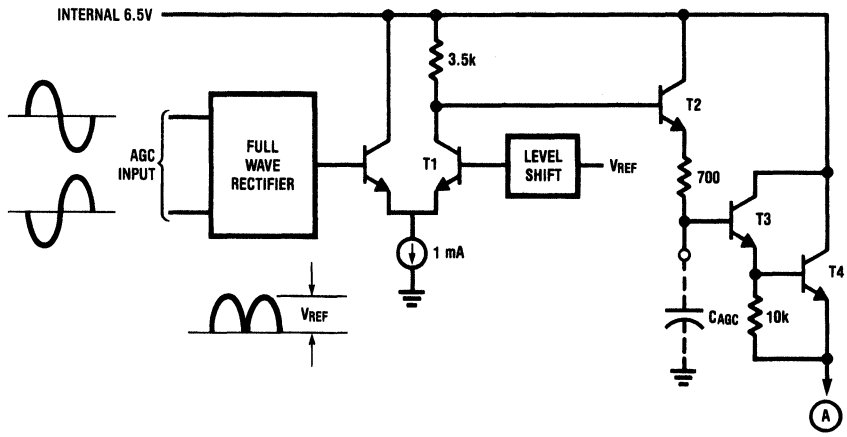
When the AGC circuit has not received an input signal for a long time, the base current of the Darlington will discharge the external C_{AGC} to 3.4V. The amplifier will now be at its highest gain. When a large signal comes in, the external C_{AGC} will be charged up with the 2.4 mA from T2 thereby reducing the gain of the amplifier. The formula, $I = C \times (dV/dt)$ can be used to calculate the time required for the amplifier to go from a gain of 200 to a gain of 6. For instance, if $C_{AGC} = 0.01$ μf, the charging current I is 2.4 mA, and the dV required for the amplifier to go through its gain range is 1.1V, then

$$dt = (0.01\ \mu F \times 1.1V)/(2.4\ mA) \text{ or } 4.6\ \mu s.$$

In reality, the gain does not change this quickly since the C_{AGC} would only be charging during a portion of the input waveform.

By using the same argument, the time required to increase the amplifier gain after the input has been suddenly reduced can be calculated. This time, the discharging current is only 1 μa so

$$dt = (0.01\ \mu F \times 1.1V)/1\ \mu A) \text{ or } 11\ ms.$$



TL/F/5283–11

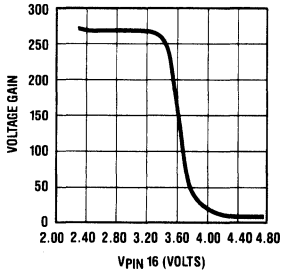
FIGURE 6. Simplified AGC Circuit

Application Information (Continued)

This time can be decreased by placing an external resistor across the C_{AGC}. For instance, if a 100k resistor is placed in parallel with C_{AGC}, then the discharge current is 40 μA. The time required to increase the amplifier gain is now 40 times faster or 275 μs. If this external resistor is made even smaller, say 10k, then the discharge time will go to 27.5 μs. Now however, there is another problem introduced. The response time of the AGC is so fast that it distorts the signal at the output of the Gain Controlled Amplifier. Distortion of the signal at the Amplifier Output can affect the time position of the peaks of this signal. Be sure to check this distortion over the range of input levels you expect to encounter, when choosing the external R and C values for the AGC.

If the value of the bleed resistor across the C_{AGC} is decreased (in order to equalize the AGC attack and decay times) the value of C_{AGC} must be increased in order to maintain an AGC response that does not distort the signal. There is a second order effect on the amplitude that results from this attack and decay time equalization. Referring to *Figure 2*, notice that the AGC is driven from a full wave rectified version of the Gate Channel Input signal. When the AGC is operated normally (ie. fast attack and slow decay) the voltage that appears across C_{AGC} is the peak detected value of this full wave rectified waveform. However, if you equalize the AGC attack and decay times the voltage across C_{AGC} is the RMS voltage (0.707 times the peak) of the full wave rectified waveform. Thus, the voltage across C_{AGC} is less and the amplitude out of the Gain Controlled Amplifier will consequently be 1.4 times larger.

It is possible to externally drive the C_{AGC} pin to control the gain of the amplifier. It must be noted that the gain of the amplifier is not always exactly 200 when the voltage on C_{AGC} is 3.4V. The transfer curve between the gain of the amplifier and the voltage on C_{AGC} is only approximate. This transfer curve will vary between parts and with temperature. Care should be taken to prevent the voltage on the C_{AGC} pin from going below ground or above 5.5V. *Figure 7* shows a typical curve of the Gain Controlled Amplifier Gain vs. the voltage across C_{AGC} (Vpin 16.)

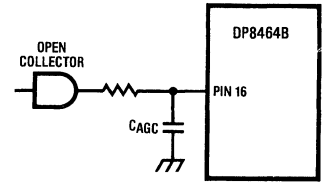


TL/F/5283–12

FIGURE 7. Gain Controlled Amplifier Gain vs. Vpin 16

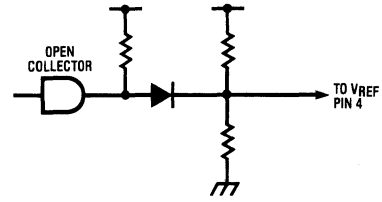
It is possible to change the time constant of the AGC circuit by switching in different external components at the desired times. For instance, as shown in *Figure 8*, an external open collector TTL gate and resistor can be added in parallel with C_{AGC} to decrease the AGC response time. Similarly, an external capacitor could be switched in to increase the response time. Since in the absence of an external resistor the discharge time of C_{AGC} is much longer than the attack time there may be some applications where it is desirable to switch in a parallel resistor to quickly discharge C_{AGC} then switch it out to force a quick attack. Because of the quick attack time, the AGC obtains the proper level quicker than it would had C_{AGC} simply been allowed to discharge to the new level.

There are some applications where it is desirable to hold the AGC level for a period of time. This can be done by raising the $\overline{\text{READ}}$/WRITE pin. This will shut off the input circuitry, and it will take time (about 2.5 μs) for the circuit to recover when going back into the read mode. *Figure 9* shows a method to hold the AGC level while remaining in the read mode (which could be used in embedded servo applications). If the voltage on V_{REF} is raised to 3V, then the amplifier output voltage cannot get large enough to turn on the circuitry to charge up C_{AGC}. For this to work properly, there can not be a large discharge current path (resistor in parallel with C_{AGC}) across C_{AGC}. The AGC block can be bypassed altogether by connecting V_{REF} to 3V. In this way, the user can use his own AGC circuit to drive the C_{AGC} pin directly.



TL/F/5283–13

FIGURE 8. Circuit to Decrease AGC Response Time



TL/F/5283–14

FIGURE 9. Circuit for AGC Hold

$\overline{\text{READ}}$/WRITE

In the normal read mode, the signal from the read/write head amplifier is in the range of 20 mVpp to 660 mVpp. However, when data is being written to the disk, the signal coming into the analog input of the pulse detector will be on the order of 600 mV. Such a large signal will disturb the AGC level and would probably saturate the amplifier. In addition, if a different read/write amplifier is selected, there will be a transient introduced because the offset of the preamplifiers are not matched. A $\overline{\text{READ}}$/WRITE input pin has been provided to minimize these effects to the pulse detector. This is a standard TTL input.

When the $\overline{\text{READ}}$/WRITE pin is low, the pulse detector is in the read mode. When the $\overline{\text{READ}}$/WRITE pin is taken high, three things happen. First, the 1k resistors across the AMP IN pins are shunted by 300Ω resistors, as described previously in the Gain Controlled Amplifier section. Next, the amplifier is squelched so there is no signal on the Amp Output.

Application Information (Continued)

Finally, the previous AGC level is held. This AGC hold function is accomplished by not allowing any current to charge up the external C_{AGC}. The voltage across this capacitor will slowly reduce due to the bias current into the Darlington (see *Figure 6*) or through any resistor placed in parallel with C_{AGC}. Therefore, as described in the Automatic Gain Control section, the gain of the amplifier will slowly increase. All of these three events happen simultaneously.

When the $\overline{\text{READ}}$/WRITE input is returned low, the pulse detector will go back to the read mode in a specific sequence. First of all, the input impedance at the Amp In is returned to 1k. Then, after approximately 1 μs, the Gain Controlled Amplifier is taken out of the squelch mode, and finally approximately 1 μs after that, the AGC circuit is turned back on. This return to the read mode is designed to minimize analog transients in order to provide stable operation after 2.5 μs. It is very important that the analog input be stable before the chip is returned to the read mode. It is recommended that other than when writing, the Pulse Detector be in the read mode at all times in order to prevent the 2.5 μs delay from slowing up the system. The $\overline{\text{READ}}$/WRITE pin may be connected to the Write Gate output of a controller (such as the DP8466 Disk Data Controller).

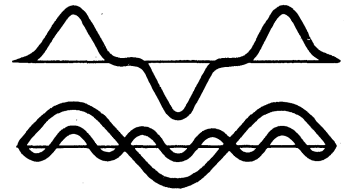
TIME CHANNEL FILTER

The peak detection is performed by feeding the output of the Gain Controlled Amplifier through an external filter to the differentiator. The differentiator output changes state when the input pulse changes direction, generally this will be at the peaks. The differentiator can also respond to noise near the baseline, in which case the comparator gating channel will inhibit the output. The purpose of the external filter is to bandwidth limit the incoming signal for noise considerations. Care must be used in the design of this filter to ensure the delay is not a function of frequency. For this reason, a high order Bessel filter with its constant group delay characteristics can be used in this application. Often, this filter must be specifically designed to correct errors introduced by the non-ideal phase characteristics of the input read head. The typcial -3 dB point for this filter is around 1.5 times the highest recorded frequency. The design of this filter is complex and will not be discussed here. However, the following discussion does give a feel for some of the considerations involved in the filter design. The reader is referred to reference #3 listed at the end of the Applications Notes for further filter design information.

Figure 10 shows a typical Region 1 waveform where there is no bit interaction. This waveform is primarily the sum of the fundamental frequency and its 3rd harmonic (higher odd harmonics are present when there is more shouldering).

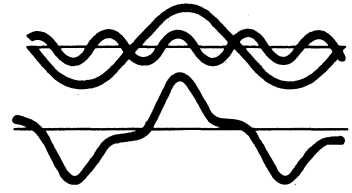
If the filter is to preserve this wave shape (this would be the case if no read/write head phase compensation were necessary) then the phase relationship between the fundamental frequency and its harmonics must not be altered. *Figure 11* shows the output when the 3rd harmonic has the proper magnitude, but the phase relationship is not maintained. The result is that the output waveform is not the same shape as the input (in a severe case it may be almost unrecognizable) and the time position of the peaks has been altered.

One electrical parameter which describes how well a filter will preserve a wave shape is called group delay. Group delay is defined as the change in phase divided by the change in frequency. If the group delay is constant over the



TL/F/5283–15

FIGURE 10. Typical Region 1 Waveform



TL/F/5283–16

FIGURE 11. Region 1 Waveform with the Incorrect Phase Relationship

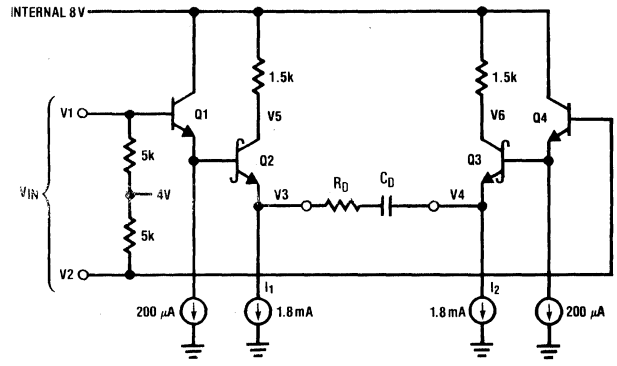
frequencies of interest, then the wave shape will be maintained. An MFM coded signal will contain three basic frequency components for the various digital patterns of data. For instance, a 10 Megabit/sec MFM signal will consist of analog frequencies of 2.5 MHz, 3.33 MHz and 5 MHz. On the outer track the bit density is the lowest and the 5 and 3.33 MHz signals will look sinusoidal while the 2.5 MHz signal will have a tendency to return to the baseline. This returning to the baseline is called shouldering and is illustrated in *Figure 10*. Since this shouldering is rich in 3rd harmonic—the 2.5 MHz signal will have a strong 7.5 MHz component. The 10 Megabit/sec MFM signal will therefore have 2.5 MHz, 3.33 MHz, 5 MHz, and 7.5 MHz components which must be filtered with constant group delay in order to reproduce the original waveform. For example, if the phase shift through the filter at 2.5 MHz is 33.3°, then at 3.33 MHz the phase shift must be 44.3°, at 5 MHz—66.6°, and at 7.5 MHz—99.9°. The group delay $\frac{d\theta}{dt}$ for this case is 13.32°/MHz. This can be better interpreted as a time delay. 33.3° of a 2.5 MHz signal is equivalent to (33.3/360) $\times$ (1/2.5 MHz) or 37 ns. Similarly, 66.6° on a 5 MHz signal is (66.6/360) $\times$ (1/5 MHz) $=$ 37 ns.

The third order Bessel Filter as shown in the 10 Mbit/sec. pulse pairing measurement board on the data sheet is designed for a constant group delay and a -3 dB point of 7.5 MHz. At this frequency the delay through the filter is 35 ns. The Gain Controlled Amplifier of the DP8464B is designed for a group delay of a 7.8 ns $\pm$0.5 ns for frequencies up to 7.5 MHz. The 7.8 ns delay in the Gain Controlled Amplifier and the 37 ns delay in the Bessel Filter do not introduce any timing error, only a delay of 44.3 ns from the Amp Input to the output of the filter.

DIFFERENTIATOR

A simplified circuit of the first stage of the differentiator is shown in *Figure 12*. The voltages at V3 and V4 are simply two diodes down from V1 and V2. Therefore the voltage

Application Information (Continued)



TL/F/5283–17

FIGURE 12. Simplified Differentiator First Stage

across the external differentiator network (C_d in series with R_d) is the differential input voltage V1 − V2. When R_d is zero, the current through C_d is $I = C \times (dV/dt)$ or $C_d \times (dV_{IN}/dt)$. The Q2 collector current is the sum of the 1.8 mA current source plus the current through C_d or

$$1.8\ mA + C_d \times (dV_{IN}/dt).$$

Similarly, the Q3 collector current is

$$1.8\ mA - C_d \times (dV_{IN}/dt).$$

Therefore, the differentiator output voltage, V5 − V6, is

$$1.5k \times 2 \times C_d \times (dV_{IN}/dt).$$

The input is at a peak when V5 − V6 = 0V.

The differentiator network (C_d and R_d) should be selected so the maximum current into the differentiator network is not greater than the minimum current of I1 and I2 over temperature. In the electrical specifications, the minimum current is specified for 1.4 mA (I_{Cd} Current into Pin 1 and 24 that discharges C_d). For example, the highest analog frequency in a 10 Megabit/sec, MFM signal is 5 MHz. Since the AGC loop has forced the input to the differentiator to 2 V_{PP} (which includes the 6 dB loss of the filter), then the voltage across the capacitor (assuming R_d is 0) is:

$$V_{IN} = 1 \times \sin(2 \times \pi \times 5E6 \times t)$$

and

$$dV_{IN}/dt = 1 \times 2 \times \pi \times 5E6 \times \cos(2 \times \pi \times 5E6 \times t)$$

and the maximum slope is

$$(dV_{IN}/dt)max = 1 \times 2 \times \pi \times 5E6 = 314E5\ V/sec.$$

For this example, C_d can now be calculated. Since $I = C \times (dV/dt)$, then for I = 1.4 mA, dV/dt = 314E5, then the maximum C_d must equal 45 pF. From this example, a following simple design equation for the value of C_d can be derived.

$$C_d = 445/(V_{IN} \times f_{max})$$

where

C_d is the maximum external differentiator capacitor in pF

V_{IN} is the peak to peak differential Time Channel input voltage

f_{max} is the maximum analog frequency in MHz

Note that this is the maximum value for the capacitor when the series resistor R_d is zero. The value of the capacitor can be increased if a series resistor is used, but the maximum current through the differentiator network must not exceed 1.4 mA. If too large a value for C_d is used, the delay through the differentiator will become dependent on frequency. This will not show up in a single frequency test such as a test for pulse pairing.

For the MFM code, the maximum analog frequency is ½ the data rate. For the ½(2,7) code, the maximum analog frequency is ⅓ the data rate. The above sinusoidal analysis is valid as long as the highest frequency on the outer track is nearly sinusoidal. If, however, there is significant shouldering of this signal then the value of C_d should be reduced accordingly.

The following table summarizes the value of C_d to use for a 2 V_{pp} differential signal to the time channel input.

Data Rate	Code	Maximum Frequency	C_d
5 mbits/sec	MFM	2.5 MHz	90 pF
5 mbits/sec	2,7	1.6 MHz	140 pF
10 mbits/sec	MFM	5.0 MHz	45 pF
10 mbits/sec	2,7	3.3 MHz	67 pF

As noted above, the value of the capacitor can be increased if a series resistor is used, but the maximum current through the differentiator network must not exceed 1.4 mA. For example, the components used in the Pulse Pairing Setup (see AC Electrical Specifications) are for a typical 10 Mbits/sec MFM drive. The combination of the C_d of 50 pF and the R_d of 430Ω gives a combined impedance of 768Ω at the highest frequency of 5 MHz. This gives a maximum current of 1.3 mA—well below the 1.4 mA limit.

A resistor is placed in series with C_d in order to bandlimit the differentiator response. This resistor also has an effect on the phase linearity of the differentiator. An ideal differentiator produces an output that is 90 degree phase shifted from the input regardless of the input frequency. The presence of the series resistor produces an output phase shift that is less than 90 degrees and changes with the input frequency. This resistor can be used to correct for frequency related phase problems encountered elsewhere in the read path.

Application Information (Continued)

To properly decode the information on the disk, the read channel must determine if there is a peak (or a "1") during a period of time called a detection window. The detection window for MFM and the (2,7) code is

$$1/(2 \times \text{data bit rate}).$$

This detection window must accommodate errors in many parts of the system including filters, data separator, and peak shift variations in the data pattern. The pulse pairing of the DP8464B should be included in the error budget calculation.

Unequal delays through the bi-directional one shots will contribute to pulse pairing. To minimize this effect, pin 2 should be connected to 22 and pin 23 should be connected to 21. If connected this way, the delays tend to cancel. For the PCC Package, Pin 26 to Pin 2, and Pin 25 to Pin 27.

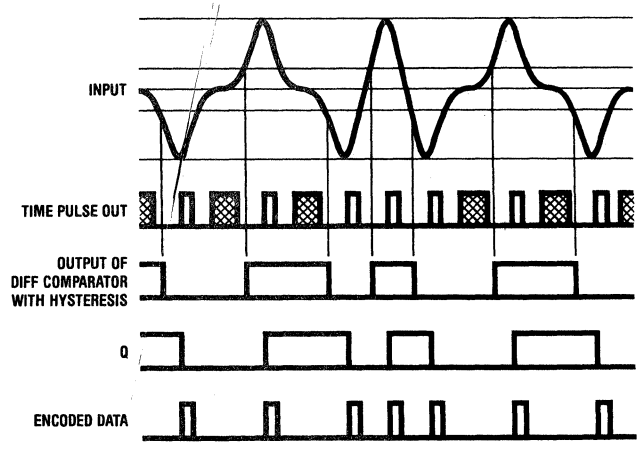
DIFFERENTIAL COMPARATOR WITH HYSTERESIS

The actual peak detection is done in the time channel with the differentiator. Unfortunately, the differentiator not only responds to signal peaks but also responds to noise at the baseline. In order to prevent this noise from generating false data, the signal at the output of the Gain Controlled Amplifier is also passed through a gating channel which prevents any output change before the input signal has crossed an established level. This gating channel comprises a differential comparator with hysteresis and a D flip-flop. The hysteresis for this comparator is set externally via the Set Hysteresis pin. The amount of hysteresis is twice the voltage on the Set Hysteresis pin. For instance, if the voltage on the Set Hysteresis pin is 0.3V, the differential input signal must be larger than 0.6V($\pm$0.3V) before the output of the comparator will change states. The 0.6V hysteresis represents 30% of a typical 2V differential input signal level to the gating channel. The hysteresis level is usually set between 15% to 40% of the differential input signal.

The operation of the gating channel is shown in *Figure 13*. At the top is a typical Region 1 waveform which exhibits shouldering on the lowest frequency and is almost sinusoidal on the highest frequency. In this example, this waveform is fed to both the timing and the gating channel. The hysteresis level (of about 25%) has been drawn on this waveform. The second waveform is the output of the differentiator and its bi-directional one shot. This is the waveform on the Time Pulse Out pin. While there is a positive edge pulse at each peak, there is also noise at the shoulders. In this example, the Time Pulse Out is connected directly to the Time Pulse In without any external delay. This output is therefore the clock for the D flip-flop.

The third waveform in *Figure 13* is the output of the Comparator with Hysteresis which goes to the D input of the flip-flop. The true peaks are the first positive edges of the Time Pulse Out which occur after the output of the comparator has changed states. The D flip-flop will "clock" in these valid peaks to the output bi-directional one shot. Therefore, the noise pulses (due to the differentiator responding to noise at the baseline) just "clock" in the old data through the flip-flop and the output does not change.

The Q output of the flip-flop drives the output bidirectional one-shot which generates the positive edges corresponding to the peaks. The width of the data pulses can be controlled by an external capacitor from the Set Pulse Width pin to ground. This pulse width can be adjusted from 20 ns to ½ the period of the highest frequency. Typical values for this capacitor are 20 pF for a 25 ns pulse width to 100 pF for a 100 ns pulse.



TL/F/5283–19

FIGURE 13. Time and Gate Channel Operation for Region 1 Signals

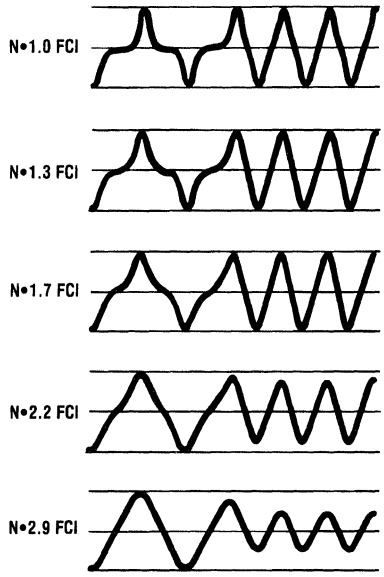
Application Information (Continued)

PULSE DETECTOR OPERATION IN REGIONS 1 AND 2

Figure 14 shows the input waveform for the lowest frequency followed by the highest frequency for an MFM code. In MFM the highest frequency is twice the lowest frequency. The outer track has the least flux changes per inch (FCI) and is illustrated in the waveforms at the top. There is so much room between the pulses that the signal returns to the baseline for the lowest frequency while there is shouldering at the highest frequency. As you go towards the inner track, the pulses become more crowded and bit interaction occurs. At the third curve down (N × 1.7 FCI), there is shouldering at the lowest frequency while the highest frequency is almost sinusoidal. At higher bit densities, the lowest frequency looks sinusoidal, while the highest frequency is decreasing in amplitude. In *Figure 14*, the first three waveforms are examples of Region 1 operation (very little change in amplitude with frequency). The last two waveforms are examples of Region 2 operation.

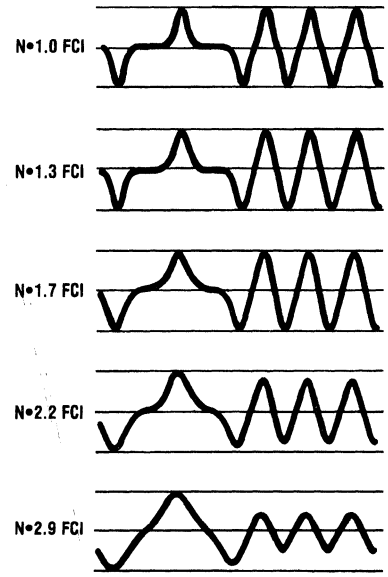
In a disk system, the bit density changes about a factor of 1.7 between the inner and the outer track. For instance, if the input waveform for the F-2F signal on the inner track of a system looks similar to waveform #4 in *Figure 14* (N × 2.2 FCI), then the outer track will have a bit density that is approximately N × 2.2/1.7 or N × 1.3 FCI. This is shown in the second waveform. Tracks half the way in will have a bit density of the average between the inner and outer tracks, in this case N × 1.7 FCI which is illustrated in the third waveform. Note that the analog waveforms change considerably with track location. Self-gating circuits ("desnakers") can be used in MFM systems which operate in these last three curves (from N × 1.7 FCI to N × 2.9 FCI). If the FCI becomes much less, the shouldering on the lowest frequency will let in too much noise. If the FCI is increased, the peak resolution gets very poor. Now we can compare these waveforms to longer run length limited codes.

Figure 15 shows the analog waveform for the lowest frequency followed by the highest frequency for a 2,7 code. In the 2,7 code, the frequency range is from F to 2.66 × F. Unlike the MFM code, there is no region where the self-gating "desnaker" will work on both the inner and outer tracks.



TL/F/5283–20

FIGURE 14. MFM F-2F Pulse Waveforms for Various Flux Changes per Inch



TL/F/5283–21

FIGURE 15. F-2.66 × F Pulse Waveforms for Various Flux Changes per Inch

Application Information (Continued)

The simplest operation is for systems operating entirely in Region 1, that is, no amplitude reduction between the highest and the lowest frequency at the inner track. The inner track is specified because the pulse interaction is most severe on the inner track. For Region 1 operation, only the Time Channel filter is required, so the Gate Channel Input is connected to the Time Channel Input. Since no external time delay is required to align the time and gate channels, the Time Pulse Out is connected directly to the Time Pulse In. The Region 1 connection is shown in *Figure 2*. The internal timing for this operation is shown in *Figure 13*.

If there is significant amplitude reduction at the highest frequency, the peak detection becomes more complex. If the worst case waveform is like the fourth waveform on *Figure 14*, then the Region 1 connection might still work satisfactorily. However, if the input begins to approach the fifth waveform, this system configuration will completely fail. One problem is that the AGC will respond to the frequency dependent amplitude modulation and distort the waveform.

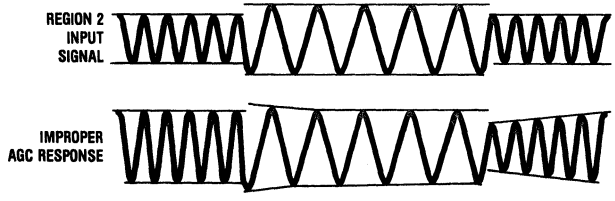
Figure 16 illustrates this problem which is encountered in systems operating in Region 2. If the input digital pattern suddenly shifts from a high frequency to a low frequency, the bit density may shift from the 70% level on the BPI curve of *Figure 1* to a point at 90% on the BPI curve. As shown, the AGC loop is correcting for this frequency-induced change in amplitude by quickly decreasing the amplifier gain. The situation gets worse if the input digital pattern shifts back to a high frequency. The AGC loop now cannot quickly increase the amplifier gain, so the output waveform will very slowly increase. The AGC response to frequency related amplitude change is not desirable since the AGC is now distorting the input waveform. This can be prevented by inserting a lead network between the Gain Controlled Amplifier's output and the AGC input, as shown in *Figure 17*. This will increase the amplitude of the higher frequency into the AGC, thereby preventing the AGC from changing gain.

Another problem encountered in Region 2 operation is that the amplitude of the highest frequency may be so low that it may not trip the hysteresis level. If this happens, these peaks would not be gated on to the output. This problem can also be corrected by placing a separate filter to the gating channel which will make the amplitude of the highest frequency equal the amplitude of the lowest frequency. This is illustrated in the following example.

Consider a disk system which uses the 2,7 code and has an input at the inner track which looks like the fifth waveform in *Figure 15*. Since the flux density on the outer track is 1/1.7 times the flux density of the inner track, the outer track waveform will look like the third waveform. One filter cannot perfectly compensate both these extremes, so we design to compensate a waveform between these two. The track which is $\frac{2}{3}$ of the way in towards the inner track is a good compromise. The filter in this example is a single zero placed such that the lowest frequency followed by the highest frequency have the same amplitude on the track $\frac{2}{3}$ of the way in. *Figure 18* shows the operation of the inner track of this example. While the gating channel filter has made the amplitudes of the two frequencies nearly the same, the time relationship to the Time Channel Input has not been preserved. The proper operation is to have the positive edge of the signal at the Time Pulse In pin, which corresponds to a peak, be the first positive edge after the output of the comparator has changed states. This can be accomplished either of two ways. One way is to insert an external delay between the Time Pulse Out and the Time Pulse In as shown in *Figure 18*. The required delay can be determined by comparing the Time Pulse Out to the Channel Alignment Output with both external filters in the circuit. Another way is to design the Time Channel Filter with more group delay. This will probably require additional poles.

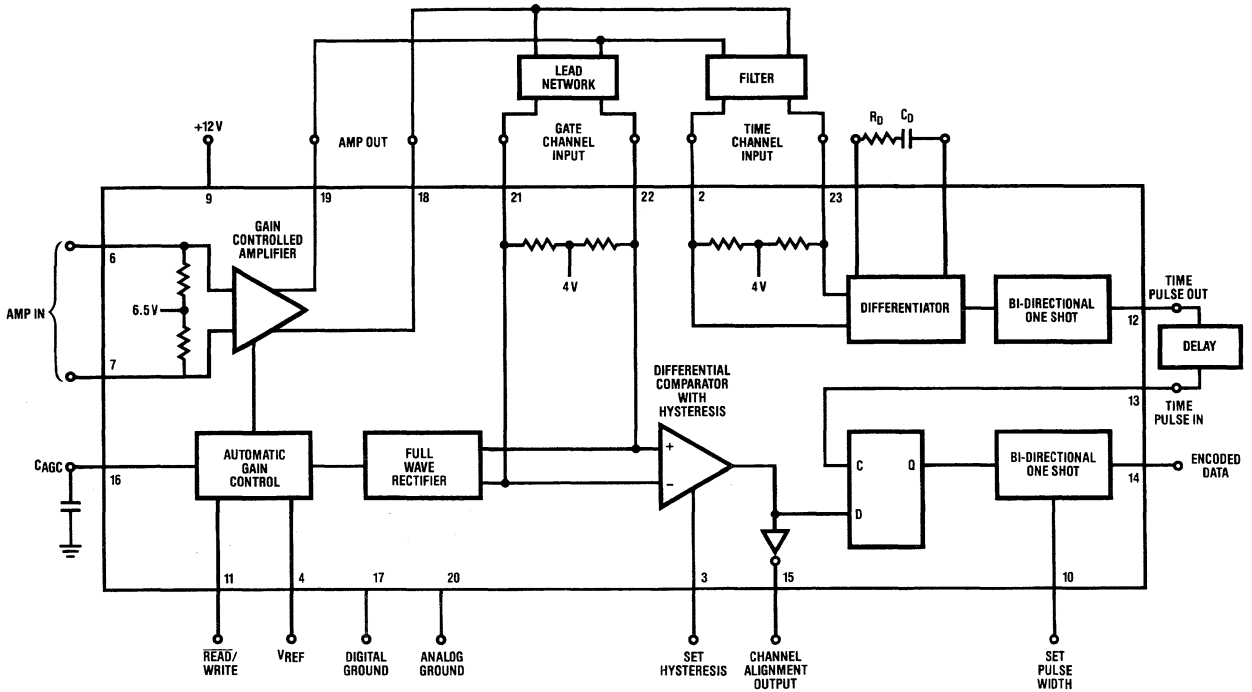
Figure 19 shows the outer track operation of our example. Notice how the system has taken care of the shoulder-induced-noise on the Time Pulse Out. The external delay has shifted the Time Pulse In so the noise is not clocking in new data to the flip-flop. It is important to select this delay such that the positive edge corresponding to a signal peak is always the first positive edge after the output of the comparator has changed states.

While the gating filter has equalized the amplitudes between the highest and the lowest frequency, the amplitude between the inner and the outer track has not been held constant. This can be seen by comparing the Gate Channel Input between *Figure 18* and *Figure 19*. In order to avoid saturating the Gain Controlled Amplifier, the voltage on the V_{REF} pin must be set so that the voltage out of the Gain Controlled Amplifier is 4 Vpp or less for all tracks. The low frequency signal on the inner track contains far more fundamental frequency than the low frequency signal on the outer track. Consequently, the low frequency inner track signal will experience more attenuation than the low frequency outer track signal in passing through the gating channel filter which, for this example, has been optimized to pass higher frequencies. The AGC tends to hold the input to the gating channel constant for a fixed V_{REF} level. Therefore the largest output from the Gain Controlled Amplifier is for the low frequency inner track signal. The voltage on V_{REF} should be adjusted so that the differential output swing of the Gain Controlled Amplifier is 4 Vpp maximum for this signal. This means that the output voltage on the outer track will be less than 4 Vpp.



TL/F/5283–22

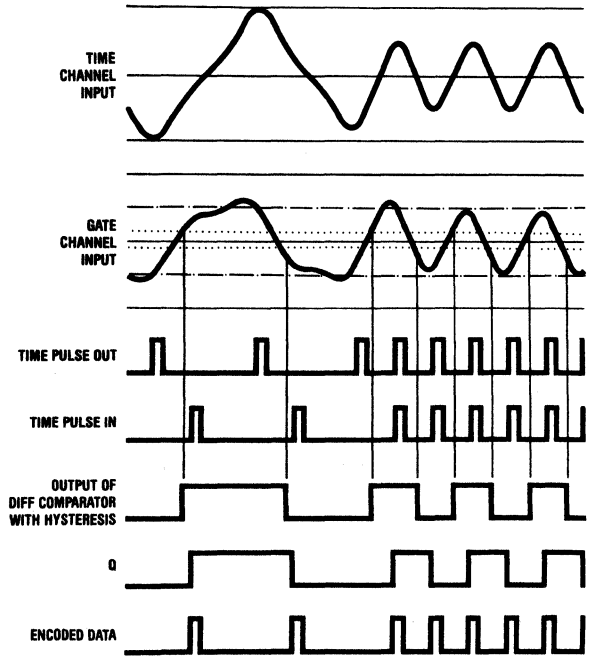
FIGURE 16. Improper AGC Response to Region 2 Signal



TL/F/5283–23

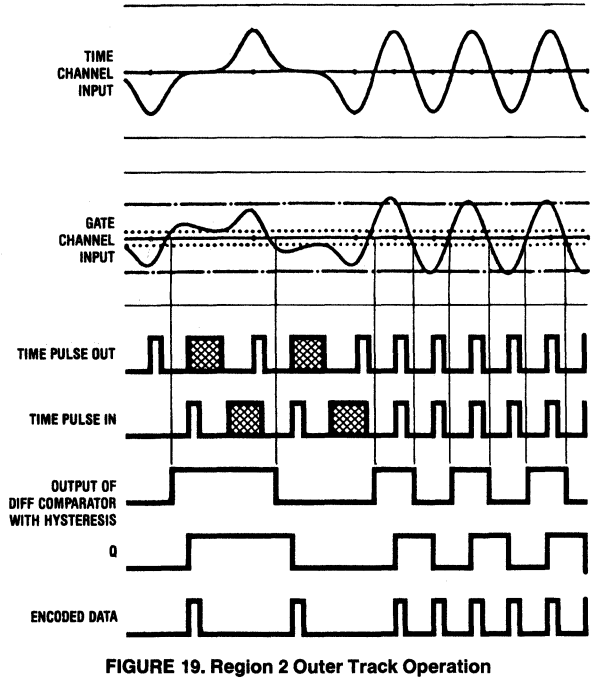
FIGURE 17. Circuit Connection for Region 2 Operation

Application Information (Continued)



TL/F/5283–24

FIGURE 18. Region 2 Inner Track Operation



TL/F/5283–25

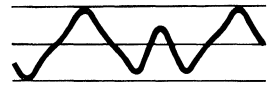
FIGURE 19. Region 2 Outer Track Operation

Application Information (Continued)

Another troublesome input pattern which should be investigated is a high frequency triplet surrounded by the lowest frequency as shown in *Figure 20*. Since the center bit of the triplet does not rise very much above the baseline, there is the possibility it will not trip the hysteresis level. This pattern should be checked to ensure the gating channel filter raises this center bit enough for the proper operation of the gating channel. The operation of the triplet in the previous example is shown in *Figure 21*.

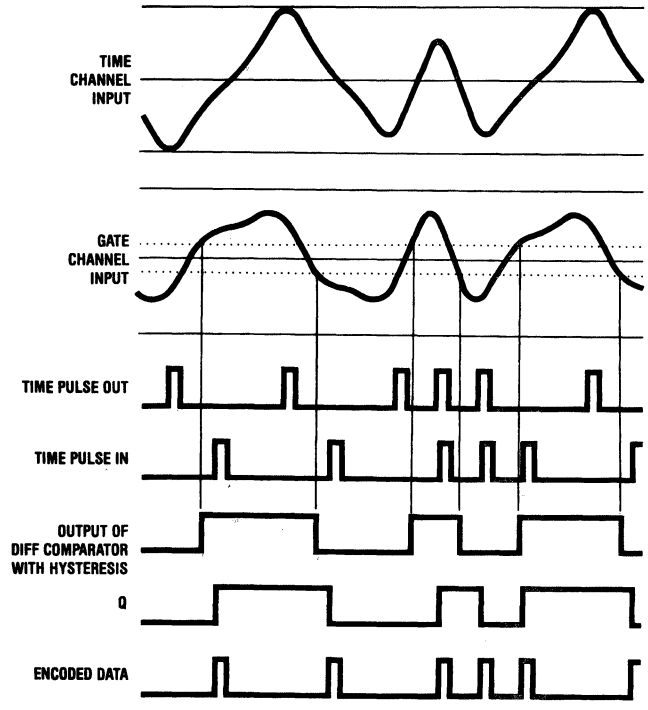
LAYOUT CONSIDERATIONS

Figure 22 is a top view of the component layout for the DP8464B application board whose schematic is shown in *Figure 23*. Care must be exercised in the board layout in order to isolate all digital signals from analog signals. The layout shown in *Figure 22* is a good example of what is required in this regard. In particular the Amp. In pins (pins 6 and 7) and the C_{DIFF} pins (pins 1 and 24) must be isolated from all digital signals. An analog ground plane will greatly aid in this isolation as will separate digital and analog grounds. The V_{CC} (pin 9) should have a 0.1 μf bypass capacitor to analog ground located close to the DP8464B. The component list is provided as an example. These components will need to be optimized for a specific read channel.



TL/F/5283–26

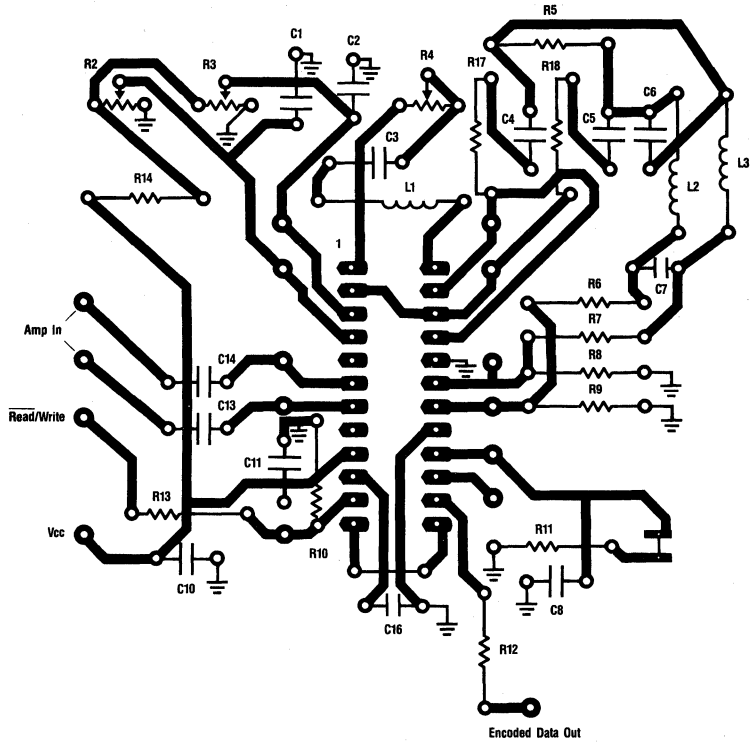
FIGURE 20. (2,7) Triplet



TL/F/5283–27

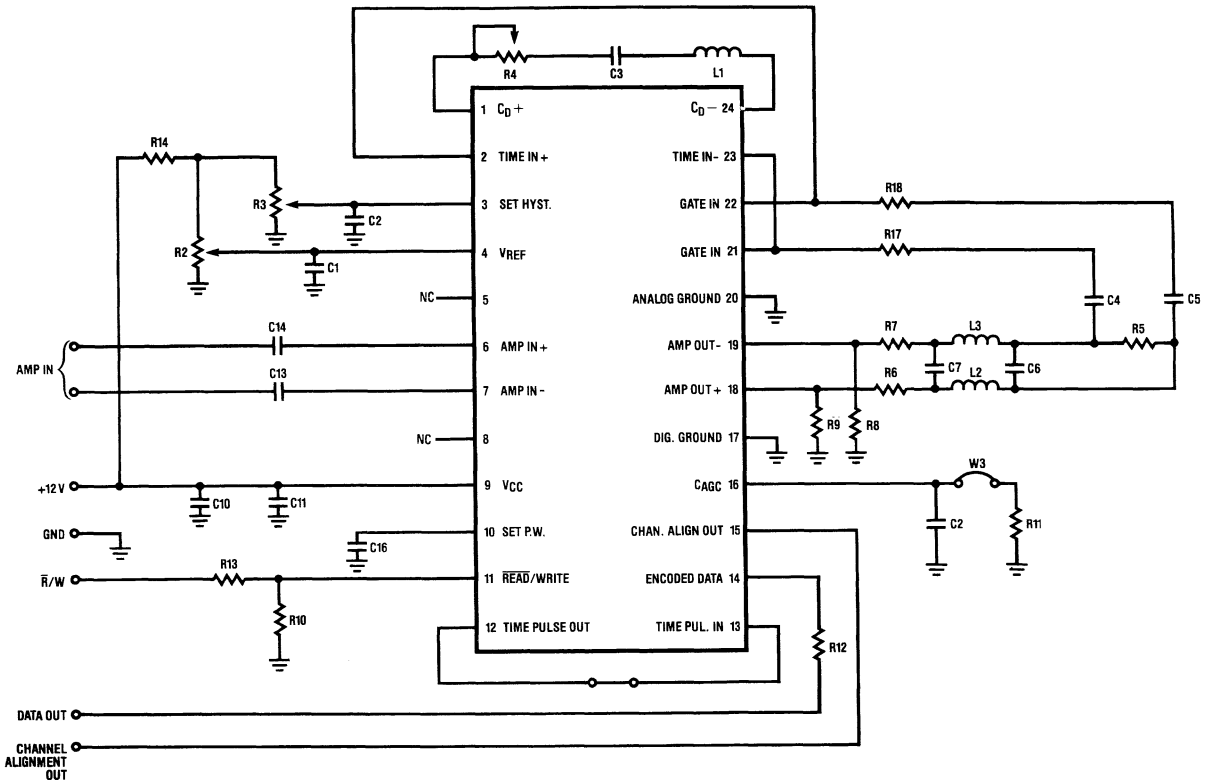
FIGURE 21. Region 2 Triplet Operation

Application Information (Continued)



TL/F/5283–28

FIGURE 22. DP8464B Component Layout—Top View



TL/F/5283–29

FIGURE 23. DP8464B Application Board Schematic

Application Information (Continued)

PARTS LIST FOR DP8464B BOARD

Component Name	Note #	Function	Value	Value for 5 Mbits/sec	Value for 10 Mbits/sec
R2	3	Adjustment for V_{REF} (AGC amplitude)	1k pot		
R3	3	Adjustment for Set Hyst. (threshold)	1k pot		
R4	2	Adjustment for differentiator network Q	5k pot		
R5	1	Low pass filter resistor	560Ω		
R6	1	Low pass filter resistor	240Ω		
R7	1	Low pass filter resistor	240Ω		
R8		Amp Out emitter bias resistor	4.3k		
R9		Amp Out emitter bias resistor	4.3k		
R10		Pull down resistor for Read/Write Pin	5.1k		
R11		Resistor in parallel with C_{AGC}	100k		
R12		Encoded Data Out damping resistor	51Ω		
R13		Read/Write damping resistor	51Ω		
R14		Divider network for Set Hyst. and V_{REF}	2.4k		
R17	6	Series resistor for Time Channel Input	Not required on DP8464B		
R18	6	Series resistor for Time Channel Input	Not required on DP8464B		
C1		V_{REF} cap	0.1 μF		
C2		Set Hyst. cap	0.1 μF		
C3	2	Differentiator cap		100 pF	50 pF
C4		Time and Gate Channel In coupling cap	0.01 μF		
C5		Time and Gate Channel In coupling cap	0.01 μF		
C6	1	Low pass filter cap		200 pF	100 pF
C7	1	Low pass filter cap		30 pF	15 pF
C8	4	C_{AGC} cap	0.01 μF		
C10		V_{CC} cap	1.0 μF		
C11		V_{CC} cap	0.1 μF		
C13	5	Amp In coupling cap	2200 pF		
C14	5	Amp In coupling cap	2200 pF		
C16		Set Pulse Width cap		100 pF	50 pF
L1	2	Differentiator inductor		3.6 μH	1.6 μH
L2	1	Low pass filter inductor		10 μH	4.7 μH
L3	1	Low pass filter inductor		10 μH	4.7 μH

BREADBOARD OPERATION NOTES

1. The low pass filter is a 3 pole Bessel with the corner frequency at 3.75 MHz for the 5Mbits/sec board (7.5 MHz for the 10 Mbits/sec board).
2. The differentiator is a simple RLC filter with the break frequency at 8.5 MHz for the 5 Mbits/sec board (17 MHz for the 10 Mbits/sec board). The resistor can be adjusted to correct for phase distortion in the channel.
3. The V_{REF} should be set at 0.5V. Since the low pass filter has a 6 dB loss, the signal on AMP OUT is 4 Vpp differential while the amplitude into the gate channel is 2 Vpp differential. The Set Hyst. should be nominally set at 0.3V.
4. The AGC attack time (the response to an increased input amplitude) is about 2 μs. To increase this time, increase the value of C8 (the AGC capacitor). The AGC decay time (the response to a decrease in amplitude) is about 10 ms. To increase this time, increase the value of R11. Care must be taken to not allow the response of the AGC loop to become too fast, otherwise loop instability may occur.
5. The input pole is set at 72 kHz (1k input impedance and a 2200 pF input coupling capacitor).
6. Pulse pairing (described in the differentiator section of this data sheet) can be caused by unequal delays through the Bi-directional one shots: To minimize this effect, pin 2 should be connected to pin 22, and pin 23 should be connected to pin 21. If connected this way, the delays tend to cancel.

REFERENCES

1. I. H. Graham, "Data Detection Methods vs. Head Resolution in Digital Recording," IEEE Transactions on Magnetics Vol. MAG-14, No. 4 (July 1978)
2. I. H. Graham, "Digital Magnetic Recording Circuits," to be published.
3. Anatol I., Zverev, Handbook of Filter Synthesis, John Wiley & Sons publisher, 1967.

National Semiconductor Corporation

PRELIMINARY

DP8466 Disk Data Controller

General Description

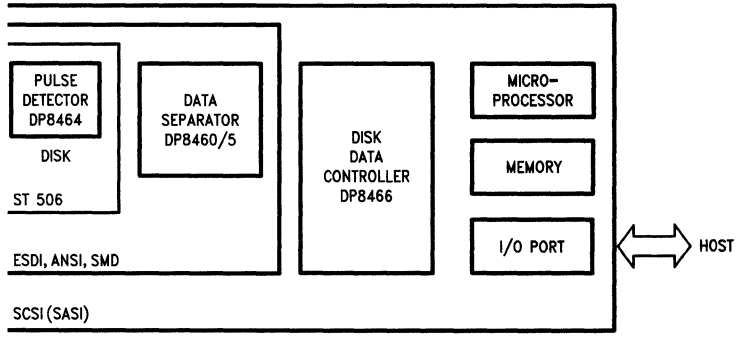
The DP8466 Disk Data Controller (DDC) is an intelligent peripheral which interfaces Winchester or Floppy disk drives to microprocessor based systems. It transfers data between a buffer memory or host system and the serial bit data stream with disk rates up to 25M-bits per second. High speed system data transfer is possible with full on-chip DMA control of buffer or main memory. The 16-bit system I/O interface allows use with any popular 8-bit, 16-bit or 32-bit microprocessor. Programmable track format enables reconfiguration of the DDC for different drive types in a multiple drive environment. Using other National DP8460 series disk data path chips, the DP8466 conforms to ST506, SMD and ESDI standard drive interfaces, as well as to intelligent standard interfaces such as SCSI (SASI) and IPI.

The DP8466 is available in three performance versions DP8466N-12, DP8466N-20 and DP8466N-25.

Part Number	Max Disk Data Rate	Max DMA Transfer Rate
DP8466N-25	25 Mbit/sec	10 Mbyte/sec
DP8466N-20	20 Mbit/sec	8 Mbyte/sec
DP8466N-12	12 Mbit/sec	6 Mbyte/sec

Features

- Easily conforms to any standard drive interface
- Compatible with floppy, hard and optical disk drives
- Compatible with 8, 16 or 32-bit microprocessor systems
- Programmable disk format
- Sector lengths up to 64k bytes, with up to 255 sectors per track
- Programmable 32 or 48-bit ECC polynomial
- Internal ECC correction in less than a sector time
- Disk data rate to 25M bits per second
- Multiple sector transfer capability
- 32 byte internal FIFO data buffer with interleavable burst capability
- 8 or 16-bit wide data transfers
- Single 32-bit or dual 16-bit DMA channel addresses
- Up to 10M bytes per second DMA transfer rate
- +5V supply, 48 pin DIP, microCMOS process



TL/F/5282–1

FIGURE 1. Typical System Configuration

Table of Contents

1.0 INTRODUCTION
2.0 PIN DESCRIPTION
3.0 INTERNAL REGISTERS OF THE DDC
4.0 DDC OPERATION
5.0 FORMAT, READ AND WRITE
6.0 CRC/ECC
7.0 DATA TRANSFERS
8.0 INTERRUPTS
9.0 ADDITIONAL FEATURES
10.0 SYSTEM CONFIGURATIONS
11.0 ABSOLUTE MAXIMUM RATINGS
12.0 DC ELECTRICAL CHARACTERISTICS
13.0 AC ELECTRICAL CHARACTERISTICS AND TIMING DIAGRAMS
14.0 AC TEST CONDITIONS
15.0 MISCELLANEOUS TIMING INFORMATION
16.0 APPENDIX

1.0 Introduction

National's DP8466 Disk Data Controller (DDC) chip is designed to concentrate only on the data aspects of a disk system, leaving the control signals to either a low cost single chip controller or an I/O port from a microprocessor. For this reason, the DDC will work with any standard drive interface.

The DP8466 is an advanced VLSI chip, fabricated in National's latest 2 μ CMOS technology, that allows for operation with disk data rates from the slowest floppy to the fast Winchester and Optical data rates of 25 megabits per second.

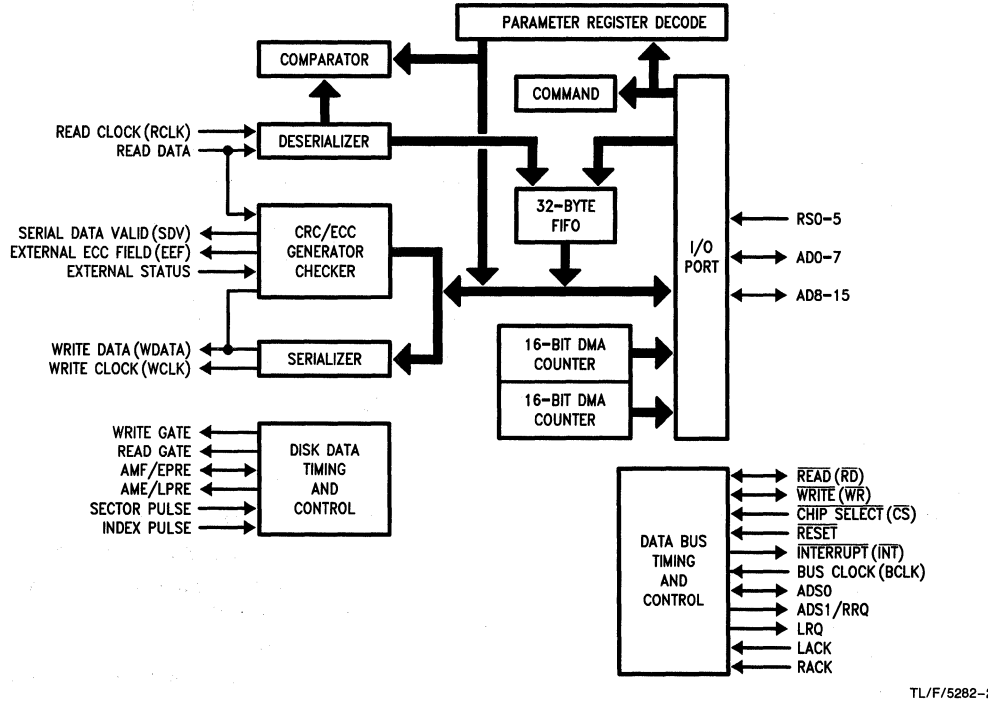
The CMOS design significantly helps the system designer because of reduced power consumption. The chip typically consumes 100 mW.

The DDC is designed for maximum programmability that not only allows the user to select any drive type he wishes, but also allows for different types of drives to be used on the same system. The chip contains 64 registers that can be loaded at any time by a microprocessor connected to the chip's bus. These registers determine the number of bytes in each field of the format, and the byte pattern that each of these fields will repeat. The number of data bytes per sector is selectable from 1 byte to 64k bytes. Finally, both the header field and the data field can each be appended with either a Cyclic Redundancy Check (CRC) field (the 16-bit code used on floppies) or a programmable Error Check and Correct (ECC) field.

The DDC allows the user to load in any 32 or 48-bit ECC polynomial from the microprocessor along with the format parameters. Once an error has been detected, the microprocessor decides whether to re-read the sector during the next revolution of the disk, or to attempt a correction. The DDC can correct errors in a time shorter than that required to read the next sector.

Key blocks in the DDC include a 32-byte FIFO and two 16-bit DMA channels that give the chip a 10 megabyte per second memory transfer capability. This high system data throughout is needed for the high speed drives now becomming available. The small FIFO allows for bursts of data to take place on the bus, thereby leaving the bus free for useful periods of time. The threshold for FIFO data storage is selectable to allow for some degree of system latency. The DDC allows for bursts of 2, 8, 16 or 24 bytes of data to be transferred between the FIFO and memory. The width of the data bus is selectable for either 8 or 16-bit transfers. The system designer selects the threshold so that when the FIFO contains the selected amount of data, the DDC will issue a request. The CPU can continue its operation and then stop to acknowledge the DDC, which then bursts the data between FIFO and memory, before the FIFO has time to overflow or underflow. With a 10 megabit per second disk data rate and a 10 megabyte per second memory transfer cycle, the bus will only be occupied for one-eighth of the time transferring data between FIFO and memory. This leaves the bus free for microprocessor usage for over 80% of the time.

Block Diagram

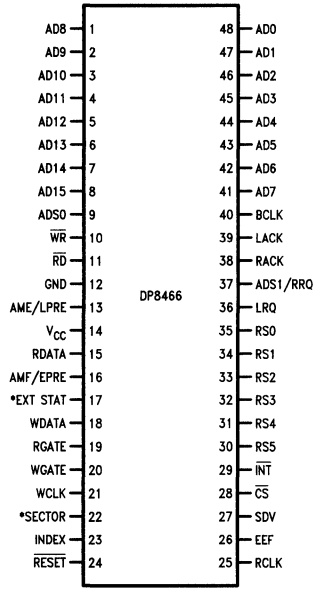


TL/F/5282-2

FIGURE 2. DDC

Connection Diagrams

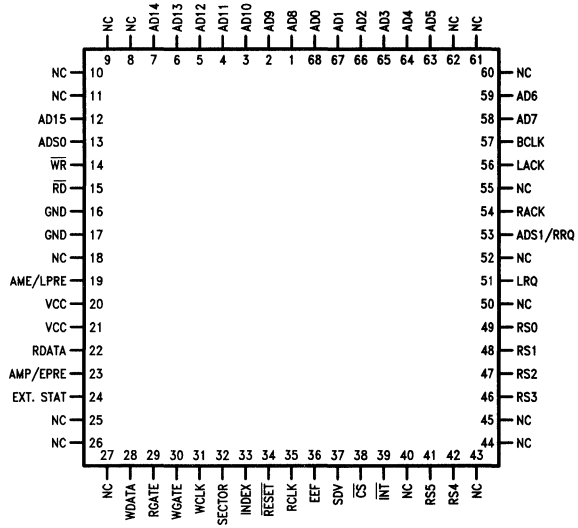
Dual-In-Line Package



TL/F/5282-3

Top View

*This pin must be grounded if not used.



TL/F/5282-76

Top View

Order Number DP8466D or DP8466N
See NS Package Number D48A or N48A

Order Number DP8466V
See NS Package Number V68A

FIGURE 3

2.0 Pin Descriptions

2.1 BUS INTERFACE PINS

Symbol	DIP Pin No.	PCC Pin No.	Type	Function
$\overline{CS}$	28	38	I	**CHIP SELECT:** Sets DDC as a standard I/O port for reading and writing registers. Configures $\overline{RD}$ and $\overline{WR}$ pins as inputs when DMA is inactive. This pin is ignored if on-chip DMA is enabled and performing a transfer.
$\overline{INT}$	29	39	O	**INTERRUPT:** An interrupt can be generated on any error, or after completion of a command, a correction cycle or any header operation.
$\overline{RESET}$	24	34	I	**RESET:** Clears FIFO, Status and Error registers. Halts DMA immediately. Halts disk read and write immediately. Does not affect parameter and most count and command registers. On power-up, must be held low for at least 32 RCLK cycles and 4 BCLK cycles. Note that both RCLK and BCLK must be active for the reset cycle to complete.
$\overline{RD}$	11	15	I/O	**READ:** • MICROPROCESSOR ACCESS MODE, with $\overline{CS}$ pin low and DMA inactive (RACK AND LACK low): Places data from FIFO or register as selected by pins RS0–5 onto the AD0–7 bus. • SLAVE MODE, with LACK pin high: Places data from FIFO onto the AD0–7/AD0–15 bus. • MASTER MODE: When DMA is active, $\overline{RD}$ pin enables data from the addressed device onto the address/data bus.
$\overline{WR}$	10	14	I/O	**WRITE:** • MICROPROCESSOR ACCESS MODE, with $\overline{CS}$ low and DMA inactive (RACK and LACK low): Latches data from AD0–7 bus to internal registers selected by RS0–5. • SLAVE MODE, with LACK pin high: Latches data from AD0–7/AD0–15 bus to FIFO. • MASTER MODE: When DMA is active, $\overline{WR}$ pin enables data from the address/data bus to the addressed device.

2.0 Pin Descriptions (Continued)

2.1 BUS INTERFACE PINS (Continued)

Symbol	DIP Pin No.	PCC Pin No.	Type	Function
BCLK	40	57	I	**BUS CLOCK:** Used as a reference clock when DDC is bus master. Used only during reset and DMA operations. Maximum ratio of RCLK/BCLK is 4 for Word Mode, and 2 for Byte Mode.
RACK	38	54	I	**REMOTE DMA ACKNOWLEDGE:** System input granting use of the bus for a remote DMA bus cycle. If RACK is de-asserted during a transfer, the current transfer cycle will complete.
LACK	39	56	I	**LOCAL DMA ACKNOWLEDGE:** System input granting use of bus for a local DMA bus cycle. If LACK is deasserted during a transfer, the current transfer cycle will complete. LACK has priority over RACK.
RS0–5	35-30	41, 42 46-49	I	**REGISTER SELECT:** Used as address inputs to select internal registers when $\overline{CS}$ pin is low.
AD0–7	48-41	58, 59 63–68	I/O	**ADDRESS/DATA 0–7:** These pins float if $\overline{CS}$ pin = 1 and DMA is inactive. • STANDARD I/O PORT, With DMA inactive and $\overline{CS}$ pin low: Command, Parameter, Count and Status register data is transferred. • SLAVE MODE, with external DMA controller active and LACK pin high: D0–7 are transferred between FIFO and memory. • MASTER MODE, with internal DMA active, and LACK pin high: A16–23, A0–7 and D0–7 are transferred depending on DMA mode and bus phase.
LRQ	36	51	O	**LOCAL DMA REQUEST:** Requests are automatically generated when the FIFO needs to have data transferred.
AD8–15	1–8	1–7 12	I/O	**ADDRESS/DATA 8–15:** • STANDARD I/O PORT, with DMA inactive and $\overline{CS}$ pin low: These pins become indeterminate, but are driven (low impedance). • SLAVE MODE, with external DMA active and LACK pin high: D8–15 are transferred between FIFO and memory. • MASTER MODE, with internal DMA active and LACK pin high: A24–31, A8–15 and D8–15 are transferred, depending on DMA mode and bus phase.
ADS0	9	13	I/O	**ADDRESS STROBE 0:** • INPUT with DMA inactive: ADS0 latches RS0–5 inputs when low. When high, data present on RS0–5 will flow through to internal register decoder. • OUTPUT: ADS0 latches low order address bits (A0–15) to external memory during DMA transfers.
ADS1/RRQ	37	53	O	**ADDRESS STROBE 1/REMOTE REQUEST:** In 32-bit DMA Mode, ADS1 latches high order address bits (A16–31) to external memory. For remote DMA modes, RRQ pin is active high when SRI or SRO bits in the OC register are set in non-tracking mode, or during a remote transfer in tracking mode. (See RT register description in DMA REGISTERS Section.)

2.2 DISK INTERFACE PINS

Symbol	DIP Pin No.	PCC Pin No.	Type	Function
RCLK	25	35	I	**READ CLOCK:** Disk data rate clock. When RGATE is high, RCLK input will be the recovered/separated clock from the recorded data and is used to strobe data into the DDC. When RGATE is low, this input should become the reference clock which will be delayed and used as WCLK to strobe data to the drive. The transition between the recovered/separated clock and reference clock must be made with no short pulses. If RCLK is inactive for longer than the maximum RCLK cycle time (see timing diagrams), the DDC will need to be reset upon RCLK becoming active. Maximum ratio of RCLK/BCLK is 4 for Word Mode, and 2 for Byte Mode.

2.0 Pin Descriptions (Continued)

2.2 DISK INTERFACE PINS (Continued)

Symbol	DIP Pin No.	PCC Pin No.	Type	Function
RGATE	19	29	O	**READ GATE:** Set active high during any disk read operation. This pin commands data separator to acquire lock. Enables RDATA input pin.
RDATA	15	22	I	**READ DATA:** Accepts NRZ disk data from the data separator/decoder.
WCLK	21	31	O	**WRITE CLOCK:** Used when NRZ data is on WDATA pin. Also active when MFM data is used, but normally not utilized. WCLK frequency follows RCLK pin.
WGATE	20	30	O	**WRITE GATE:** When writing data onto a disk, WGATE is asserted high with the first bit of data and deasserted low after the last bit of data. WGATE is also de-asserted on reset or on detection of an error.
WDATA	18	28	O	**WRITE DATA:** During any write operation, MFM or NRZ encoded data is output to disk, dependent upon MFM bit status in the DF register. This pin is inactive low when WGATE is low.
AMF/EPRE	16	23	I/O	**ADDRESS MARK FOUND/EARLY PRECOMPENSATION:** Address mark input is monitored if the HSS bit in the DF register is low (for soft sectoring). If the MFM bit in the DF register and the EP bit in the OC register are both set, then this pin becomes the EPRE control. If both functions are used, WGATE pin determines the function as follows: • WGATE asserted: EPRE output. • WGATE de-asserted: AMF input.
AME/LPRE	13	19	O	**ADDRESS MARK ENABLE/LATE PRECOMPENSATION:** If the MFM bit in the DF register is low, AME will indicate that an address mark byte(s) is being output on WDATA pin. If the MFM bit in the DF register and the EP bit in the OC register are both set, LPRE control is output (if internal MFM encoding is used).
SECTOR	22	32	I	**SECTOR PULSE:** In hard sectored drives, this signal comes from the start of a sector. In a soft sectored drive this pin must be tied low.
INDEX	23	33	I	**INDEX PULSE:** This signal comes from the disk drive, indicating the start of a track.
SDV	27	37	O	**SERIAL DATA VALID:** Asserted when the DDC is either issuing or receiving header field, internal header CRC/ECC, data field, or internal data CRC/ECC information. Mainly used for external ECC and diagnostics.
EEF	26	36	O	**EXTERNAL ECC FIELD:** Only used if the External ECC Byte Count register(s) are non-zero. Asserted when external ECC check bits are being generated (WGATE high) and checked (RGATE high).
EXT STAT	17	24	I	**EXTERNAL STATUS:** *IMPORTANT NOTE: This pin MUST be tied low if it is not to be used.* This pin has three functions: • 1: If EEW bit in the RT register is set, the read and write strobes are extended for both remote and local transfers as long as this pin is high. This is the External Wait State function. • 2: If the EEW bit in the RT register is low, this pin will accept a pulse granting valid byte alignment on the last bit of the synch byte before header or data bytes. This is an OR function with the internal synch detect. • 3: External ECC Check. Only used if External ECC Byte Count register(s) are non-zero, and EEW bit in the RT register is low. After the last byte of external ECC, this pin will accept a pulse confirming that there has been no error. A CRC/ECC error will be flagged if this pulse is not received.
V_{CC} GND	14, 12	20, 21 16, 17		**POWER, GROUND:** +5V DC is required. It is suggested that a decoupling capacitor be connected between these pins. It is essential to provide a path to ground for the GND pin with the lowest possible impedance. Otherwise any voltage spikes resulting from transient switching currents will be reflected in the logic levels of the output pins.

3.0 Internal Registers of the DDC

The numerous registers within the DDC are presented below, grouped according to their function. A key is given as an aid for the use of each register. The key data is only suggested for common operation, and should not be considered as an absolute requirement. Following this listing is a description of each register, in the order of which they are listed below. The HA column at the left of this listing gives the Hex Address of each register.

KEY

- D May be updated when a different drive type is selected
- C May be updated before each command
- R May be read at any time
- F Used during formatting
- I Used during initialization
- NO Operation is not possible

COMMAND

HA	Register	Bits	Write	Read
10	Drive Command Register (DC)	8	C	NO
11	Operation Command Register (OC)	8	C	NO
35	Disk Format Register (DF)	8	D	NO
00	Status Register (S)	8	NO	R
01	Error Register (E)	8	NO	R
12	Sector Counter (SC)	8	C	R
13	Number of Sector Operations Counter (NSO)	8	C	R
0F	Header Byte Count (HBC)/Interlock	3	F	R
36	Header Diagnostic Readback (HDR)	8	NO	R

DMA

HA	Register	Bits	Write	Read
37	DMA Sector Counter (DSC)	8	NO	R
37	Remote Transfer Register (RT)	8	I	NO
36	Local Transfer Register (LT)	8	I	NO
1A	Remote Data Byte Count (L)	8	C	R
1B	Remote Data Byte Count (H)	8	C	R
1C	DMA Address Byte 0	8	C	R
1D	DMA Address Byte 1	8	C	R
1E	DMA Address Byte 2	8	C	R
1F	DMA Address Byte 3	8	C	R

FORMAT

HA	Register	Bits	Write	Read
21	ID Preamble Byte Count	5	D	R
31	ID Preamble Pattern	8	D	R
22	ID Synch #1 (AM) Byte Count	5	D	R
32	ID Synch #1 (AM) Pattern	8	D	R
23	ID Synch #2 Byte Count	5	D	R
33	ID Synch #2 Pattern	8	D	R
24	Header Byte 0 Control Register (HC0)	5	D	R
14	Header Byte 0 Pattern	8	D	R
25	Header Byte 1 Control Register (HC1)	5	D	R
15	Header Byte 1 Pattern	8	D	R

FORMAT (Continued)

HA	Register	Bits	Write	Read
26	Header Byte 2 Control Register (HC2)	5	D	R
16	Header Byte 2 Pattern	8	D	R
27	Header Byte 3 Control Register (HC3)	5	D	R
17	Header Byte 3 Pattern	8	D	R
28	Header Byte 4 Control Register (HC4)	5	D	R
18	Header Byte 4 Pattern	8	D	R
29	Header Byte 5 Control Register (HC5)	5	D	R
19	Header Byte 5 Pattern	8	D	R
2B	ID External ECC Byte Count	5	D	R
2C	ID Postamble Byte Count	5	D	R
3C	ID Postamble Pattern	8	D	R
2D	Data Preamble Byte Count	5	D	R
3D	Data Postamble Pattern	8	D	R
2E	Data Synch #1 (AM) Byte Count	5	D	R
3E	Data Synch #1 (AM) Pattern	8	D	R
2F	Data Synch #2 Byte Count	5	D	R
3F	Data Synch #2 Pattern	8	D	R
3B	Data Format Pattern	8	F	R
38	Sector Byte Count L	8	D	R
39	Sector Byte Count H	8	D	R
2A	Data External ECC Byte Count	5	D	R
20	Data Postamble Byte Count	5	D	R
30	Data Postamble Pattern	8	D	R
34	Gap Byte Count	8	F	R
3A	Gap Pattern	8	F	R

CRC/ECC

HA	Register	Bits	Write	Read
02	ECC SR Out 0	8	NO	R
03	ECC SR Out 1	8	NO	R
04	ECC SR Out 2	8	NO	R
05	ECC SR Out 3	8	NO	R
06	ECC SR Out 4	8	NO	R
07	ECC SR Out 5	8	NO	R
02	Polynomial Preset Byte 0 (PPB0)	8	D	NO
03	Polynomial Preset Byte 1 (PPB1)	8	D	NO
04	Polynomial Preset Byte 2 (PPB2)	8	D	NO
05	Polynomial Preset Byte 3 (PPB3)	8	D	NO
06	Polynomial Preset Byte 4 (PPB4)	8	D	NO
07	Polynomial Preset Byte 5 (PPB5)	8	D	NO
08	Polynomial Tap Byte 0 (PTB0)	8	D	NO
09	Polynomial Tap Byte 1 (PTB1)	8	D	NO
0A	Polynomial Tap Byte 2 (PTB2)	8	D	NO
0B	Polynomial Tap Byte 3 (PTB3)	8	D	NO
0C	Polynomial Tap Byte 4 (PTB4)	8	D	NO
0D	Polynomial Tap Byte 5 (PTB5)	8	D	NO
0E	ECC/CRC Control (EC)	8	D	NO
08	Data Byte Count L	8	NO	R
09	Data Byte Count H	8	NO	R

3.0 Internal Registers of the DDC (Continued)

DUAL-PURPOSE REGISTERS

Some of the above listed registers have dual functions depending on whether they are being written to or read from. These registers are repeated below to help clarify their operation.

HA	Register	Bits	Write	Read
02	ECC SR Out 0	8	NO	R
02	Polynomial Preset Byte 0 (PPB0)	8	D	NO
03	ECC SR Out 1	8	NO	R
03	Polynomial Preset Byte 1 (PPB1)	8	D	NO
04	ECC SR Out 2	8	NO	R
04	Polynomial Preset Byte 2 (PPB2)	8	D	NO
05	ECC SR Out 3	8	NO	R
05	Polynomial Preset Byte 3 (PPB3)	8	D	NO
06	ECC SR Out 4	8	NO	R
06	Polynomial Preset Byte 4 (PPB4)	8	D	NO
07	ECC SR Out 5	8	NO	R
07	Polynomial Preset Byte 5 (PPB5)	8	D	NO
08	Polynomial Tap Byte 0 (PTB0)	8	D	NO
08	Data Byte Count (0)	8	NO	R
09	Polynomial Tap Byte 1 (PTB1)	8	D	NO
09	Data Byte Count (1)	8	NO	R
36	Header Diagnostic Readback (HDR)	8	NO	R
36	Local Transfer Register (LT)	8	I	NO
37	DMA Sector Counter (DSC)	8	NO	R
37	Remote Transfer Register (RT)	8	I	NO

3.1 COMMAND REGISTERS

DRIVE COMMAND (DC) Hex Address (10) Write Only

The locations within this register, when written to, initiate disk commands and chip functions. For a disk operation, after the DDC has been configured, this register is loaded to initiate command execution.

DO2	DO1	H02	H01	FMT	MSO	SAIS	RED
7	6	5	4	3	2	1	0

RED: Re-enable DDC

A 1 should be written into this location during the power up initialization process (see POWER UP AND INITIALIZATION Section), or after an error has been encountered in order to re-enable the DDC to accept commands. (NOTE: If the RES bit in the OC register has been set, a 0 should be written to that location before this operation is performed.) If no error has been encountered, and a command is being issued, a zero should be written to this bit.

SAIS: Start at Index or Sector

0 Operation begins only upon receipt of an index pulse.

1 Operation begins on either an index pulse or sector pulse for hard sector drives or immediately for soft sector drives.

MSO: Multi-sector Operation

0 Single-sector operation.

1 Multi-sector operation using NSO register.

FMT: Format Mode

0 No Format Operation.

1 When set, along with other DC register bits, will initiate disk formatting upon receipt of an index pulse.

HO1, 2: Header Operation Bits:

H02 H01

0 0 *IGNORE HEADER:* associated data transfer operation will take place with any valid sector encountered.

0 1 *COMPARE HEADER:* Normal mode used to find a specific sector. The Header Pattern registers contain the comparison pattern.

1 0 *WRITE HEADER (Write ID):* Normally used only during Format mode to write ID patterns to disk.

1 1 *READ HEADER (Read ID):* Reads header information from disk for diagnostic purposes.

DO1, 2: Data Operation Bits:

D02 D01

0 0 *NO OPERATION:* Can be used only with an Ignore Header command. No disk operation is performed with this combination, and it can be used along with the RED command to re-enable the DDC (see OPERATING MODES).

0 1 *CHECK DATA:* No DMA action and no data movement between disk and FIFO. CRC/ECC checks are calculated and interrupts, if enabled, are asserted on proper conditions. DFE bit in Error register will be set if a data CRC/ECC error occurs unless in Interlock Mode.

1 0 *WRITE DATA:* Initiates local DMA action to fill the FIFO. Writes data to disk with the proper pre and post appendages in the data field. FIFO is replenished by local DMA.

1 1 *READ DATA:* Data enters FIFO from disk, and local DMA transfer is initiated when the FIFO contains the number of bytes specified by the Burst Length in the LT register.

The following table shows a list of valid commands combining the H01, H02, D01, D02, FMT bits from the DC register and the FTF bit in the DF register. No other DC register combinations are allowed.

3.0 Internal Registers of the DDC (Continued)

Valid DDC Commands

DC Register D02	D01	H02	H01	FMT	DF Reg FTF	Operation
0	0	0	0	0	X	No Operation
0	1	0	1	0	X	Check Data, Compare Header
0	1	1	0	0	X	Check Data, Write Header
0	1	1	1	0	X	Check Data, Read Header
1	0	0	0	0	X	Write Data, Ignore Header
1	0	0	1	0	X	Write Data, Compare Header, (normal write)
1	0	1	0	0	X	Write Data, Write Header
1	0	1	0	1	0	Write Data,Write Header, Format with No FIFO Table
1	0	1	0	1	1	Write Data, Write Header, FIFO Table Format
1	1	0	0	0	X	Read Data, Ignore Header, (recover data)
1	1	0	1	0	X	Read Data, Compare Header, (normal read)
1	1	1	1	0	X	Read Data, Read Header

OPERATION COMMAND (OC)
Hex Address (11) **Write Only**

The fields within this register enable on-chip operations. In non-tracking mode, a remote DMA operation will be initiated by loading the SRO or SRI bits in this register.

IR 7	SCC 6	EP 5	SRO 4	SRI 3	EHI 2	EI 1	RES 0

RES: Reset DDC

0 Clears a previously set RES function. Allows normal operation.

1 DDC immediately enters a stand-by mode. The FIFO is reset, Status and Error registers are cleared and all operations in progress are stopped. DDC is placed in the Reset mode (see OPERATING MODES). RGATE and WGATE pins are de-asserted if active. All DMA counters are cleared. Format Parameter, DMA Address and ECC registers are unaffected.

EI: Enable Interrupts

0 Disabled, INT pin remains inactive high.

1 Enables interrupts generated by the following:

- Correction cycle complete.
- Error which sets ED bit in Status register.
- Command successfully completed (including independent remote DMA transfer).

EHI: Enable Header Interrupt

EI bit must be set if this bit is set.

0 Disabled.

1 Interrupt issued at start of ID postamble field when:

- Header matches in Compare Header operation.
- Header finished in Read, Write or Ignore Header operation.

SRI, SRO: Start Remote Input, Start Remote Output

These bits are only operational in non-tracking mode. The Remote Start Address and Remote Data Byte Count registers must be loaded first.

SRI SRO

0 0 Remote DMA operation unchanged.

0 1 *START REMOTE OUTPUT*: Asserts RRQ pin and RCB flag in Status register, to begin a remote DMA operation from memory to I/O Port.

1 0 *START REMOTE INPUT*: Asserts RRQ pin and RCB flag in Status register, to begin a remote DMA operation from I/O Port to local memory.

1 1 *STOP CURRENT REMOTE OPERATION*: RRQ pin is de-asserted and RCB flag is reset in Status register.

EP: Enable Precompensation

0 Early and late precompensation signals are forced low during a disk write operation.

1 Permits precompensation signals to be output to external precompensation circuitry (see MFM ENCODED DATA). This bit is only valid if the MFM bit is set in the DF register.

SCC: Start Correction Cycle

0 No correction is attempted.

1 Setting this command will begin the internal correction cycle. The CCA flag in the Status register is set and drive commands should not be issued during this time. At the completion of the cycle, an interrupt is issued.

IR: Interlock Required (Interlock Mode)

0 No interlock function.

1 The interlock (HBC) register must be written to after the header operation has completed and before the DDC encounters the data postamble field. This allows updating of header bytes during a Format operation or changing of drive commands during a multi-sector operation. Normally used with the header interrupt enabled.

3.0 Internal Registers of the DDC (Continued)

DISK FORMAT (DF) Hex Address (35) Write Only

ID2	1D1	IH2	1H1	FTF	HSS	SAM	MFM
7	6	5	4	3	2	1	0

MFM: MFM Encode

(See MFM Encoded Data section.)

0 NRZ data is output on the WDATA pin when WGATE is active.

1 MFM data is output on the WDATA pin when WGATE is active. Also configures AMF/EPRE and AME/LPRE pins as EPRE and LPRE outputs when Write Gate is active. Precompensated outputs are enabled by the EP bit in the OC register.

SAM: Start with Address Mark

(See Formatting section)

0 Address Marks will be generated in the synch #1 fields if MFM bit = 1, or AME will be generated if MFM bit = 0.

1 Address Mark Enable will be generated in ID preamble if MFM bit = 0.

HSS: Hard or Soft Sectored

(See Hard Sector vs. Soft Sector Operation).

0 Sets DDC for soft sectored operation.

1 Sets DDC for hard sectored operation.

FTF: FIFO Table Format

0 Formatting is done without the use of DMA.

1 The local DMA channel loads the correct number of header bytes (HBC register) per sector into the FIFO from local memory. This data is then substituted for the header bytes during a format operation.

IH1, 2: Internal Header Appendage

IH2 IH1

0 0 No CRC/ECC is internally appended, but external ECC must be attached.

0 1 16-bit CRC CCITT polynomial is appended.

1 0 32-bit programmable ECC code is appended.

1 1 48-bit programmable ECC code is appended.

External ECC may be used with any internal CRC/ECC selection. 1 to 31 bytes of external ECC may be added.

ID1, 2: Internal Data Appendage

ID2, ID1

0 0 No CRC/ECC internally appended.

0 1 16-bit CRC CCITT polynomial is appended.

1 0 32-bit programmable ECC code is appended.

1 1 48-bit programmable ECC code is appended.

External ECC can be appended to any of the four cases dependent upon the Data External ECC Byte Count register.

STATUS (S) Hex Address (00) Read Only

The RESET pin and the RES bit in the OC register reset all of the bits in this register.

ED	CCA	LCB	RCB	LRQ	HMC	NDC	HF
7	6	5	4	3	2	1	0

HF: Header Fault

This bit is valid after a Compare Header or Read Header operation.

SET CRC/ECC error detected in a header field.

RESET This bit is reset when the DDC begins the next disk operation after a new disk command has been issued.

All ID fields entering the DDC during the operation are checked. The HF bit will be set if an error is detected in any header field encountered. However, if the header being sought is found and has no CRC/ECC error, the HF bit is reset. This bit does not produce an error that will stop operation, assert an interrupt, or set the ED bit in the Status register in a compare header operation, but will in a read header operation.

This bit could provide useful diagnostic information if a Sector Not Found error occurs (see Error Register in this section).

NDC: Next Disk Command

SET DDC will accept a new command into the DC register. The header operation is completing the last sector being operated on.

RESET On receipt of a new disk command.

HMC: Header Match Completed

For each of the following, this bit is set and the interrupt is generated at the start of the header postamble field.

Compare Header Operation:

SET Header field correctly matched with no CRC/ECC error.

RESET At beginning of subsequent header operation.

Read Header Operation:

SET Header field has been read with no CRC/ECC error.

RESET At beginning of subsequent header operation.

Ignore Header or Write Header Operation:

SET Always set at end of header field.

RESET At beginning of subsequent header operation.

LRQ: Local Request

This bit follows the LRQ pin, and allows application of the DDC in a polled mode.

SET LRQ pin is asserted.

RESET LRQ pin is not asserted.

RCB: Remote Command Busy

Non-Tracking Mode:

SET When OC register is loaded with a DMA instruction.

RESET Upon completion of the instruction or upon internal or external reset.

Tracking Mode:

SET When RRQ pin is first asserted in a disk write mode, or when the Drive Command register is loaded in a disk read mode.

RESET Upon completion of the instruction or upon internal or external reset.

LCB: Local Command Busy

SET When command requiring local DMA is loaded.

3.0 Internal Registers of the DDC (Continued)

RESET Upon completion of the last local or remote DMA transfer (in tracking mode) or upon internal or external reset.

CCA: Correction Cycle Active

SET On asserting SCC bit in the OC register.

RESET At the end of the correction cycle, simultaneously with the INT pin, if enabled.

ED: Error Detected

SET On assertion of one or more bits in the Error register.

RESET Upon internal or external reset.

ERROR(E) Hex Address (01) Read Only

Any bit set in this register generates an interrupt (if EI bit in the OC register is set) and stops the current operation. The RESET pin and the RES bit in the OC register reset all of the bits in this register.

LI	CF	FDL	NDS	SO	SNF	DFE	HFASM
7	6	5	4	3	2	1	0

HFASM: Header Failed Although Sector Number Matched

(See HFASM description in ADDITIONAL FEATURES)

SET The header bytes(s) marked with the EHF bit in the corresponding HC register(s) matched correctly, but other header bytes were in error.

RESET Upon internal or external reset.

DFE: Data Field Error

SET On detection of a data field CRC/ECC error in a Read Data or Check Data operation. This bit may be set when another error occurs; especially an error occurring during a Write operation. These errors would be Sector Overrun or FIFO Data Lost.

RESET Upon internal or external reset.

The RED command must be loaded into the DC register if error correction is to be attempted.

SNF: Sector Not Found

SET When header cannot be matched for two consecutive index pulses in any Compare Header operation.

RESET Upon internal or external reset.

SO: Sector Overrun

SET If RGATE is active and FIFO is being written to when a sector or index pulse is received. If WGATE is active, this bit is set when a sector or index pulse is received.

RESET Upon internal or external reset.

An SO error will not occur during a Format operation.

NDS: No Data Synch

SET If a sector or index pulse occurs while the DDC is waiting to byte align on the first data synch field (synch #1 or synch #2), or if the DDC byte aligns to the first synch word of the data field but does not match to subsequent bytes (synch #1 or synch #2).

RESET Upon internal or external reset.

FDL: FIFO Data Lost

SET During a disk read operation if the FIFO overflows, or during a disk write operation if the FIFO is read when it is empty.

RESET Upon internal or external reset.

CF: Correction Failed

SET If correction is attempted (SCC bit set in OC register) and correction failed.

RESET Upon internal or external reset.

LI: Late Interlock

Will only occur if IR bit in OC register is set.

SET Controlling logic has failed to write to the Interlock (HBC) register before the end of the data field of the present sector.

RESET Upon internal or external reset.

SECTOR COUNTER (SC)

Allowable Value 0–255 **Hex Address (12) Read/Write**

In a multi-sector operation, the SC register is first loaded with the starting sector number. It is incremented after each header operation is completed. The contents of the SC register will replce any header Byte if the SSC bit is set in the corresponding HC register.

NUMBER OF SECTOR OPERATIONS COUNTER (NSO)

Allowable Value 0–255 **Hex Address (13) Read/Write**

In a multisector operation, the NSO register is loaded with the number of sectors to be operated on. It is decremented after every header operation. When zero, the command is finished. This counter must be reloaded after a reset of the DDC.

HEADER BYTE COUNT (HBC)/INTERLOCK

Allowable Value 2–6 **Hex Address (0F) Read/Write**

This register loads the DMA with the number of header bytes to expect in a Read Header, or a Format operation where FIFO table formatting is used. This register is also used in interlock mode to signal completion of update. The upper five bits of this register are pulled low when read.

HEADER DIAGNOSTIC READBACK (HDR)

Hex Address (36) Read Only

If a Compare Header/Check Data operation is performed and an HFASM error occurs, the header bytes for that sector will have been loaded into the FIFO. By consecutively reading this address, the header bytes are read from the FIFO to the microprocessor. Data will be valid for only the number of header bytes specified in the parameter RAM. (NOTE: This is a dual function register, sharing operation with the Local Transfer register, see DMA REGISTER.)

SECTOR BYTE COUNT REGISTER (L, H)

Allowable Value 1–64k **Hex Address (38, 39) Read/Write**

The two bytes (most and least significant) that comprise this register are loaded during initialization, and define the data field size for each sector. The number of bytes transferred with local DMA is always equal to what has been loaded into this register. Loading *both* with zero is not allowed.

3.0 Internal Registers of the DDC (Continued)

3.2 DMA REGISTERS

LOCAL TRANSFER (LT) Hex Address (36) Write Only

This is a dual function register, sharing operation with the Header Diagnostic Readback (HDR) register (see COMMAND REGISTERS). If any internal DMA is being used, or if the Remote Data Byte Count registers will be read by the processor, the LT (and RT) register must be loaded before the Sector Byte Count and Remote Data Byte Count register pairs.

LBL2	LBL1	LTEB	LA	LSRW	RBO	LWDT	SLD
7	6	5	4	3	2	1	0

SLD: Select Local DMA Mode

0 *SLAVE MODE*: External DMA must be used in place of on-chip DMA.

1 *NON-TRACKING MODE*: Local DMA is enabled. Whenever local transfers are needed, the DDC becomes the bus master.

TRACKING MODE: Local and remote DMA are enabled. DMA transfers are interleaved (see DMA in DATA TRANSFER section).

LWDT: Local Word Data Transfer

0 Address increments by 1, 8 bit wide transfers.

1 Address increments by 2, 16 bit wide transfers. Address, A0, remains unchanged as it was set by the DMA address.

RBO: Reverse Byte Order

Valid if LWDT bit is set.

0 First byte to/from FIFO is mapped onto the AD0–7 bus.

1 First byte to/from FIFO is mapped onto AD8–15 bus (e.g. 68000).

LSRW: Local Slow Read And Write

0 DMA cycles are four clock periods.

1 DMA cycles are five clock periods. RD and WR strobes are widened by one clock period.

LA: Long Address

Valid only if SLD = 1, and SRD = 0 in Remote Transfer register.

0 16 address bits are issued and strobed by the ADS0 pin. ADS1/RRQ is available for use by the remote DMA.

1 32 address bits are issued, the lower 16 are strobed by ADS0 pin. The most significant 16 address lines are only issued when a rollover from the least significant 16 address lines occurs, or after loading the upper half of the 32-bit address. When the upper 16 address lines are issued, that DMA cycle is five clock cycles long if no internal or external wait states are used.

LTEB: Local Transfer Exact Burst

0 When DMA tranfer is needed, the FIFO will be filled when writing to disk or emptied when reading from disk.

1 When DMA tranfer is needed, the FIFO will receive (when writing) or deliver (when reading) an exact burst of data.

LBL1, 2: Local Burst Length

LBL2 LBL1

0 0 1 word (2 byte)

0 1 4 word (8 byte)

1 0 8 word (16 byte)

1 1 12 word (24 byte)

When reading from disk, these bits select the number of bytes needed in the FIFO in order to generate an LRQ signal. When writing, these bits select the number of bytes that need to be removed from a full FIFO in order to generate an LRQ. In either case, if the LTEB bit is set, this bit pair indicate how many data transfers will be allowed before LRQ is removed.

REMOTE TRANSFER (RT) Hex Address (37) Write Only

This is a dual function register, sharing operation with the DMA Sector Counter (DSC) (see DSC at the end of this section). If any internal DMA is being used, or if Remote Data Byte Count registers will be read by the processor, the RT (and LT) register must be loaded before the Sector Byte Count and Remote Data Byte Count register pairs.

RBL2	RBL1	RTEB	TM	RSRW	EEW	RWDT	SRD
7	6	5	4	3	2	1	0

SRD: Select Remote DMA

0 Remote DMA inhibited, ADS1/RRQ pin is configured as ADS1.

1 Remote DMA enabled. This is necessary but not sufficient to start remote transfer.

RWDT: Remote Word Data Transfer

0 Remote address increments by 1.

1 Remote address increments by 2. Address A0 remains unchanged as it was set by the starting DMA address.

EEW: Enable External Wait

0 No external wait states acknowledged. Functions 2 and 3 of EXT STAT pin are enabled (see PIN DESCRIPTIONS).

1 The EXT STAT pin will lengthen RD and WR strobes during DMA transfers as long as it is maintained at a high level.

RSRW: Remote Slow READ/WRITE

0 Remote DMA cycles are four clock periods long.

1 Remote DMA cycles are five clock periods long, if external wait states are not asserted.

TM: Tracking Mode

See Tracking Mode description in DATA TRANSFER Section.

0 DMA channels are independent and addresses are allowed to overlap.

1 DMA channel addresses are not allowed to overlap.

RTEB: Remote Transfer Exact Burst

0 If a remote transfer has been initiated, the RRQ pin will remain asserted until the number of bytes specified by the Remote Data Byte Count registers has been transferred, or until the oper-

3.0 Internal Registers of the DDC (Continued)

ation is reset or SRI and SRO bits in the OC register are both set when in non-tracking mode, or when DMA sector counter reaches zero when in tracking mode.

1 If a remote transfer has been initiated, the RRQ pin will remain asserted until the exact number of bytes specified by RBL1 and RBL2 has been transferred, or if any of the conditions described in the previous paragraph occur.

RBL1, 2: Remote Burst Length

LBL2	LBL1	
0	0	1 word (2 byte)
0	1	4 word (8 byte)
1	0	8 word (16 byte)
1	1	12 word (24 byte)

REMOTE DATA BYTE COUNT (L, H)

Allowable Value 0–64k Hex Address (1A, 1B) READ/WRITE

This pair of registers specifies the number of bytes in one remote transfer using the 16-bit address of the remote DMA channel. In the non-tracking mode, the remote DMA can transfer 1–64k bytes independent of the local DMA. Loading both registers with zero will be interpreted as a 64k byte count. These registers are ignored in tracking mode.

DMA ADDRESS BYTE 0–3

Allowable Value 0–255 Hex Address (1C–1F) READ/WRITE

These address bytes are configured dependent on the current DMA mode. In *32-bit mode*, all four bytes form the physical address with 1F containing the most significant byte. In *16-bit mode*, bytes 0 and 1 form the low and high bytes of the local DMA channel, and bytes 2 and 3 form the low and high of the remote DMA channel, if enabled.

DMA SECTOR COUNTER (DSC)

Hex Address (37) Read Only

This counter is only valid during tracking mode and holds the difference between the number of sectors transferred by the local and remote DMA channels. In tracking mode, when DSC = 0, remote transfer is disabled in a disk read operation so invalid data is not exchanged between local and host memory. This is a dual function register, sharing operation with the Remote Transfer (RT) register described earlier in this section.

3.3 FORMAT REGISTERS

The disk format is defined by using the format pattern and control registers. Generally, these registers are set up in pairs. In each pair, one register is loaded with an appropriate 8-bit pattern that will be written to the disk during a Format or Write command, or will be used during a Read or Compare command for byte alignment or a comparison in locating a sector. Refer to *Figure 4*, below, for a listing of the format registers, and the manner in which they are paired. The FORMAT, READ AND WRITE Section contains a listing and description of each of the format fields.

The other register in the pair is used to control the use of the corresponding pattern register. These Byte Count registers are loaded with a 5-bit binary number indicating the number of times the associated pattern will be repeated, therefore defining the size of that particular field (0–31 bytes). The Gap Byte Count register is the only one with 8 bits, allowing a field of up to 255 bytes in length.

The External ECC Count registers do not perform any pattern repetition. The external ECC appendage is provided from outside the DDC, and must be fit into the field whose length is defined by these registers (0–31 bytes). If any field is to be excluded from the disk format, the Byte Count register associated with that field must be loaded with zero. This is particularly important with the External ECC Byte Count registers. If these are non-zero, the EXT STAT pin will expect a pulse for each external ECC field during a Read operation. If these pulses are not supplied, the operation will be aborted in an error condition. Also, no more than two consecutive format fields may be deleted at one time.

The Header Byte Control registers also do not perform any pattern repetition, nor do they define field size. They are provided for controlling the function of each corresponding header byte.

HEADER CONTROL (HC0–5)

Hex Address (24–29) Read/Write

There is one HC register for each of six Header Byte pattern registers.

NU	NCP	EHF	SSC	HBA
4	3	2	1	0

HBA: Header Byte Active

0 The corresponding Header Byte is not included in the header byte field and will not be used in the ID operation. All other bits in each HC register in which this bit is set to zero must also be set to zero. A minimum of two Header Bytes must be enabled out of six, with no more than two disabled consecutively.

1 The corresponding Header Byte contains valid data and will be used in the ID operation.

SSC: Substitute Sector Counter

0 The corresponding Header Byte as stored in the pattern register is directly written to the disk for a Write Header command, and will be compared for Compare Header command.

1 The contents of the Sector Counter (SC) are substituted for this Header Byte during a Write Header command and compared during a Compare Header command. This is normally used in multisector operations.

EHF: Enable HFASM Function

See HFASM function description in ADDITIONAL FEATURES.

0 HFASM function is disabled.

1 HFASM function is enabled. The corresponding Header Byte is designated as that byte that must match in order to generate an HFASM error, typically the sector number.

NCP: Not Compare

0 The corresponding Header Byte will be compared normally.

1 A valid comparison will always be assumed, regardless of the true outcome.

NU: Not Used

This bit must be set to zero. If set to 1 unspecified operations may occur.

3.0 Internal Registers of the DDC (Continued)

Pattern Register	Hex Addr	Pattern Source	Control Function	Hex Addr	Control Register
ID Preamble	31	*Internal*	*Repeat 0–31x*	21	ID Preamble Byte Count
ID Synch #1 (AM)	32			22	ID Synch #1 (AM) Byte Count
ID Synch #2	33			23	ID Synch #2 Byte Count
Header Byte 0	14		*Define/Control*	24	Header Byte 0 Control
Header Byte 1	15			25	Header Byte 1 Control
Header Byte 2	16			26	Header Byte 2 Control
Header Byte 3	17			27	Header Byte 3 Control
Header Byte 4	18			28	Header Byte 4 Control
Header Byte 5	19			29	Header Byte 5 Control
ID External ECC	*	*External*	*0–31 Bytes*	2B	ID External ECC Byte Count
ID Postamble	3C	*Internal*	*Repeat 0–31x*	2C	ID Postamble Byte Count
Data Preamble	3D			2D	Data Preamble Byte Count
Data Synch #1 (AM)	3E			2E	Data Synch #1 (AM) Byte Count
Data Synch #2	3F			2F	Data Synch #2 Byte Count
Data Format	3B		*Field Size 1–64k Bytes*	38	Sector Byte Count L
				39	Sector Byte Count H
Data External ECC	*	*External*	*0–31 Bytes*	2A	Data External ECC Byte Count
Data Postamble	30	*Internal*	*Repeat 0–31x*	20	Data Postamble Byte Count
Gap	3A		*Repeat 0–255x*	34	Gap Byte Count

*These are not pattern registers.

FIGURE 4. Format Registers

3.4 CRC/ECC REGISTERS

The following registers are for programming and controlling the CRC/ECC functions of the DDC. Many of these registers have dual functions, depending on whether they are being written to or read from. Take care in noting which these are, to avoid confusion later. Only a basic functional description of these are provided here. Detailed instructions on their use can be found in the CRC/ECC section.

ECC SR OUT 0–5 Hex Address (02–07) Read Only

The syndrome bytes for performing a correction are available from these registers, and are externally XOR'ed with the errored data bytes. These are dual function registers, sharing operation with the Polynomial Preset Bytes.

POLYNOMIAL PRESET BYTES 0–5 (PPB0–5)

Hex Address (02–07) Write Only

The ECC shift registers can be preset by loading a bit pattern into these registers. These are dual function registers, sharing operation with the ECC SR Out registers.

POLYNOMIAL TAP BYTES (PTB0–5)

Hex Address (08–0D) Write Only

These registers are used for programming the taps for the internal 32 or 48-bit ECC polynomial. PTB0 and PTB1 are dual function registers, sharing operation with the Data Byte Counters.

DATA BYTE COUNTER 0, 1 (LS, MS)

Hex Address (08, 09) Read Only

The Data Byte Counters indicate the location of the byte in error after an ECC cycle. These are dual function registers, sharing operation with the Polynomial Tap Bytes 0 & 1. The Sector Byte Count Register must be reloaded with the sector length plus the number of ECC bytes before the start of a correction cycle. If the CF bit in the Error register is reset after a correction, the Data Byte Counter will contain an offset pointing to the first byte in error.

ECC/CRC Control (EC)

Hex Address (OE) Write Only

DNE	IDI	IEO	HNE	CS3	CS2	CS1	CS0
7	6	5	4	3	2	1	0

CS0–CS3: Correction Span Selection Bits

These four bits program the number of bits that the ECC circuit will attempt to correct. Errors longer than the correction span will be treated as non-correctable. The allowable correction span is 3–15 bits. If a span outside this range is loaded, the DDC will automatically default to a span of three bits.

For example, a five bit correction span would load as:

CS3	CS2	CS1	CS0
0	1	0	1

HNE: Header Non-Encapsulation

0 Header address mark and/or synch fields are encapsulated in the CRC/ECC calculation.

1 Header address mark and/or synch fields are not encapsulated in the CRC/ECC calculation.

NOTE: The SAM bit in the DF register must be reset when performing a Compare or Read Header operation, and the HNE bit is active low. If this is not done, the CRC/ECC calculation will begin at the synch word of the header, resulting in a Header Fault that will abort a Read operation or a Sector Not Found error for a Compare Header operation.

IEO: Invert ECC Out

See note under IDI bit, below.

0 Checkbits exiting ECC/CRC shift register are unaltered.

1 Checkbits exiting ECC/CRC shift register are inverted.

3.0 Internal Registers of the DDC (Continued)

IDI: Invert Data In

0 Data and checkbits entering the ECC/CRC shift register are unaltered.

1 Data and checkbits entering the ECC/CRC shift register are inverted.

NOTE: This inversion option has been included for compatability with a few systems that require ECC input and/or output inversion.

DNE: Data Non-Encapsulation

0 Data address mark and/or synch fields are encapsulated in the CRC/ECC calculation.

1 Data address mark and/or synch fields are not encapsulated in the CRC/ECC calculation.

4.0 DDC Operation

4.1 MICROPROCESSOR ACCESS

The DDC requires microprocessor control to initiate operations and commands, and to check chip status. All registers in the DDC appear as unique memory or I/O locations. Each can be randomly accessed and operated on. When the DMA is not performing a memory transfer, the chip can be accessed as a memory location or standard I/O port. Only eight bits of data may be transferred at this time, using pins AD0–7 (the upper 8 bits of a 16 bit microprocessor are not used). Six dedicated address pins (RS0–5) individually select all of the DDC's internal registers. By using these dedicated lines with an address strobe input (ADS0), the chip can be used in both multiplexed and demultiplexed address bus environments. The ADS0 and RS0–5 pins operate as a fall through type latch. By asserting CS active low, the DDC recognizes it has to be a slave and allows RD and WR to effect the internal registers. With multiplexed address and data lines, a positive strobe pulse on ADS0 will latch the address. The ADS0 line may be derived from a microprocessor address strobe such as ALE. In systems with a dedicated address bus (demultiplexed), ADS0 may be pulled high to allow address information to flow through the latch. Finally, by applying CS and a RD or WR strobe, any of the 64 internal locations can be accessed. It is important to note that most registers are read or write only. Some registers, however, change function dependent on whether they are being read from or written to (see Dual Function register list in INTERNAL REGISTERS).

4.2 OPERATING MODES

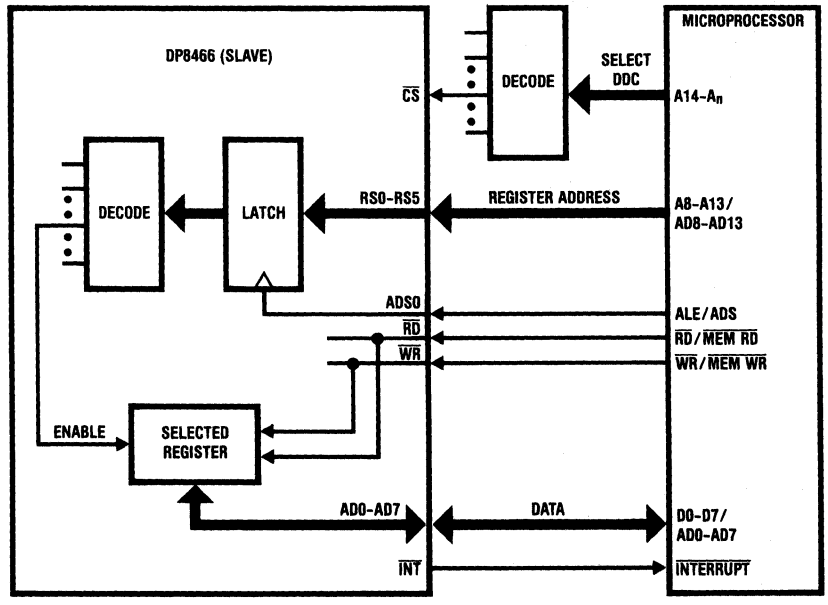
The DDC can be thought of as operating in four modes: *RESET, COMMAND ACCEPT, COMMAND PERFORM* and *ERROR*. These modes are given here in order to provide a functional operating description of the DDC, particularly when an error has been encountered.

Mode 1 *RESET*: All functions are stopped, and no command can be issued. During power up and before initalization, the DDC is held in this mode. To leave this mode, pin 24 (RESET) must be high, a 0 must be written to the RES location in the OC register, and a RED command loaded into the DC register. This places the DDC into MODE 2.

Mode 2 *COMMAND ACCEPT*: The DDC is free and ready to receive the next command (NDC bit set in Status register). Upon receipt of a command, the DDC will enter MODE 3.

Mode 3 *COMMAND PERFORM*: The directed operation is performed. If no error is encountered, the DDC will return to MODE 2. An error will put the DDC into MODE 4.

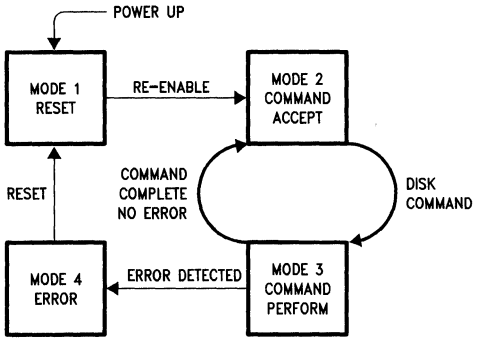
Mode 4 *ERROR*: The error needs to be serviced, and then the DDC can be reset by MODE 1.



TL/F/5282–4

FIGURE 5. Microprocessor Access to DP8466

4.0 DDC Operation (Continued)

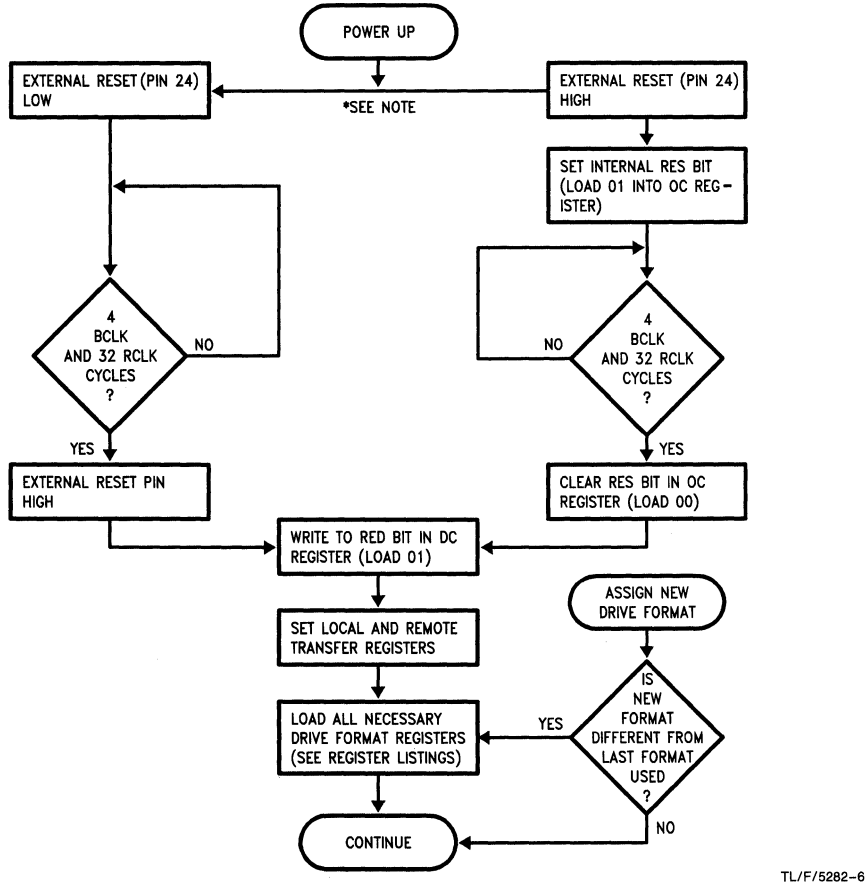


TL/F/5282–5

FIGURE 6. DDC Operating Modes

4.3 POWER UP AND INITIALIZATION

In powering up the DDC, the counters and registers must be initialized before a drive can be assigned and the appropriate information loaded. This can be done by either holding pin 24 (RESET) low, or by setting the internal RES bit in the OC register. Both require that the DDC be held in the reset condition for a minimum of 32 RCLK periods and 4 BCLK periods before the reset condition can be cleared. *Figure 7* shows a general algorithm for both methods. After power up, and whenever a new drive is assigned, the appropriate drive format registers need to be loaded before any drive operation is performed.



TL/F/5282–6

Note: As shown various methods are possible for power up, and it is up to the user as to which is more suitable. The DDC should be reset and RCLK and BCLK should be applied after power up, otherwise it may draw an excessive amount of current, and may cause bus contention.

FIGURE 7. Power Up and Initialization Algorithm

5.0 Format, Read and Write

5.1 DISK FORMATTING

The formatting process is carried out through the format parameter and pattern registers (see FORMAT REGISTERS). These registers should be loaded during the initialization process for the particular drive in use. The pattern registers are loaded with the specific 8-bit pattern to be written to the disk. The count registers specify the number of times each 8-bit pattern is to be written. In loading these registers, several things need to be kept in mind:

- If any byte count register is loaded with zero, that field will be excluded, and no pattern for the corresponding pattern register need be loaded.
- At least two header bytes must be used, with no more than two consecutive unused header bytes. This also applies to all the fields in the format, where no more than two consecutive fields may be deleted. The one exception is the internal header ECC and external header ECC field. At least one of these fields must be present.
- If the disk is hard sectored, no gap byte count needs to be loaded. See Hard Vs. Soft Sector Operation in the FORMAT, READ AND WRITE Section.

The sector format options that are provided with the DDC are shown in *Figure 8*. The fields common to the ID and data fields, such as the preamble, Synch, CRC/ECC and postamble fields, perform similar functions, and are briefly discussed below.

PREAMBLE: Allows the PLL in the data separator to achieve phase lock.

SYNCH #1 and 2: Synch #1 contains the missing clock address mark for use with soft sectored disks. Generally, this field is not used in hard sectored disks. The synch #1 field can be used to extend the preamble or the synch fields in hard sectored mode. Synch #1 and #2 fields allow for byte alignment of the DDC.

HEADER BYTES: Used to uniquely identify each sector. Examples are sector number, cylinder number, track number, etc.

DATA: Information to be stored.

CRC/ECC: This field is generated and checked internally.

EXT.ECC: Used with external ECC circuitry. Provides space for externally generated ECC bytes.

POSTAMBLE: Allows read gate turn off time for the PLL to unlock. Provides a pad so that the write splice does not occur at the end of the CRC.

GAP 3: Provides protection against speed variation. In soft sectored mode, its length is determined by the Gap Byte Count register. In hard sectored mode, this gap will continue until the next sector pulse.

Format operations always start with an index pulse, and end with the next index pulse, thus making one track. The DDC has three approaches for formating disks:

Internal Sequential
FIFO Table
Interlock Type

INTERNAL SEQUENTIAL

This mode is used where the sector number is incremented for each physically adjacent sector, that is, for an interleave of one. This mode may be used on a multi-sector operation to format a whole track of sequential sectors. The header bytes other than the sector number, such as cylinder number and head number, are loaded. The Sector Counter (SC) is loaded with the first sector number desired on the track and the HC register with SSC = 1. The Number of Sector Operations (NSO) counter is loaded with the number of sectors per track. Finally, the FMT bit is set in the DC register in addition to bits for a Write Header/Write Data, multi-sector operation. Formatting begins upon loading the DC register. The last sector number written will therefore be [SC] + [NSO] − 1.

FIFO TABLE

This approach is ideal for sector interleaving and offers the minimum of microprocessor intervention during the format operation. The microprocessor sets up the header bytes of each sector, contiguously in memory. The local DMA channel or external DMA is used to transfer the header byte sets into the FIFO. Each set transferred is used once for each header field. The local DMA transfers a new set for each sector. The number of sectors transferred is determined by the NSO register.

The format operation follows the sequence below:

(1) Before the format operation, a full track of header byte sets is loaded into a memory area accessible to the local DMA channel. Each header byte set must contain an even number of bytes. If it contains an odd number of bytes, an extra "dummy" byte must be inserted so that each header byte set will be contained in an even byte boundary.

(2) The DMA address is loaded with the location of the first byte of the first header byte set.

ID FIELD

ID PREAMBLE	ID SYNCH #1 (AM)	ID SYNCH #2	HEADER BYTES	ID CRC/ECC*	ID EXT ECC*	ID POSTAMBLE
0–31 Bytes	*0–31 Bytes*	*0–31 Bytes*	*2–6 Bytes*	*0, 2, 4 or 6 Bytes*	*0–31 Bytes*	*0–31 Bytes*

DATA FIELD

DATA PREAMBLE	DATA SYNCH #1 (AM)	DATA SYNCH #2	DATA FORMAT PATTERN	DATA CRC/ECC	DATA EXT ECC	DATA POSTAMBLE	GAP 3
0–31 Bytes	*0–31 Bytes*	*0–31 Bytes*	*1–64k Bytes*	*0, 2, 4 or 6 Bytes*	*0–31 Bytes*	*0–31 Bytes*	*0–255 Bytes*

*Note the ID CRC/ECC field and the ID EXT ECC field must not be set to zero simultaneously.

FIGURE 8. Sector Format Fields

5.0 Format, Read & Write (Continued)

(3) The Header Byte Count (HBC) is loaded with the number of header bytes in each sector (2–6 bytes).

(4) The Disk Format (DF) register is loaded with the FTF bit set.

(5) The Drive Command (DC) register is loaded for a Write Header/Write Data, multi-sector, format operation.

INTERLOCK TYPE

This approach offers the most versatility, but requires fast microprocessor intervention. It may be used to format a whole track of interleaved sectors. It can also be used for creating files of varying sector length, but this can be very tricky. The DDC can format sectors with data lengths from 1 to 64k bytes with single byte resolution.

Interlock type formatting uses the interlock mode and the header complete interrupt to enable the microprocessor to directly update any format parameter bytes. The Operation Command (OC) register is loaded wth IR (Interlock Mode), EHI and EI bits set. The Disk Format (DF) register should be loaded with the FTF bit reset. The header byte pattern for each selected header byte must be loaded into the relevant register. The NSO register is loaded with the number of sectors to be formatted. The DC register is then loaded for a Write Header/Write Data, multi-sector, format operation.

After the header field is written in the first sector, the DDC issues the header complete interrupt. With interlock mode set, the controlling microprocessor has the block of time until the preamble field of the next sector to read status, load the next sector's header bytes into the DDC registers and confirm this had been accomplished by writing to the Interlock (HBC) register. This must be done after the HMC interrupt for every sector, including the last sector of the operation. If this is not done, a Late Interlock error will occur when a subsequent command is loaded in the DC register.

In a non-format operation, the user has only until the end of the data field to write to the HBC register (see Data Recovery Using The Interlock Feature in ADDITIONAL FEATURES). This operation is repeated until the NSO register decrements to zero. An interrupt will then be issued indicating that the operation has completed.

5.2 READ AND WRITE

For initiating Read/Write operations, the necessary format registers need to be loaded with the appropriate information to enable the DDC to identify the desired sector. Multi-sector operations will also require the Number of Sector Operations (NSO) counter and the Sector Counter (SC). Algorithms outlining the read/write operations are shown in *Figures 10* and *11*. For each of these, it is assumed that the parameters for the desired sector(s) have been loaded, and that the head is positioned over the proper track.

READ

During a read operation, header data passing under the disk head is compared to the header bytes in the DDC parameter RAM. If a match is found after a read command is issued, the data field of the identified sector will start filling the FIFO. Once the selected threshold data level (burst length) is reached, the Local DMA Request (LRQ) pin will be asserted, signaling that a transfer is required. When the LACK pin grants the bus, either the exact burst length or the entire FIFO contents are transferred to memory. The FIFO continues filling, and this process repeats until the entire data field has been transferred to memory.

WRITE

A similar process occurs in reverse for a write operation. The DMA fills the FIFO, and when the correct sector is found, this data begins to be written to disk. When the data in the FIFO falls by an amount equal to the burst length, a transfer request is issued on LRQ. When LACK is granted, the DMA either fills the FIFO or transfers the exact number of bytes specified in the burst length. This process continues until a number of bytes specified by the Sector Byte Count register has been written to the disk.

Multi-sector operations follow the same procedure, but the operation is repeated on the number of sectors specified in the Number of Sector Operations (NSO) counter, with an interrupt being generated on completion of the last sector.

5.3 HARD SECTOR vs. SOFT SECTOR OPERATION

The choice between hard and soft sectored operation is made through the use of the HSS bit in the Drive Format register. This bit, in conjunction with other control bits can set the DDC to perform a number of functions depending on whether a read, write or format operation is to be enacted. HSS = 0 sets the DDC for soft sectored operation, and HSS = 1 sets the DDC for hard sectored operation.

FORMAT

In hard sectored operation, the DDC assumes that sector pulses are present, and will ignore the gap count. Gap bytes will be written until a pulse is detected on the SECTOR pin.

In soft sectored operation, the gap count will be used for every sector except the last. The Gap Byte Count register determines the Gap 3 length. For the last sector, gap bytes will be written until an index pulse is received.

READ

When reading, the need for the AMF input pulse is determined by the HSS bit. For soft sectoring, the AMF input is required for at least one bit time within the Synch #1 fields in both the ID and Data sections of the sector. For hard sectoring, the AMF input is not required.

The HSS bit in the DF register, and the SAIS command in the DC register define when RGATE is asserted for various sector formats. This is outlined below.

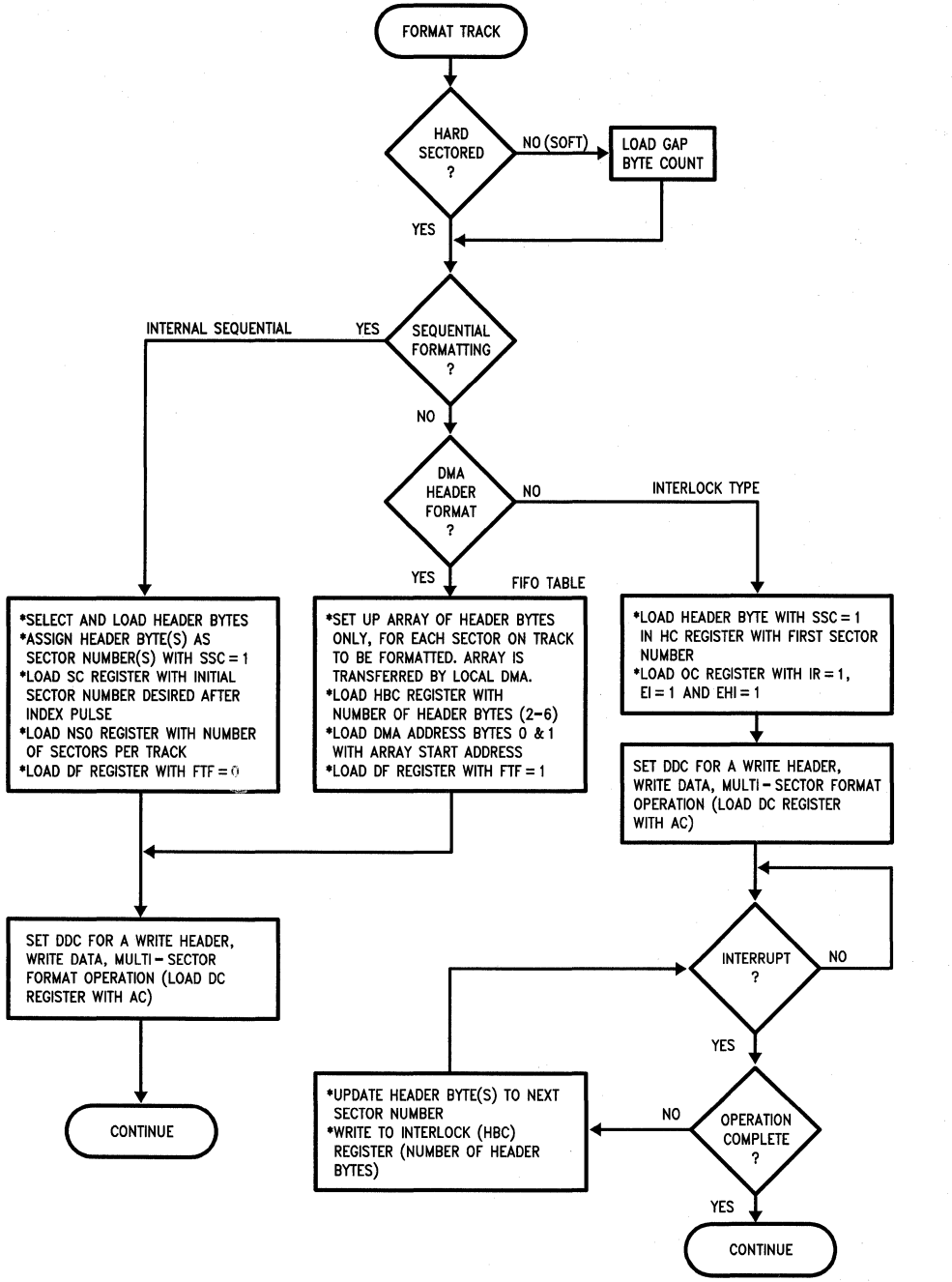
HSS	SAIS	RGATE ASSERTED:
0	0	On index pulse
0	1	On receipt of instruction
1	0	On index pulse
1	1	On index or sector pulse

WRITE

The HSS, MFM and SAM bits in the DF register determine the use of the address mark and the AME pin as follows:

HSS	MFM	SAM	FUNCTION
0	0	0	AME pin activated during ID and data synch #1 fields.
X	0	1	AME pin activated during ID preamble.
0	1	X	Missing clocks inserted in ID and data synch #1 fields. AME pin indicates LPRE (if enabled).
1	0	0	AME pin disabled.
1	1	X	Synch #1 fields written without missing clock pulse.

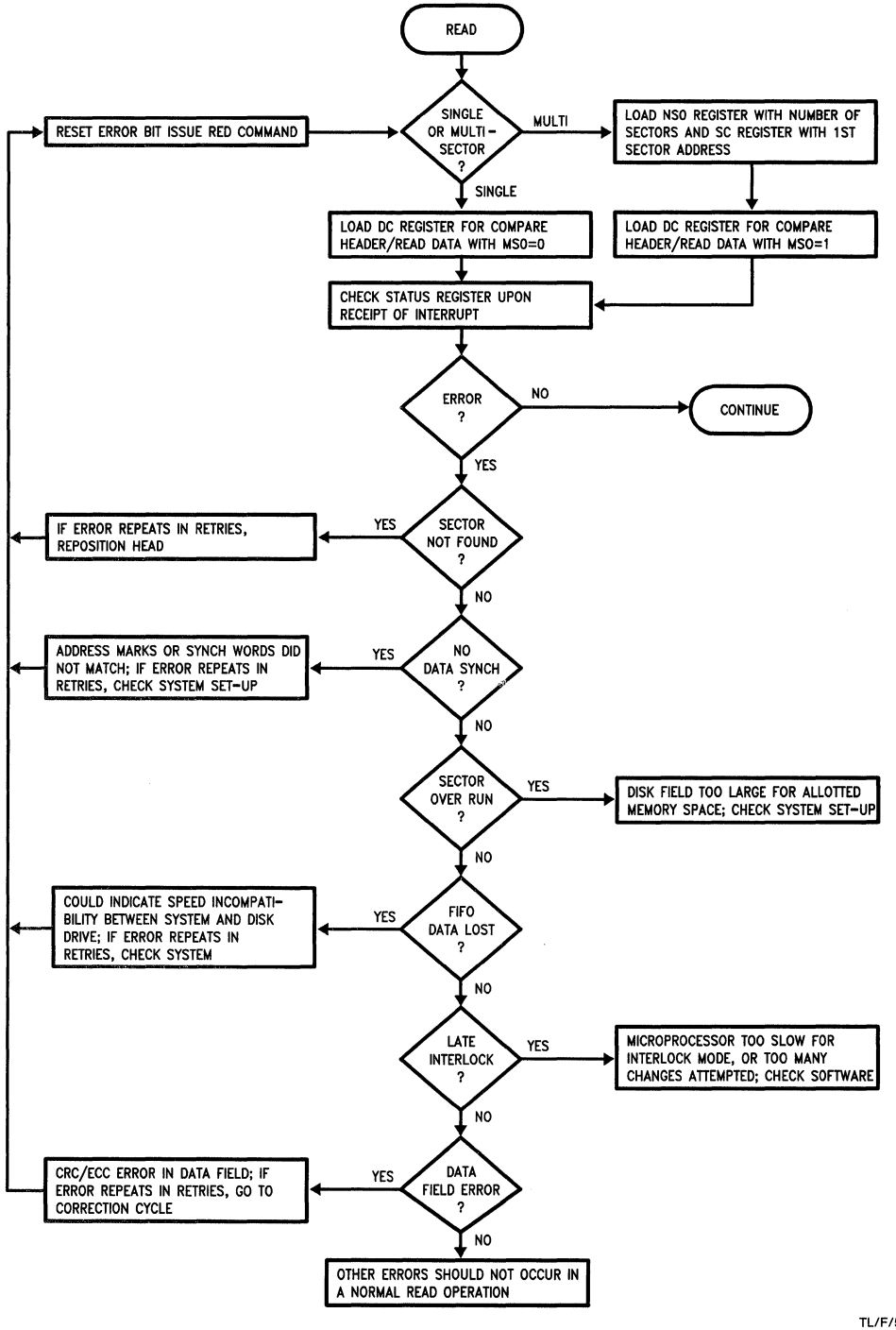
5.0 Format, Read & Write (Continued)



TL/F/5282-7

FIGURE 9. Format Track Algorithm

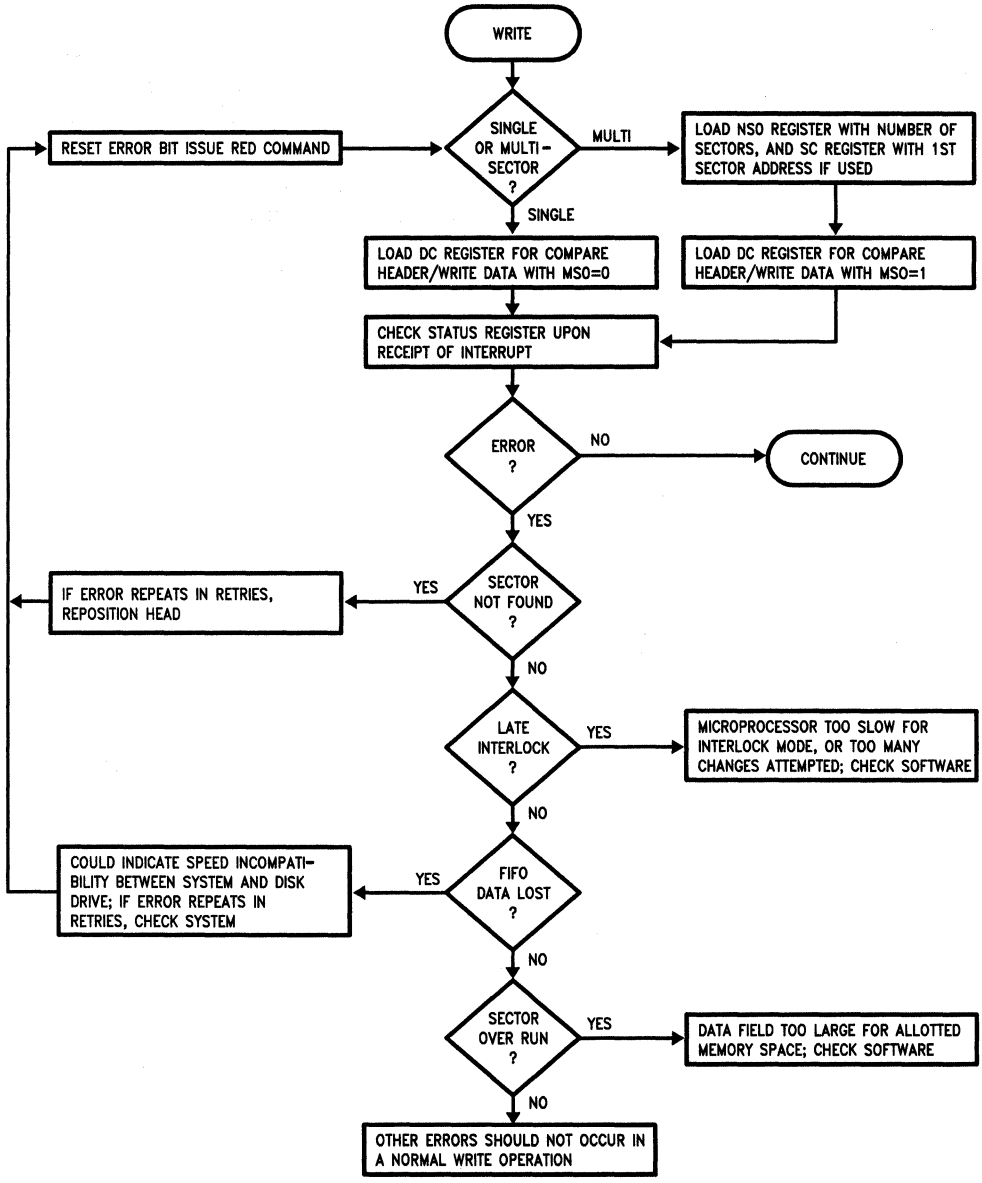
5.0 Format, Read & Write (Continued)



TL/F/5282–8

FIGURE 10. Simple Read Operation

5.0 Format, Read & Write (Continued)



TL/F/5282-9

FIGURE 11. Simple Write Operation

5.0 Format, Read & Write (Continued)

5.4 MFM ENCODED DATA

MFM encoding of write data is controlled by the MFM bit in the DF register MFM = 1 sets the DDC to write MFM data to the disk. MFM = 0 sets the DDC to write NRZ data to the disk.

PRECOMPENSATION OF MFM ENCODED DATA

When the MFM bit in the DF register and the EP bit in the OC register are set, precompensation will be indicated on the EPRE and LPRE pins. Precompensation is issued for the middle bit of a 5-bit field. In the DP8466, early and late precompensation will be enacted for all of the combinations as shown below. All other patterns will not require precompensation. Precompensation can be disabled by setting the EP bit in the OC register inactive low.

EPRE NRZ PATTERNS	LPRE NRZ PATTERNS
00 0 10	00 1 10
00 0 11	00 1 11
01 1 00	10 0 00
01 1 01	10 0 01
11 1 00	10 1 10
11 1 01	10 1 11

Precompensation outputs are aligned to provide symmetrical set-up and hold times relative to the rising edge of the WDATA outputs. This gives a half period of RCLK set-up time on precompensation outputs. This is shown in *Figure 12*. Two bits of zero precede the preamble fields at the leading edge of the write gate when writing MFM data due to MFM encoded delays.

5.5 ADDRESS MARK PATTERNS, MISSING CLOCK

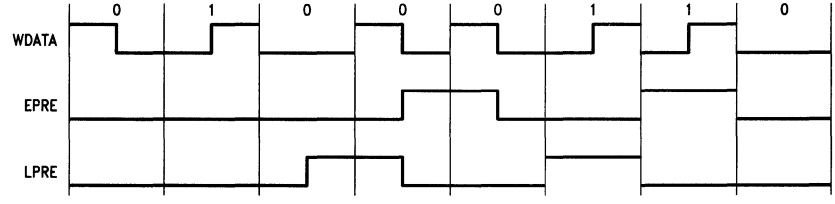
During writing and formatting a sector with MFM encoding enabled, a clock violation, or missing clock pulse, will be inserted in the synch #1 field. This indicates the address mark. For an example of this, refer to *Figure 13*.

When writing MFM encoded data with precompensation enabled, only the following hex values are allowed to be loaded into the synch #1 pattern registers:

A1, C2, C3, E1, 84, 85, 86, 87

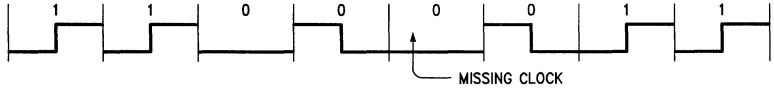
With no precompensation, any pattern containing 100001 is valid.

During a soft sectored read operation, an AMF pulse will be expected on the AMF/EPRE pin during each byte of the synch #1 field.



TL/F/5282–10

FIGURE 12. Example of EPRE and LRPE Outputs



TL/F/5282–11

FIGURE 13. Missing Clock Example

6.0 CRC/ECC

6.1 PROGRAMMING CRC

The DDC is set for internal CRC by programming the disk Format (DF) and ECC/CRC Control (EC) registers. The CRC-CCITT polynomial used by the DDC for the CRC code is given below:

$$P(x) = x^{16} + x^{12} + x^5 + 1$$

The DDC uses the pattern preset to all 1's for the CRC calculation. *Note:* If no CRC/ECC is used for the ID fields, an external ECC must be used.

6.2 PROGRAMMING ECC

There are two sets of six registers used to program the ECC. One set of six is used to program the polynomial taps, while the other set is used to establish a preset pattern (typically all 1's). Bits contained in the ECC Control (EC) register are used to control the correction span. The DF register contains bits for choosing the desired type of appendage: Either 32 or 48-bit programmable ECC polynomials, or the 16-bit CCITT CRC polynomial is possible.

PROGRAMMING POLYNOMIAL TAPS

To program a polynomial into the shift register, each tap position used in the code must be set to 0, and all unused taps should be set to 1. The bit assignment for these registers in 48 and 32-bit modes is shown in the tables that follow. It is important that for 32-bit codes, PTB2 and PTB3 all be set to 1's. Failure to do so will result in improper operation. Also, x^{48} and x^{32} are implied, i.e., a 32-bit ECC will always contain the x^{32} term and a 48-bit ECC will always contain the x^{48} term. For both ECC's, the term x^0 (or 1) is also implied, even though this bit is accessible.

Tap Assignment 48-Bit Mode

REG #	ADDR	BIT NUMBER							
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PTB0	08	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0
PTB1	09	x^{15}	x^{14}	x^{13}	x^{12}	x^{11}	x^{10}	x^9	x^8
PTB2	0A	x^{23}	x^{22}	x^{21}	x^{20}	x^{19}	x^{18}	x^{17}	x^{16}
PTB3	0B	x^{31}	x^{30}	x^{29}	x^{28}	x^{27}	x^{26}	x^{25}	x^{24}
PTB4	0C	x^{39}	x^{38}	x^{37}	x^{36}	x^{35}	x^{34}	x^{33}	x^{32}
PTB5	0D	x^{47}	x^{46}	x^{45}	x^{44}	x^{43}	x^{42}	x^{41}	x^{40}

Tap Assignment 32-Bit Mode

REG #	ADDR	BIT NUMBER							
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PTB0	08	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0
PTB1	09	x^{15}	x^{14}	x^{13}	x^{12}	x^{11}	x^{10}	x^9	x^8
PTB2	0A	1	1	1	1	1	1	1	1
PTB3	0B	1	1	1	1	1	1	1	1
PTB4	0C	x^{23}	x^{22}	x^{21}	x^{20}	x^{19}	x^{18}	x^{17}	x^{16}
PTB5	0D	x^{31}	x^{30}	x^{29}	x^{28}	x^{27}	x^{26}	x^{25}	x^{24}

PROGRAMMING PRESET PATTERN

To program the preset pattern that the shift registers will be preset to, PPB0–PPB5 must be initialized. As in the polynomial taps, x^{48}, x^{32}, and x^0 are implied. The assignment of the bits for 48 and 32 bit modes is shown in the tables on the following pages.

The value programmed into each register will be the preset pattern for the eight bits of the corresponding shift register. For typical operation, these will be programmed to all 1's. All unused presets must be set to 0. In 32-bit mode, PPB2 and PPB3 must be set to all 0's. Failure to do so will result in improper operation.

Preset Bit Assignment 48-Bit Mode

REG #	ADDR	BIT NUMBER							
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PPB0	02	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0
PPB1	03	x^{15}	x^{14}	x^{13}	x^{12}	x^{11}	x^{10}	x^9	x^8
PPB2	04	x^{23}	x^{22}	x^{21}	x^{20}	x^{19}	x^{18}	x^{17}	x^{16}
PPB3	05	x^{31}	x^{30}	x^{29}	x^{28}	x^{27}	x^{26}	x^{25}	x^{24}
PPB4	06	x^{39}	x^{38}	x^{37}	x^{36}	x^{35}	x^{34}	x^{33}	x^{32}
PPB5	07	x^{47}	x^{46}	x^{45}	x^{44}	x^{43}	x^{42}	x^{41}	x^{40}

Preset Bit Assignment 32-Bit Mode

REG #	ADDR	BIT NUMBER							
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
PPB0	02	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0
PPB1	03	x^{15}	x^{14}	x^{13}	x^{12}	x^{11}	x^{10}	x^9	x^8
PPB2	04	0	0	0	0	0	0	0	0
PPB3	05	0	0	0	0	0	0	0	0
PPB4	06	x^{23}	x^{22}	x^{21}	x^{20}	x^{19}	x^{18}	x^{17}	x^{16}
PPB5	07	x^{31}	x^{30}	x^{29}	x^{28}	x^{27}	x^{26}	x^{25}	x^{24}

RECOMMENDED POLYNOMIAL AS AN EXAMPLE

To program the 32-bit polynomial of the form:

$$x^{32} + x^{28} + x^{26} + x^{19} + x^{17} + x^{10} + x^6 + x^2 + 1$$

with a preset of all 1's, a correction span of 5-bits with no header/data encapsulation, the following registers would be programmed as shown. Note that PTB2 and PTB3 must be all 1's and PPB2 and PPB3 must be all 0's in 32-bit mode.

6.0 CRC/ECC (Continued)

Polynomial Taps

REG #	BIT NUMBER 7	6	5	4	3	2	1	0
PTB0	1	0	1	1	1	0	1	0
PTB1	1	1	1	1	1	0	1	1
PTB2	1	1	1	1	1	1	1	1
PTB3	1	1	1	1	1	1	1	1
PTB4	1	1	1	1	0	1	0	1
PTB5	1	1	1	0	1	0	1	1

Preset Pattern

REG #	BIT NUMBER 7	6	5	4	3	2	1	0
PTB0	1	1	1	1	1	1	1	1
PTB1	1	1	1	1	1	1	1	1
PTB2	0	0	0	0	0	0	0	0
PTB3	0	0	0	0	0	0	0	0
PTB4	1	1	1	1	1	1	1	1
PTB5	1	1	1	1	1	1	1	1

ECC Control Register

BIT #	7	6	5	4	3	2	1	0
SET	1	0	0	1	0	1	0	1

6.3 OPERATION DURING CORRECTION

The DDC can be set to correct an error any time one has been detected and before another operation has begun. The user decides when to initiate the correction. The sector in question can be re-read several times to insure that the error is repeatable. If so, the error can be considered a hard error on the disk and a correction can be attempted. Since the DDC does not contain drive control circuitry, it is the user's responsibility to provide the programming for the execution of any re-read operations and the associated decision making.

The syndrome bytes in the ECC shift register will contain the bit error information. The bytes in error will already have been transferred to memory. Once initiated, the correction is performed internal to the DDC, leaving the bus free for other operations. An interrupt will be issued within the time it takes to read a sector, indicating whether the error was corrected or not. During this time, the erroneous sector in memory will remain unchanged.

Error correction time is determined by the error's location in the sector. The nearer to the start of the sector, the longer the DDC takes to locate the error. This time can be determined using the formula shown at right. It should be noted that this is internal correction time only; more time is required for the microprocessor to perform additional operations.

Before initiating a correction operation, the DDC needs to be reset, and re-enabled (see Operating Modes in DDC OPERATION). The Sector Byte Count registers must be initialized to *[sector length]* + 4 for 32-bit mode or *[sector length]* + 6 for 48-bit mode. The correction command should be issued when the counter has been updated.

The DDC will issue an interrupt after the correction cycle is complete. Other activities (such as completion of remote DMA) may issue interrupts before this happens. These interrupts should be serviced to allow the Correction Cycle Complete interrupt to be issued. The CCA bit in the Status register will be high during the entire correction cycle. It will be reset when the cycle has completed. The ED bit in the Status register will remain active throughout the correction cycle.

If after an interrupt, the Status register is read and the CCA bit is low, the Error register is read to see if the correction was successful. If the CF bit is set, this signifies that the error was non-correctable. This usually means that two errors have occurred with extremities exceeding the selected correction span. Failure to correct an error is serious and the system should be notified that the data from that sector is erroneous.

If the CF bit was not set, the error was corrected. The microprocessor then computes the address of the first byte in the data field that contains the error. That address is: *[current value of DMA Address Bytes 0 & 1] - [Sector Byte Count L & H]* + *[Data Byte Count L & H]* − 1.

Errors are corrected by XOR'ing syndrome bytes (ECC SR Out 0–5) with the bytes in the data record in memory that contain the error. The Data Byte Count can be used to determine whether the error is in the ECC or data field. If the Data Byte Count is greater than the maximum sector length, the error is in the ECC field and no correction should be attempted. If the Data Byte Count is less than the sector length, the error is in the data field (or it may straddle the data and ECC fields) and may be corrected.

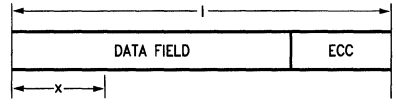
For performing a correction with 32-bit ECC, the following shift registers should be read sequentially to obtain the syndrome byte pattern:

ECC SR Out 1, ECC SR Out 4, ECC SR Out 5

ECC SR Out 2 and 3 are not used in 32-bit mode and will contain 0's if read. ECC SR Out 0 will contain all 0's if the error is correctable, and may contain some set bits if it is not.

ECC SR Out 1 will always contain the first bits in error. The succeeding bits will be contained in ECC SR Out 4 and 5. If the maximum span of 15 bits is used, all three registers may be needed, depending on where the first bit occurs.

To correct the error, the syndrome bits in these registers are XOR'ed with the data bits contained in buffer memory. The corrected data is then written back to the buffer memory, replacing the data in error. The address of the first byte in error is computed by the microprocessor as described above.



TL/F/5282–12

Approximate Correction Time = $(I - x)/f$

I = Entire length of data field and ECC appendage (in bits)

x = Distance from least significant bit to first error location (in bits)

f = read clock frequency (in hertz)

FIGURE 14. Calculating Correction Time

6.0 CRC/ECC (Continued)

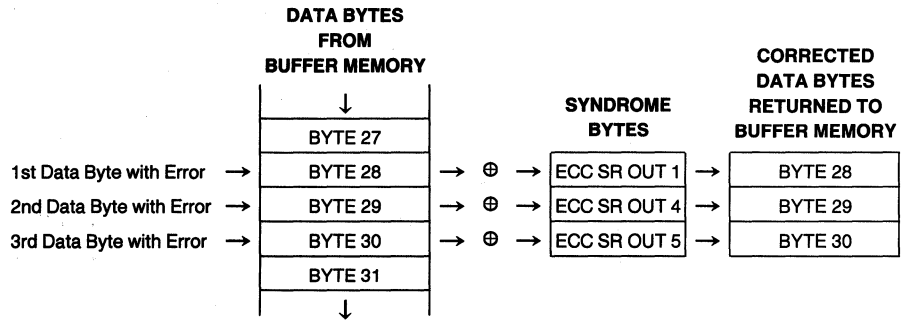


FIGURE 15. 32-Bit ECC Correction Process

To perform a 48-bit ECC correction, the following registers should be read sequentially:

ECC SR Out 1, ECC SR Out 2, ECC SR Out 3

ECC SR Out 0, 4 and 5 are not used for outputting syndrome bits for correction in 48-bit mode and will contain 0's for a correctable error. If the error is non-correctable, these registers may contain some set bits. Syndrome bit location and error correction is performed as in 32-bit mode.

EXAMPLE OF A 32-BIT CORRECTION

Shown in *Figure 17*, is a record with several bits read in error from disk. Bits D4, D11, D13 and D14, now located in memory, were incorrectly and need to be corrected. As can be seen, the correction pattern provided in ECC SR Out 1 and 2 can be used to correct bits D4, D11, D13 and D14. The CPU reads the Data Byte Count and computes that it points to the first byte read from disk. This byte is XOR'ed with ECC SR Out 1 and is written back to memory. The second byte read from the disk is XOR'ed with ECC SR Out 4 and then written back. ECC SR Out 5 need not be used since it contains all 0's.

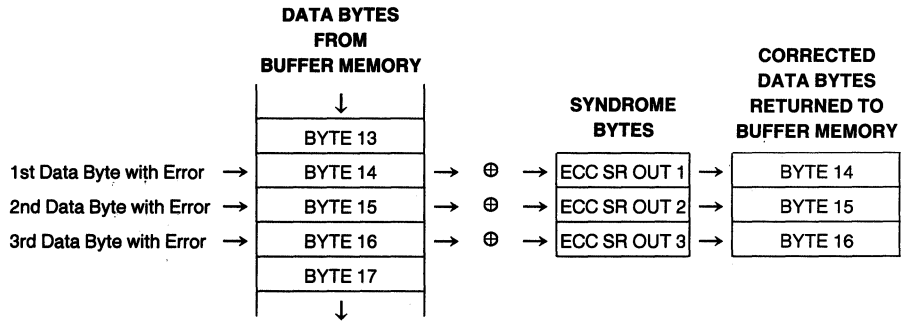


FIGURE 16. 48-Bit ECC Correction Process

Syndrome Pattern

REGISTER	*BIT NUMBER* 7	6	5	4	3	2	1	0
ECC SR OUT 1	0	0	0	1	0	0	0	0
ECC SR OUT 4	0	1	1	0	1	0	0	0
ECC SR OUT 5	0	0	0	0	0	0	0	0

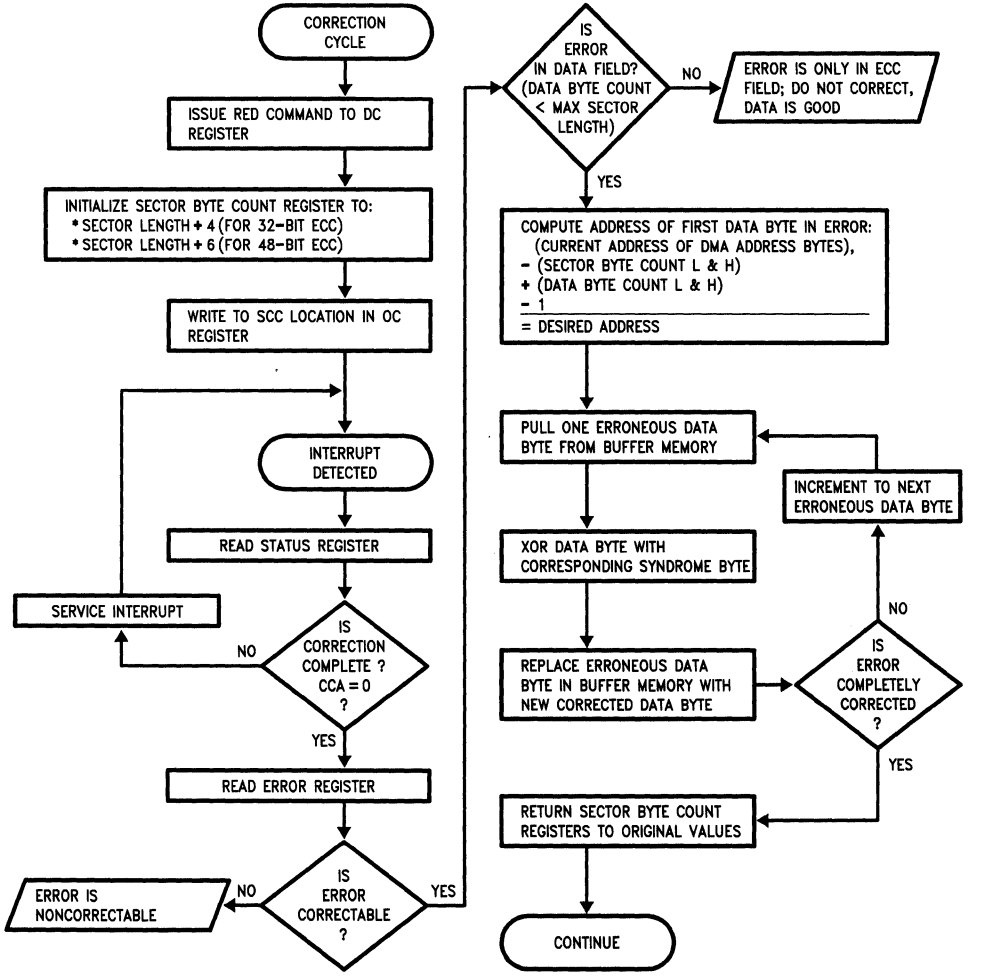
Buffer Memory

CORRESPONDING BUFFER DATA BIT PATTERN							
D7	D6	D5	*	D3	D2	D1	D0
D15	*	*	D12	*	D10	D9	D8
D23	D22	D21	D20	D19	D18	D17	D16

* = location of bits in error

FIGURE 17. Example of a 32-Bit Correction

6.0 CRC/ECC (Continued)



TL/F/5282–13

FIGURE 18. Correction Cycle Algorithm

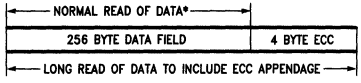
THIS CYCLE CAN ONLY BE INITIATED AFTER A READ DATA OPERATION HAS BEEN COMPLETED

6.0 CRC/ECC (Continued)

A note of caution: If the DDC is in the tracking DMA mode when a data error occurs, the remote DMA channel will transfer the sector in error to its destination in the system. The DDC will still interrupt to indicate that it has detected an error. It is then up to the system to get the DDC to correct the error in buffer memory and retransfer the corrected data to the system.

6.4 ECC CHECK USING LONG READ AND LONG WRITE

During a normal read or write operation, the size of the data field is specified by the Sector Byte Count register pair. If the data field is extended during a readback, the ECC appendage can be read in as data and analyzed outside the DDC. This is what is known as a *long read*.



TL/F/5282–14

*Read length defined by Sector Byte Count register pair.

FIGURE 19. Example of a Long Read

Likewise, an externally generated ECC appendage can be added to the data and written to the disk as data with or without the onboard CRC/ECC generator enabled. This is known as a *long write*.

By using long reads and long writes in conjunction with external software used to produce data fields and external CRC/ECC appendages, various diagnostic programs can be devised to test the DDC's internal correction functions and ECC generation circuitry. These tests could be incorporated in the initialization algorithm to test the chip each time it is powered up.

7.0 Data Transfer

7.1 DIRECT MEMORY ACCESS (DMA)

The DDC is designed to work efficiently in two major system configurations:

(1) A single system bus with shared data buffer/system memory (see *Figure 20*).

(2) A dual bus environment with a local microprocessor, buffer memory and DP8466 on a local bus interfacing the host system bus through an I/O port (see *Figure 21*).

All DMA activity is supported by the following three features:

PROGRAMMABLE BURST LENGTH (THRESHOLD)

Here, the transfer of data between the 32-byte FIFO on the DDC and the external memory (local or main) involves the use of internal or external local DMA channel. While writing to the disk, the DDC will initiate a transfer when the FIFO has been depleted by the burst length. It will also initiate a transfer while reading from the disk when the FIFO fills to the burst length. This length is selectable from 2, 8, 16 or 24 bytes, allowing for the variations in bus latency time encountered in most systems.

At the start of a write operation, the FIFO will be filled up in a series of bursts of the programmed length.

If the exact burst option is not selected, the FIFO will be completely filled (if writing to disk) or emptied (if reading from disk) in one DMA operation. The burst length is always the threshold at which the transfer will be requested and is independent of the DMA mode, including slave.

8-BIT/16-BIT WIDE TRANSFERS

Byte or word wide data transfer can be selected for both local and remote DMA channels. Word wide tranfers with local DMA use the AD0–15 pins, and byte wide use the AD0–7 pins. Both the local and the remote DMA addresses are incremented by 2 for word wide transfers, and 1 for byte wide transfers. Commands and DDC parameter registers are loaded and read only 8-bits at a time, using AD0–7.

REVERSE BYTE ORDER

This option is only valid for 16-bit wide transfers using the local DMA channel. This should not be used for 8 bit wide transfers. It enables the two bytes being transferred to be mapped with the high order byte to AD0–7 and the low order byte to AD8–15, or vice-versa.

The DDC has provisions to accommodate five DMA modes. These are as follows:

EXTERNAL DMA:	1. Slave Mode
INTERNAL DMA, Single Bus:	2. 16-Bit Local Mode
	3. 32-Bit Local Mode
Multiple Bus:	4. Non-Tracking Mode
	5. Tracking Mode

All five modes accommodate the three configurations just described. All DMA modes, except external slave, use an incrementing address. Local channel transfers always have priority over remote channel transfers unless externally reprioritized. If the local channel is used, its transfer length is always automatically loaded from the Sector Byte Count register pair.

7.2 EXTERNAL DMA

SLAVE MODE

In this mode, no on-chip DMA control is used. LRQ and LACK pins are connected to an external DMA controller. After LACK has been granted, I/O RD and I/O WR from the DMA controller are used to strobe data between the internal FIFO and the DDC I/O port. 8-bit and 16-bit wide data transfers are possible. Throughout this data sheet, reference has been made to the use of on-chip DMA for the transfer of data. It is important to note here that external DMA can be used in place of this if so desired.

7.3 INTERNAL DMA

The following four modes all use on-chip DMA control with at least the local channel serving as bus master for data transfers between the internal FIFO and memory.

SINGLE BUS SYSTEMS

The following two modes support a single bus and a single shared buffer/system memory. Bus access should be guaranteed before the FIFO overflows or empties during a disk transfer operation. A FIFO Data Lost error (FDL bit in Error register) will be flagged and the operation aborted if this fails to happen. Different system latency times can be accommodated by the selectable burst length.

16-BIT LOCAL MODE

SLD bit is set and LA bit is reset in the LT register. Only the 16-bit local DMA channel is enabled. 64k bytes are directly addressable by the DDC. Address data is presented on AD0–15 and latched with ADS0. Transfers always take 4 BCLK cycles if no wait states are issued.

7.0 Data Transfer (Continued)

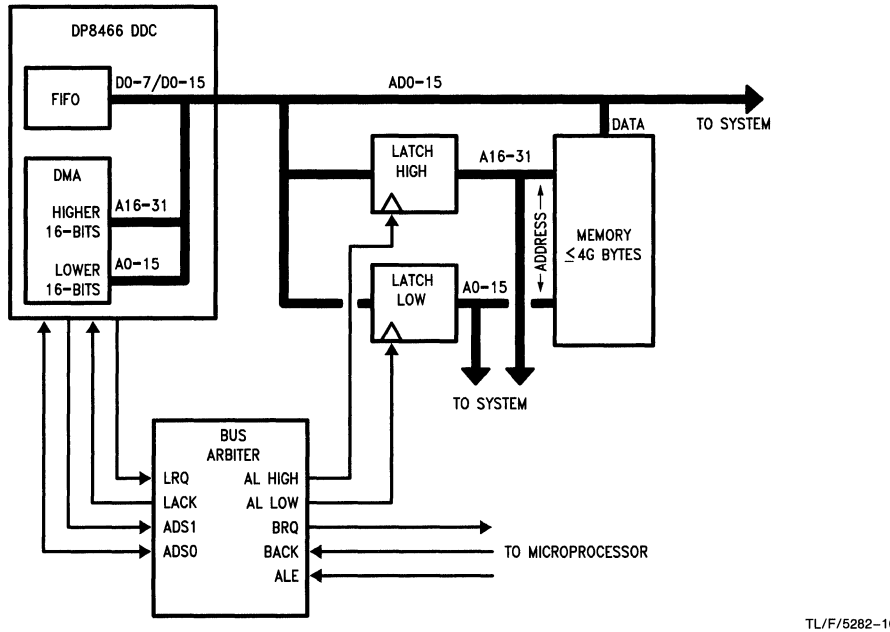


FIGURE 20. Single System Bus, 32-Bit Address DMA

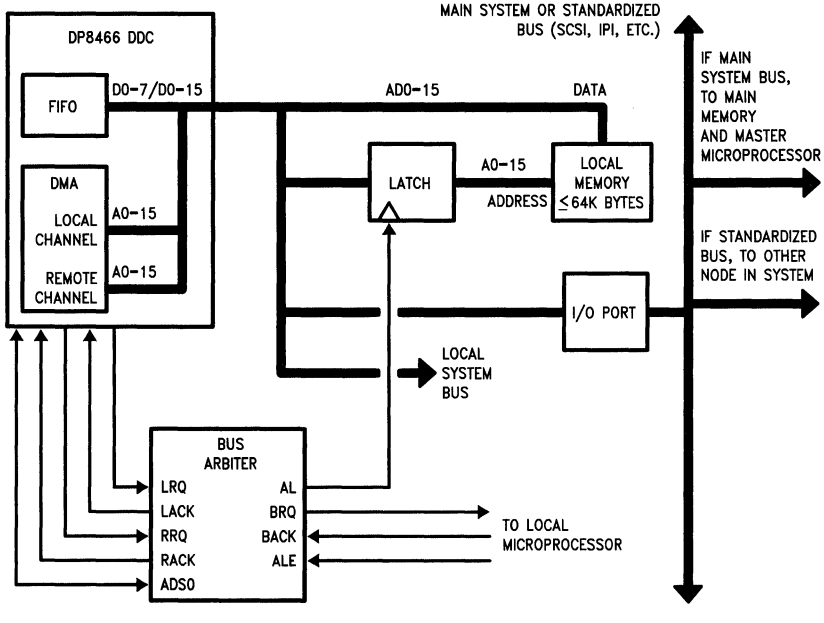


FIGURE 21. Dual System Bus, 16-Bit Address DMA

7.0 Data Transfer (Continued)

32-BIT LOCAL MODE

SLD bit and LA bit are both set in the LT register. SRD bit in the RT register must be reset. The local DMA channel is now set to issue 32-bit addresses using the remote DMA channel as the upper 16-bit address register. 4 G bytes are addressable by the DDC. During the first DMA cycle of a newly programmed address, or after a roll-over of the lower 16-bit address counter occurs, ADS1 strobes a new high order word (A16-31) into the external address latches. Each time this happens, the DMA cycle is 5 BCLK periods long. When a new high order address is not needed, the DMA cycle is 4 BCLK periods long. ADS0 is used as an output to latch the low order word (A0–15) from the AD0–15 pins into the address latch.

MULTIPLE BUS SYSTEMS

The following two modes support a dual bus environment, where a local microprocessor, buffer memory and the DP8466 interface to the host through an I/O port. The difference between tracking and non-tracking mode is whether the DDC or the controlling microprocessor ensures that an attempt to read data from buffer memory does not occur before data has been written there. Basic algorithms for both are shown in *Figures 22* and *23*.

TRACKING MODE

SLD bit set and LA bit reset in the LT register. SRD bit and TM bit set in the RT register. The DDC ensures that data is not overwritten by data transferred from the FIFO.

This mode effectively turns the buffer memory into a large FIFO. This is accomplished through the use of the DMA Sector Counter (DSC), which keeps track of the difference between sectors read/written to the disk and the sectors transferred to/from the host system. Each time the source transfers a sector of data into buffer memory (length determined by the Sector Byte Count register pair), the DSC register is incremented. It is decremented each time the destination has transferred a sector of data. Whenever the DSC register contents become zero, destination transfers are inhibited. This mode facilitates multi-sector operations.

Example: Tracking Mode, Disk Read

- Source is local DMA
- Destination is remote DMA
- DSC register is reset automatically upon start of operation
- Local and remote start address, SC, NSO, OC and finally DC registers are loaded. Other registers may need to be updated, but this is a minimum set.

A sector is read from the disk and is transferred in bursts from the FIFO to the buffer memory by local DMA. The DSC register then increments and the remote channel can begin transferring the first sector from the buffer memory to the host system. Burst transfers can be interleaved with local DMA, remote DMA and microprocessor all sharing the bus. The local channel bursts have priority over remote bursts. If the remote channel manages to transfer a sector before the local channel has completed the next sector, the DSC register will decrement to zero. Further remote transfers are inhibited until the local channel completes another sector and increments the DSC. In other words, each time a local sector has been transferred, the DSC is incremented and each time a remote sector completes, the DSC is decremented. Therefore, the DDC prevents further buffer memory contents that have not been previously loaded with valid data by the local DMA from being transferred to the host system. The remote channel continues operation until the last byte from the buffer memory has been transferred. An interrupt is issued upon completion of the operation.

NON-TRACKING MODE

SLD bit set and LA bit reset in the LT register. SRD bit set and TM bit reset in the RT register. The remote and local channel addresses are completely independent. The controlling microprocessor must insure that the data to be transferred by the remote channel is not over-written by the local channel and vice-versa. DMA address and count registers are set up independently. Remote start address (DMA Address Bytes 2 and 3) and Remote Data Byte Count registers must be loaded before SRI or SRO bits are set in the OC register. Local or remote transfers may already be in progress when the other channel is started. The local channel has priority over the remote channel. Local bus utilization is then interleaved between the local channel, the remote channel and the controlling microprocessor.

By setting both SRI and SRO simultaneously, any non-tracking remote DMA operation will stop. The present remote address and remote data byte count will be retained and the local DMA will be unaffected. Loading the original OC instruction (input or output) will restart the original instruction from the last remote DMA address.

DMA Mode Select Table

DMA Mode	LT Register		RT Register	
	SLD	LA	SRD	TM
SLAVE	0	0	0	0
16-BIT LOCAL	1	0	0	0
32-BIT LOCAL	1	1	0	0
TRACKING	1	0	1	1
NON-TRACKING	1	0	1	0

NOTE: In either tracking or non-tracking mode, if either channel is loaded with an odd byte transfer count, the DDC will transfer the next higher even number of bytes. For example, if 511 was loaded into the Remote Data Byte Count registers, 512 bytes would be transferred, with valid data only in the first 511 bytes.

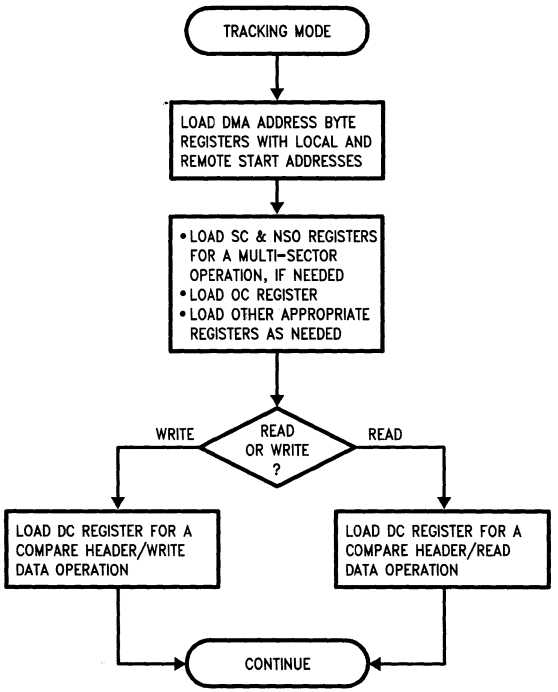
DMA WAIT STATES

INTERNAL

Both DMA channels can independently be set to lengthen the RD and WR strobes by one clock cycle (LSRW bit in the LT register and RSRW bit in the RT register). This lengthens each transfer from 4 cycles to 5 cycles of the BCLK.

EXTERNAL

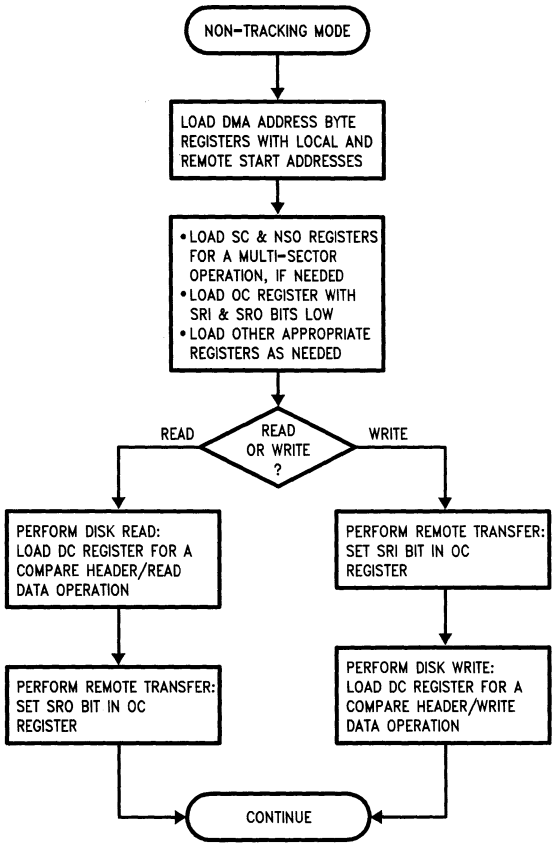
By enabling the external wait states (in the RT register), the EXT STAT pin is configured to insert wait states in each RD and WR pulse as long as this input is high. This is valid for both the local and remote DMA channels.



TL/F/5282–17

NOTE: DMA operation is completely automatic for the duration of the command. For example, when reading disk, local DMA empties/fills the FIFO and remote DMA transfers data at least one sector behind the local channel to an I/O port. For disk write, local channel will be at least one sector behind the remote channel.

FIGURE 22. Tracking Mode for Normal Disk Read/Write



TL/F/5282–18

NOTE: This is the most basic of non-tracking mode operations, and unlimited, more versatile algorithms can be built up from this.

FIGURE 23. Non-Tracking Mode for Normal Disk Read/Write

8.0 Interrupts

Interrupts can only occur if the EI bit in the OC register is set. If it is not set, the INT pin is always de-asserted high. 16 RCLK periods (3.2 μs at 5 Mbit/sec data rate) must pass before servicing an interrupt (i.e. reading Status). Failure to do this will result in servicing the same interrupt twice. There are four general conditions that may cause an interrupt to occur:

Operation Complete

Header Complete

Error

Correction Cycle Complete

OPERATION COMPLETE

This interrupt indicates that the current DDC operation has completed and the DDC is ready to execute a new command. Commands can be loaded sooner by setting EHI bit in the OC register. The Next Disk Command (NDC) bit in the Status register is set coincident with the Header Complete interrupt. New disk commands can be loaded before DMA operation is finished if NDC is set. If the command is a multi-sector operation, the end of operation interrupt will occur only after the operation is completed in the last sector of operation. The INT pin is asserted low when:

- Disk operation is completed for any command that is not a disk read operation.
- A read operation in the tracking DMA mode after the remote transfer is complete.
- A read operation in the non-tracking DMA mode after the local transfer is complete.
- A non-tracking mode remote DMA transfer is completed. This is independent of the disk operation or the local DMA.

HEADER COMPLETE:

If the EHI and EI bits are set in the OC register, an interrupt will occur when any header operation is complete. Multi-sector operations will generate an interrupt after each header in each sector has been operated on. It is asserted two bit times into the ID postamble. This function allows the changing of header bytes (and parameter RAM in general) *on the fly*. The Header Complete interrupt can be used in conjunction with the Interlock Required (IR) bit in the OC register set to insure that changes have been completed before the next sector is encountered (see Interlock Type formatting). Another normal mode of use would be to notify the controlling microprocessor when the next disk command can be loaded. This interrupt is coincident with the Next Disk Command (NDC) bit being set in the Status register.

ERROR

Any bit set in the Error register sets the ED bit in the Status register and causes an interrupt.

CORRECTION CYCLE COMPLETE

An interrupt will occur at the end of an internal correction cycle, regardless of whether the error was corrected or not. If the error was non-correctable, the CF bit will be set in the Error register. This will not generate two interrupts.

CLEARING INTERRUPTS

The INT pin will be forced inactive high any time the Status register is being read. If an interrupt condition arises during a status read, this condition will assert INT as soon as the status read is finished.

Interrupts can also be cleared by setting the internal RES bit, or by asserting the external RESET pin.

9.0 Additional Features

9.1 DATA RECOVERY USING THE INTERLOCK FEATURE

The potential use of the interlock feature is in recovering data from a sector with an unreadable header field. It is assumed that the number of the sector physically preceding the bad sector on the disk is known. A single-sector operation will be performed on these sectors, and the Drive Command register will be changed in between them. The following steps will recover the data:

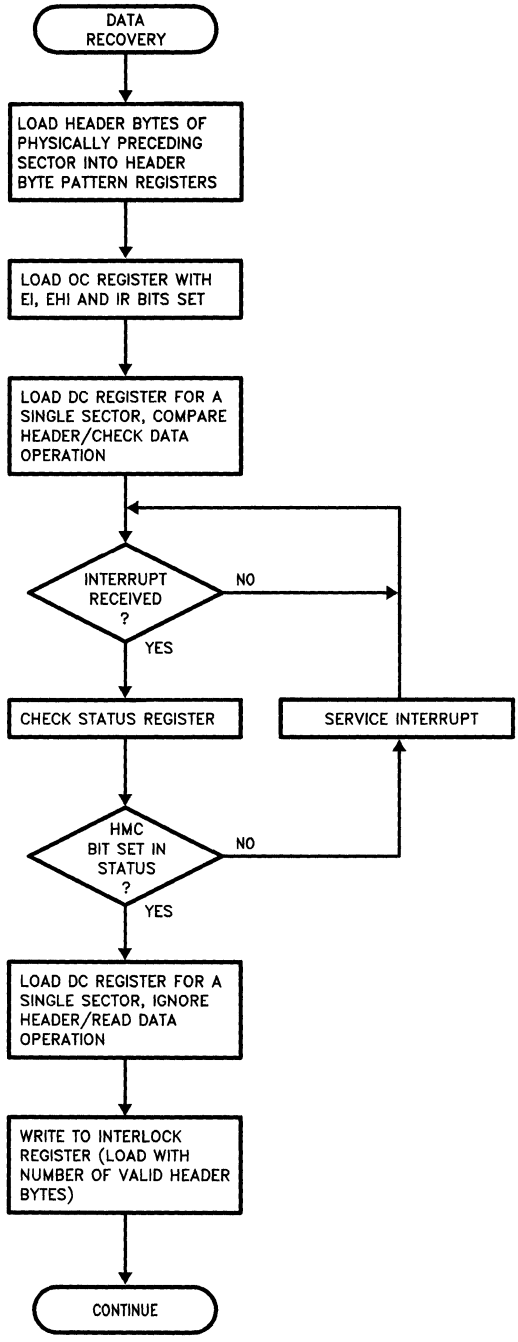
- The header bytes of the physical sector preceding the desired sector are loaded into the relevant byte pattern registers.
- The OC register must be loaded with the EI, EHI and IR bits set. This enables the Header Complete interrupt as well as the interlock feature.
- The DC register is loaded for a single-sector, Compare Header/Check Data operation.
- After the Header Complete interrupt, the DC register must be loaded with an Ignore Header/Read Data operation, and the Interlock (HBC) register written to. If the controlling microprocessor fails to write to the HBC register before the end of the data field of the first sector, a Late Interlock error (LI bit in Error register) will be flagged, and the operation will be terminated with an interrupt.
- When the HMC interrupt occurs on the second sector, the Interlock (HBC) register must be written to again in order to avoid LI error.
- The operation will terminate normally when the data from the badly labeled sector has been read.

9.2 HFASM FUNCTION

The Header Failed Although Sector number Matched (HFASM) function on the DDC can be used to perform maintainance and diagnostic functions, both of which will be briefly outlined here.

The HFASM function is enabled by setting the EHF bit in at least one of the Header Control registers, with a Compare Header command loaded into the DC register. More than one header byte may have its EHF bit set. If any one of the header byte(s) with it's EHF bit set matched, but any other header byte(s) (regardless of the state of their EHF bit) don't match, an HFASM error will occur.

9.0 Additional Features (Continued)



TL/F/5282–19

FIGURE 24. Data Recovery Algorithm

9.0 Additional Features (Continued)

In this way, the HFASM function performs a maintenance type function, and can often indicate that the head is positioned over the wrong track. It is independent of whether or not a CRC failure has occurred. An HFASM failure will not stop operation until the header CRC bytes have been compared and the CRC check is completed.

To perform a diagnostic function, the header can be read and analyzed. This can be done only during a Compare Header/Check Data operation with HFASM enabled. This causes the header patterns coming from the disk to be written into the FIFO. We must assume that the FIFO is empty (or has been reset before the operation) in order for this operation not to interfere with data transfers. If an HFASM error occurs during a Header Compare, the FIFO will be left intact and the header with the error can be read out of the FIFO from the Header Diagnostic Readback (HDR) register. (Note: LWDT of the local transfer register must be set to match the bus width of the accessing MP for this function.)

If an HFASM error did not occur, the FIFO will be cleared and the header patterns that were stored there will be lost. This process can only be enabled for one disk command. The Compare Header/Check Data command will enable this function. Any other command will disable it.

10.0 Typical System Configurations

10.1 LOW COST SYSTEM

In a single bus system, the DDC can directly address 4G bytes of main memory. The 16-bit I/O port (AD0–15) is externally demultiplexed and buffered with the octal latches and drivers. The main microprocessor, through a separate disk drive control I/O block, is responsible for commands like Head Select, Seek, TRK 000, Drive select, etc. Bus access must be guaranteed before the FIFO overflows or empties. A short burst length (LT and RT registers) accommodates longer bus latency times and helps to insure this. The burst capability allows for other bus operations to be interleaved while the FIFO is filling (during a read) or emptying (during a write). If long, important CPU operations are required, the next configuration must be used.

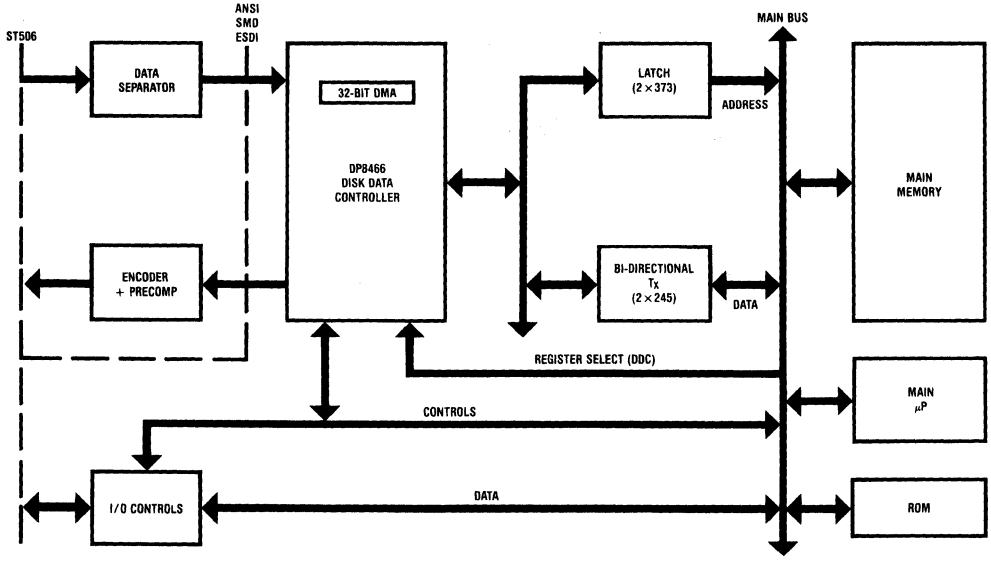


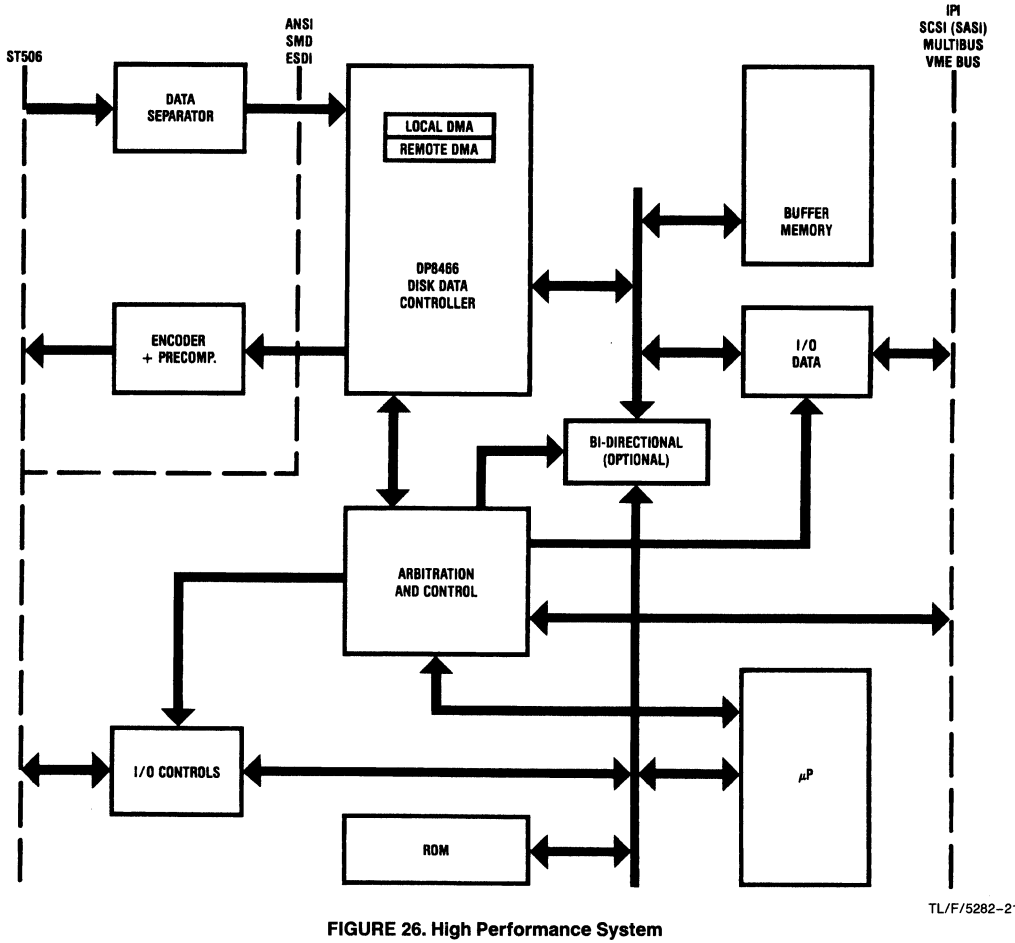
FIGURE 25. Low Cost System Configuration

TL/F/5282–20

10.0 Typical System Configuration (Continued)

10.2 HIGH PERFORMANCE SYSTEM

This configuration provides a local bus for the DDC to share with the local microprocessor and a buffer memory. Here, whole blocks of data can be transferred between the DDC and buffer memory without interfering with the system bus. This leaves the main CPU to perform important operations and to allow data transfers when it is ready. This configuration is also used in intelligent drives or systems that comply to SCSI or IPI specifications. A local bus, dedicated microprocessor and buffer memory are main characteristics of an intelligent disk interface. The buffer memory can be used as a cache for track or file buffering and command lists can be down-loaded for execution by the microprocessor. The two DMA channels can both directly address 64k bytes of buffer memory. The local DMA channel transfers data between the buffer memory and the internal FIFO. The remote DMA channel transfers data between the buffer memory and the host I/O port. With the addition of a bi-directional buffer isolating the DDC from the microprocessor, simultaneous drive operations can be accomplished. While the DDC is transferring data via DMA with the buffer memory, the local microprocessor can issue drive control commands.



TL/F/5282–21

FIGURE 26. High Performance System

INDEX

11.0 ABSOLUTE MAXIMUM RATINGS

12.0 DC ELECTRICAL CHARACTERISTICS

13.0 AC ELECTRICAL CHARACTERISTICS & TIMING DIAGRAMS

13.1 Register Read (Latched Register Select: ADS0 Active)
13.2 Register Read (Non-Latched Register Select: ADS0 = 1)
13.3 Register Write (Latched Register Select: ADS0 Active)
13.4 Register Write (Non-Latched Register Select: ADS0 = 1)
13.5 LRQ Timing with External DMA
13.6 Reading FIFO Data in DMA Slave Mode
13.7 Writing FIFO Data in DMA Slave Mode
13.8 Local and Remote DMA Acknowledge
13.9 DMA Address Generation
13.10 DMA Memory Write
13.11 DMA Memory Read
13.12 DMA with Internal Wait States
13.13 DMA with External Wait States
13.14 DMA Control Signals
13.15 Local and Remote DMA Interleaving
13.16 RRQ Assertion after Writing to OC Register for a Remote Transfer
13.17 Read Data Timing
13.18 RGATE Assertion from Index or Sector Pulse Input
13.19 Write Data Timing for NRZ Type Data
13.20 WGATE Assertion from Index or Sector Pulse Input
13.21 Write Data Timing for MFM Type Data
13.22 Positional Timing for SDV and EEF
13.23 Field Envelope Timing
13.24 EXT STAT Timing When Used as External Byte Synch
13.25 EXT STAT Timing When Using External ECC

14.0 AC TIMING TEST CONDITIONS

15.0 MISCELLANEOUS TIMING INFORMATION

15.1 Status Register Timing
15.2 Error Register Timing
15.3 General Timing for Read Gate
15.4 Write Gate Timing
15.5 Normal Interrupts
15.6 Derating Factor

11.0 Absolute Maximum Ratings*

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−0.5 to V_{CC}+0.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC}+0.5V
Storage Temperature Range (TSTG)	−65°C to +150°C
Power Dissipation (PD)	500 mW
Lead Temperature (TL) (Soldering 10 sec.)	260°C

ESD maximum rating: All pins are capable of withstanding 1600 volts.

*Absolute Maximum Ratings are those values beyond which damage to the device may occur.

12.0 DC Electrical Characteristics (V_{CC} = 5V ±10%, unless otherwise specified) T_A = 0°C to +70°C

Symbol	Parameter	Conditions	Typ	Limit	Units
V_{IH}	Minimum High Level Input Voltage	(Note 1)		2.0	V
V_{IL}	Maximum Low Level Input Voltage	(Note 1)		0.8	V
V_{OH1}	Minimum High Level Output Voltage (Note 2)	$\|I_{OUT}\|$ = 20 μA		V_{CC} − 0.1	V
V_{OH2}		ADS0, ADS1 $\|I_{OUT}\|$ = 4.0 mA For All Other Outputs $\|I_{OUT}\|$ = 2.0 mA		3.5	V
V_{OL1}	Minimum Low Level Output Voltage (Note 2)	$\|I_{OUT}\|$ = 20 μA		0.1	V
V_{OL2}		ADS0, ADS1 $\|I_{OUT}\|$ = 4.0 mA For All Other Outputs $\|I_{OUT}\|$ = 2.0 mA		0.4	V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND		±1	μA
I_{OZ}	Maximum TRI-STATE Output Leakage Current	V_{OUT} = V_{CC} or GND		±10	μA
I_{CC}	Average Supply Current DP8466N-12 (Note 3)	V_{IN} = V_{CC} or GND BCLK = RCLK = 12 MHz I_{OUT} = 0 μA	12	30	mA
	Average Supply Current DP8466N-20 (Note 3)	V_{IN} = V_{CC} or GND RCLK = 20 MHz BCLK = 16 MHz, I_{OUT} = 0 μA	20	40	mA
	Average Supply Current DP8466N-25 (Note 3)	V_{IN} = V_{CC} or GND BCLK = 20 MHz RCLK = 25 MHz I_{OUT} = 0 μA	25	45	mA

12.0 DC Electrical Characteristics (Continued)

Note 1: Limited functional test patterns are performed at these levels. The majority of functional test patterns are performed with input levels of 3V for AC Timing Verification.

Note 2: Outputs are "conditioned" for Tested States by normal functional test patterns. Device clocks are disabled and a purely static measurement is performed.

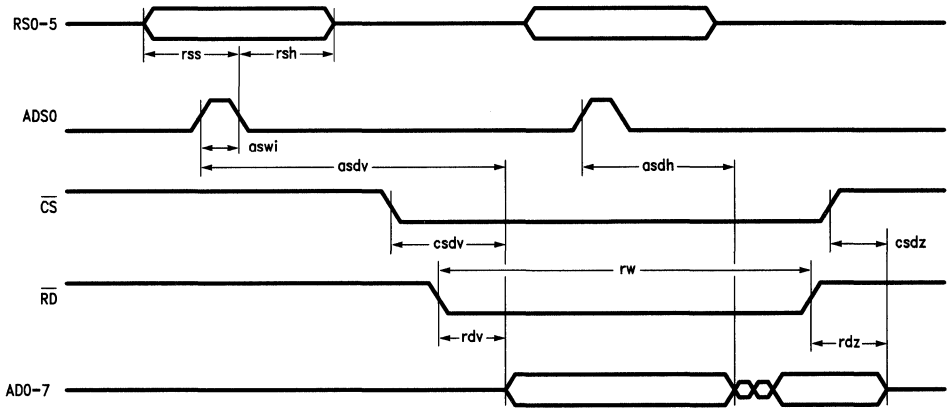
Note 3: Device is in normal operating mode and is measured with bypass capacitor of 0.1 μF between V_{CC} and Ground.

13.0 AC Electrical Characteristics & Timing Diagrams

NATIONAL SEMICONDUCTOR PRELIMINARY TIMING FOR THE DP8466

Note: Refer to 11.4 for AC Timing Test Conditions.
Refer to 11.5.6 for derating factor.

13.1 REGISTER READ (Latched Register Select: ADS0 Active)



TL/F/5282-22

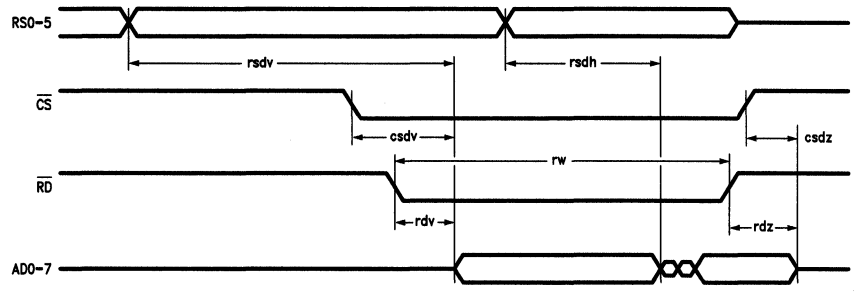
Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rss	Register Select Setup to ADS0 Low	10		15		ns
rsh	Register Select Hold to ADS0 Low	10		15		ns
aswi	Address Strobe Width In	20		30		ns
asdv	Address Strobe to Data Valid (Note 1)		150		200	ns
csdv	Chip Select to Data Valid		125		150	ns
rdv	Read Strobe to Data Valid		125		150	ns
rw	Read Strobe Width		10		10	μs
csdz	Chip Select to Data TRI-STATE (Note 2)	20	80	20	90	ns
rdz	Read Strobe for Data to TRI-STATE (Note 2)	20	80	20	90	ns
asdh	Data Hold from ADS0 (Note 1)	20		20		ns

Note 1: asdv and asdh timing is referenced to the leading edge of ADS0 or the leading edge of valid address, whichever comes last.

Note 2: TRI-STATE note: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive this line with no contention.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.2 REGISTER READ (Non-Latched Register Select: ADS0 = 1)



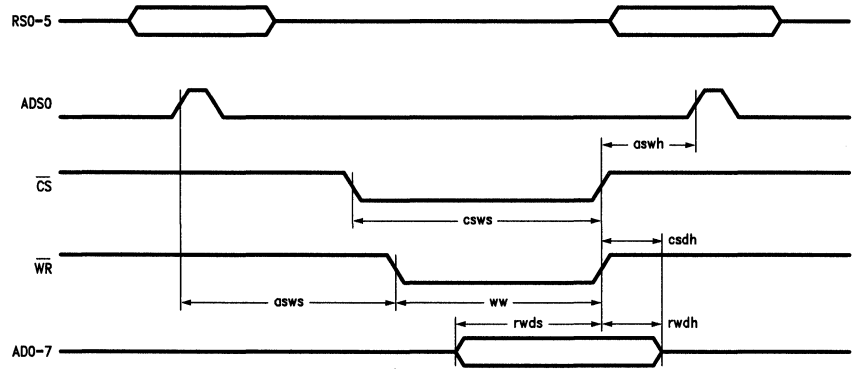
TL/F/5282–23

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rsdv	Register Select to Data Valid (ADS0 = 1) (Note 1)		150		200	ns
csdv	Chip Select to Data Valid		125		150	ns
rdv	Read Strobe to Data Valid		125		150	ns
rw	Read Strobe Width		10		10	μs
csdz	Chip Select to Data TRI-STATE (Note 2)	20	80	20	90	ns
rdz	Read Strobe for Data to TRI-STATE (Note 2)	20	80	20	90	ns
rsdh	Data Hold from Register Select Change (Note 1)	20		20		ns

Note 1: rsdv and rsdh timing assumes that ADS0 is true when RS0–5 changes.

Note 2: TRI-STATE note: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive this line with no contention.

13.3 REGISTER WRITE (Latched Register Select: ADS0 Active)



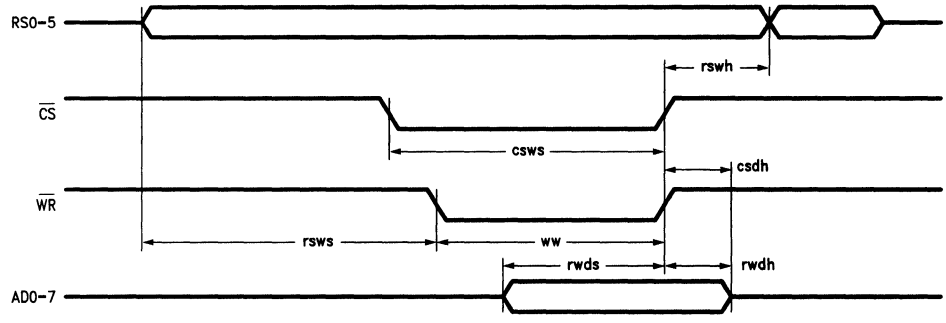
TL/F/5282–24

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
asws	Address Strobe to Write Setup (Note 1)	15		20		ns
csws	Chip Select to Write Setup	50		70		ns
csdh	Chip Select Data Hold	7		10		ns
rwds	Register Write Data Setup	40		50		ns
rwdh	Register Write Data Hold	3		5		ns
ww	Write Strobe Width	50		70		ns
aswh	ADS0 Hold from Write (Note 1)	10		15		ns

Note 1: asdv and asdh timing is referenced to the leading edge of ADS0 or the leading edge of valid address, whichever comes last.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.4 REGISTER WRITE (Non-Latched Register Select: ADS0 = 1)

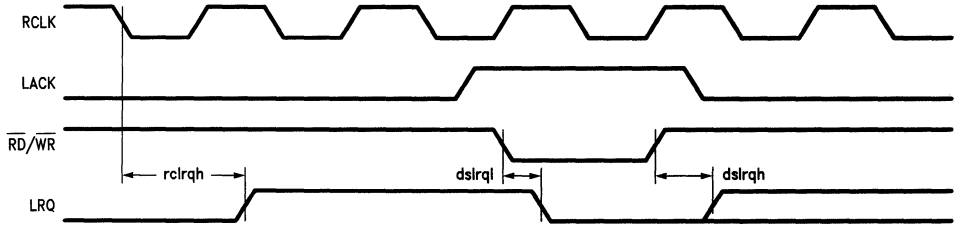


TL/F/5282–25

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rsws	Register Select to Write Setup (Note 1)	10		15		ns
csws	Chip Select to Write Setup	50		70		ns
csdh	Chip Select to Data Hold	7		10		ns
rwds	Register Write Data Setup	40		50		ns
rwdh	Register Write Data Hold	3		5		ns
ww	Write Strobe Width	50		70		ns
rswh	Register Select Hold from Write (Note 1)	15		20		ns

Note 1: rsws and rswh assume that ADS0 is true when RS0–5 changes.

13.5 LRQ TIMING WITH EXTERNAL DMA



TL/F/5282–26

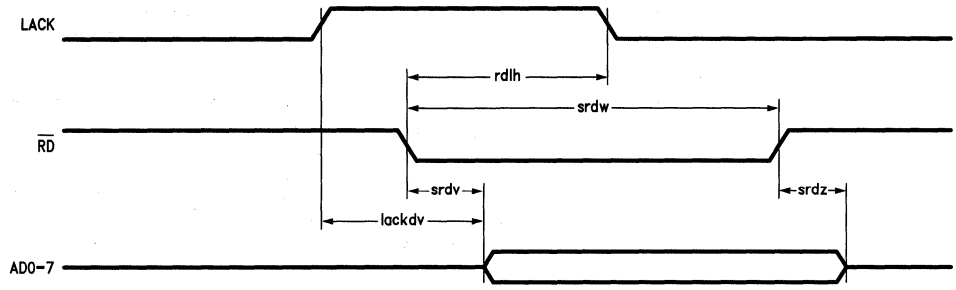
Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
dslrql	Data Strobe to LRQ Low (Note 1)		75		100	ns
dslrqh	Data Strobe to LRQ High (Note 2)		75		100	ns
rclrqh	Read Clock to LRQ High		75		100	ns

Note 1: LRQ is removed at the end of a burst or an operation by the assertion of the $\overline{RD}$ or $\overline{WR}$ strobe.

Note 2: LRQ is reissued at the end of a burst by the removal of the $\overline{RD}$ or $\overline{WR}$ strobe. This only occurs if sufficient data/space is in the FIFO for another burst and the DDC is operating in the exact burst mode.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.6 READING FIFO DATA IN DMA SLAVE MODE



TL/F/5282–27

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
lackdv	LACK to Data Valid		80		90	ns
rdlh	Read Strobe to LACK Hold (Note 1)	10		15		ns
srdv	Slave Read Strobe to Data Valid		60		70	ns
srdz	Slave Read Strobe to Data TRI-STATE (Note 3)	20	80	20	90	ns
srdw	Slave Read Strobe Width		8rcyc − 100		8rcyc − 100	ns

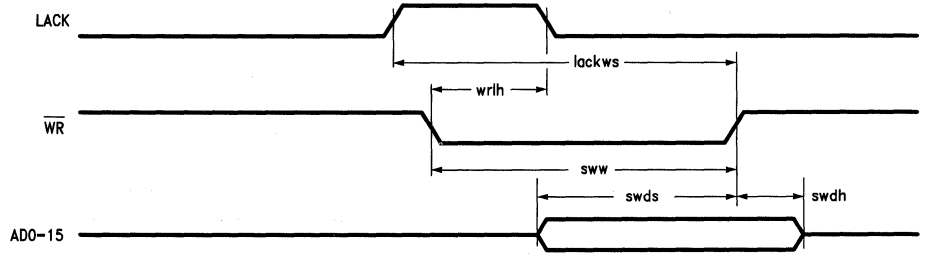
Conditions: Disk read operation, DMA disabled, LRQ output true.

Note 1: The read cycle begins when LACK and $\overline{RD}$ are true. From this point, LACK must be held true for rclh.

Note 2: The maximum rate of FIFO transfers is limited to 1 transfer per 2 rcyc + 75 ns while data is being transferred between the disk and the FIFO.

Note 3: TRI-STATE note: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive this line with no contention.

13.7 WRITING FIFO DATA IN DMA SLAVE MODE



TL/F/5282–28

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
lackws	Local Acknowledge to Write Setup	50		60		ns
wrlh	Write to LACK Hold (Note 1)	10		15		ns
swds	Slave Write Data Setup	5		10		ns
swdh	Slave Write Data Hold	20		28		ns
sww	Slave Write Strobe Width (Note 2)	30	8rcyc − 100	40	8rcyc − 100	ns

Conditions: Disk write operation, DMA disabled, LRQ output true.

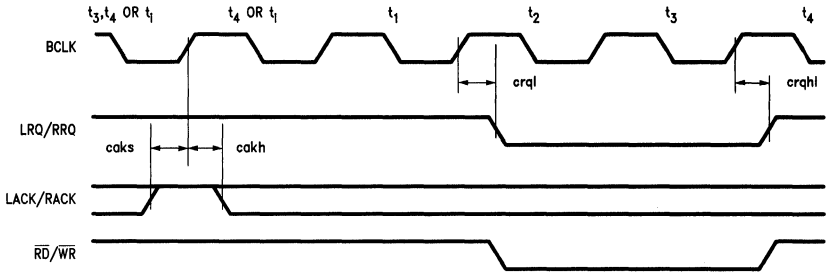
Note 1: The write cycle begins when LACK and $\overline{WR}$ are true. From this point, LACK must be held true for wrlh.

Note 2: The write cycle begins when LACK and $\overline{WR}$ are true. From this point, $\overline{WR}$ must remain true for sww.

Note 3: The maximum rate of FIFO transfers is limited to 1 transfer per 2 rcyc + 75 ns while data is being transferred between the disk and the FIFO.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.8 LOCAL AND REMOTE DMA ACKNOWLEDGE

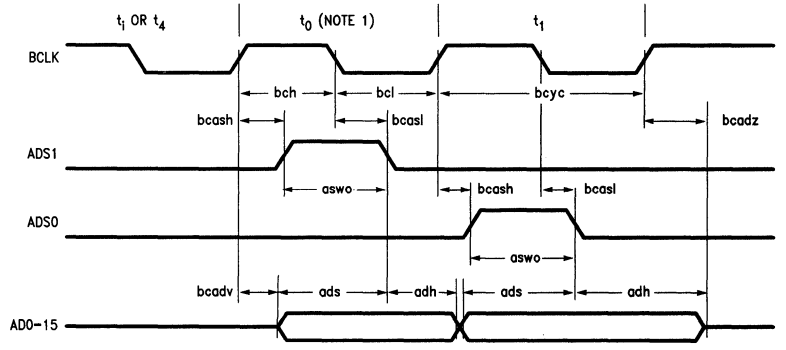


TL/F/5282–29

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
crql	t2 Clock to Request Low		80		100	ns
crqhl	t4 Clock to Request High (Exact Burst Length Limited)		80		100	ns
caks	Acknowledge Setup to Clock	20		25		ns
cakh	Acknowledge Hold from Clock	10		15		ns

Note 1: The Local and Remote Acknowledges are sampled at the beginning of bus cycles t4 and t1.
Note 2: Local Acknowledge has internal priority over Remote Acknowledge.
Note 3: Local and Remote Acknowledge are ignored if their respective Request output is false.
Note 4: Above timing is for 16 bit address updates. For 32 bit Local address mode, cycle t0 occurs on the first transfer of an operation or when the lower 16 bits of the address roll over.

13.9 DMA ADDRESS GENERATION



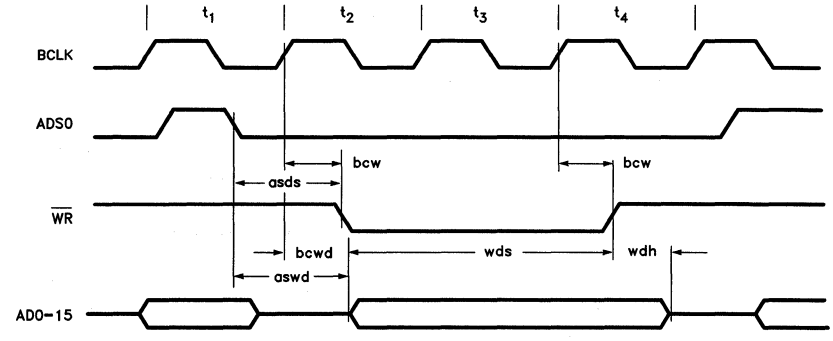
TL/F/5282–30

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
bcyc	Bus Clock Cycle Time (Notes 2, 3)	50	10,000	80	10,000	ns
bch	Bus Clock High Time (Note 3)	22.5	10,000	32	10,000	ns
bcl	Bus Clock Low Time (Note 3)	22.5	10,000	32	10,000	ns
bcash	Bus Clock to Address Strobe High		45		55	ns
bcasl	Bus Clock to Address Strobe Low		50		60	ns
aswo	Address Strobe Width In	bch		bch		
bcadv	Bus Clock to Address Valid		60		70	ns
bcadz	Bus Clock to Address TRI-STATE (Note 4)	20	80	20	90	ns
ads	Address Setup to ADS0/1 Low	bch − 20		bch − 27		ns
adh	Address Hold from ADS0/1 Low	bcl − 5		bcl − 10		ns

Note 1: Cycle t0 occurs only on the first transfer of an operation or when the lower 16 bits of the address rolls over.
Note 2: The rate of bus clock must be high enough that data will be transferred to and from the FIFO faster than the data being transferred to and from the disk.
Note 3: For DP8466-20, minimum bcyc = 60 ns minimum bch = bcl = 28 ns.
Note 4: TRI-STATE note: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive this line with no contention.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.10 DMA MEMORY WRITE



TL/F/5282–31

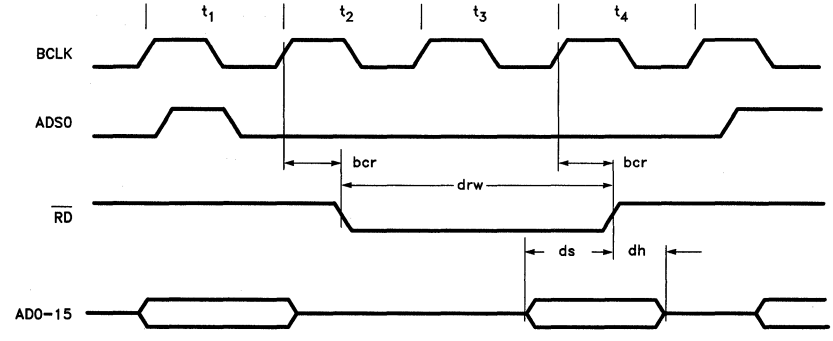
Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
bcw	Bus Clock to Write Strobe		50		60	ns
wds	Data Setup to $\overline{WR}$ High (Note 1)	2bcyc − 35		2bcyc − 45		ns
wdh	Data Hold from $\overline{WR}$ high (Note 1)	10	50	10	60	ns
bcwd	Data Valid from t2 Clock (Note 1)		75		90	ns
asds	Address Strobe to Data Strobe (Note 2)		bcl + 10		bcl + 20	ns
aswd	Address Strobe to Write Data Valid		bcl + 40		bcl + 60	ns

Conditions: DMA write, Local or Remote transfer, internal DMA.

Note 1: Data is enabled on AD0–15 only in local DMA transfers.

Note 2: Data strobe is either $\overline{RD}$ or $\overline{WR}$ out.

13.11 DMA MEMORY READ



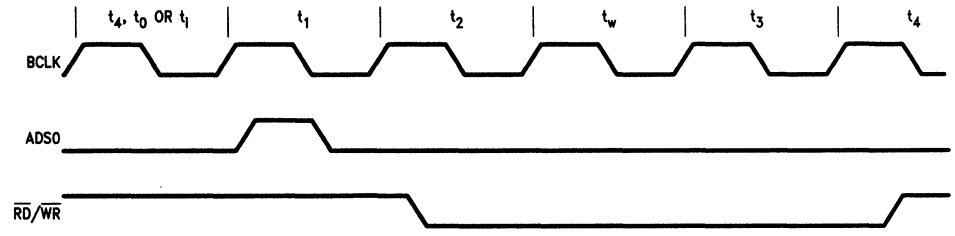
TL/F/5282–32

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
bcr	Bus Clock to Read Strobe		50		60	ns
ds	Data Setup to Read Strobe High	27		32		ns
dh	Data Hold from Read Strobe High	0		0		ns
drw	DMA Read Strobe Width Out	2bcyc − 10		2bcyc − 15		ns

Note 1: ds and dh timing are for Local transfers only.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.12 DMA WITH INTERNAL WAIT STATES



TL/F/5282–33

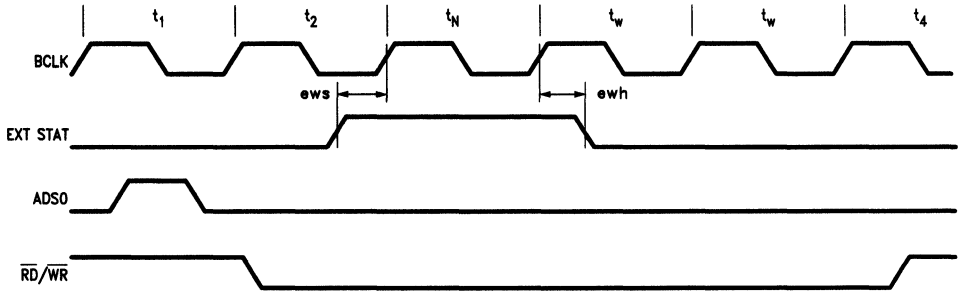
Conditions: Local or Remote DMA transfer, read or write, internal DMA.

Note 1: Addition of an internal wait state will lengthen RD/WR strobes by an additional bus clock cycle.

Note 2: Internal wait states are enabled by setting the Slow Read/Write bits in the Local and Remote Transfer registers.

Note 3: If used, external wait states will be added between cycles t3 and t4.

13.13 DMA WITH EXTERNAL WAIT STATES



TL/F/5282–34

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
ews	External Wait Setup to t3 Clock	15		20		ns
ewh	External Wait Hold after tw Clock	10		15		ns

Conditions: Read or write, internal DMA mode. Local or Remote transfer.

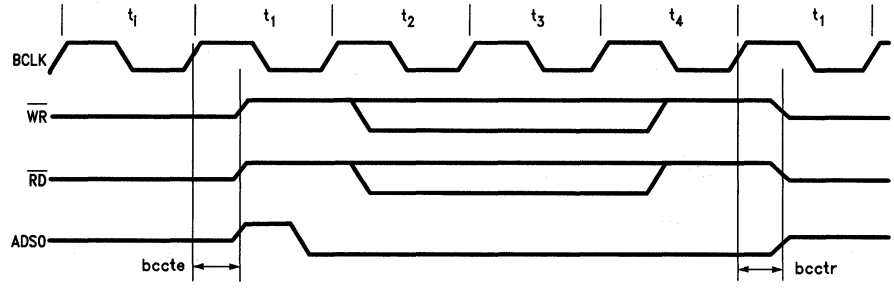
Note 1: Addition of external wait states will extend RD/WR strobes by an integral number of bus clock cycles.

Note 2: If enabled, an internal wait state is added between cycles t2 and t3.

Note 3: EXT STAT is sampled upon entering states t3 and tw, and adds wait states one bus clock cycle later.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.14 DMA CONTROL SIGNALS

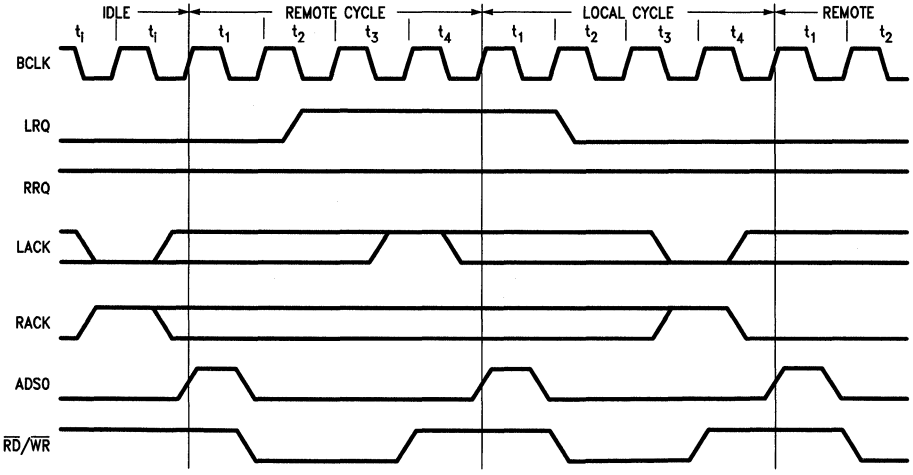


TL/F/5282–35

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
bccte	Bus Clock to Control Enable (WR, RD, ADS0)		55		70	ns
bcctr	Bus Clock to Control Release (WR, RD, ADS0) (Note 1)		60		70	ns

Note 1: TRI-STATE note: These limits include the RC delay inherent in our test method. These signals typically turn off within 15 ns, enabling other devices to drive this line with no contention.

13.15 LOCAL AND REMOTE DMA INTERLEAVING



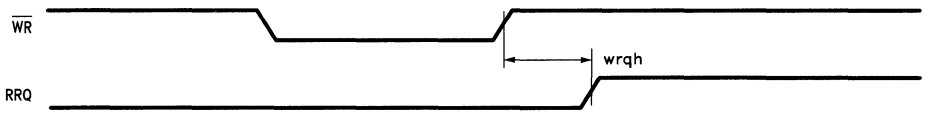
TL/F/5282–36

Note 1: Timing of the acknowledge pulses are used for illustration. Acknowledges need only to be set up with respect to t4 and t1 clock cycle.

Note 2: If both LACK and RACK are asserted with both LRQ and RRQ pending, a local DMA transfer will be performed.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.16 RRQ ASSERTION AFTER WRITING TO OC REGISTER FOR REMOTE TRANSFER

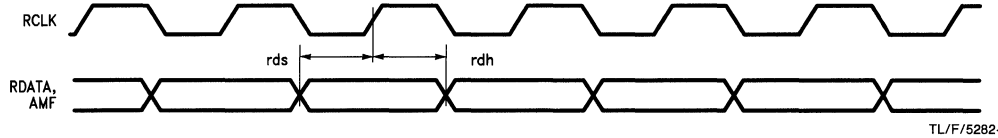


TL/F/5282–37

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
wrqh	Write Strobe to Remote Request High		100		150	ns

Conditions: Non-tracking mode, writing "Start Remote Input/Output" to the Operation Command register.

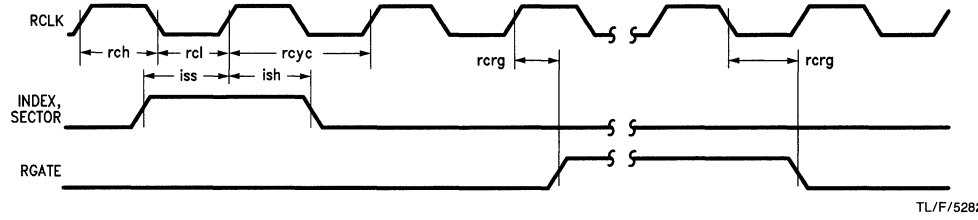
13.17 READ DATA TIMING



TL/F/5282–38

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rds	Read Data/AMF Setup to Read Clock	10		15		ns
rdh	Read Data/AMF Hold to Read Clock	10		15		ns

13.18 RGATE ASSERTION FROM INDEX OR SECTOR PULSE INPUT



TL/F/5282–39

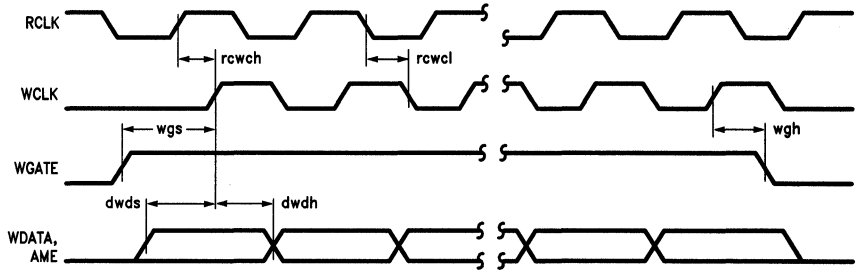
Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rcyc	Read Clock Cycle Time (Note 2)	40	10,000	80	10,000	ns
rch	Read Clock High Time (Note 2)	16	10,000	32	10,000	ns
rcl	Read Clock Low Time (Note 2)	16	10,000	32	10,000	ns
iss	Index/Sector Setup to Read Clock	10		15		ns
ish	Index/Sector Pulse Hold	10		15		ns
rcrg	Read Clock to Read Gate		45		60	ns

Note 1: INDEX/SECTOR low must meet iss/ish timing for proper INDEX/SECTOR pulse detection.

Note 2: For DP8466-20, minimum rcyc = 50 ns, minimum rch and rcl = 20 ns.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.19 WRITE DATA TIMING FOR NRZ TYPE DATA

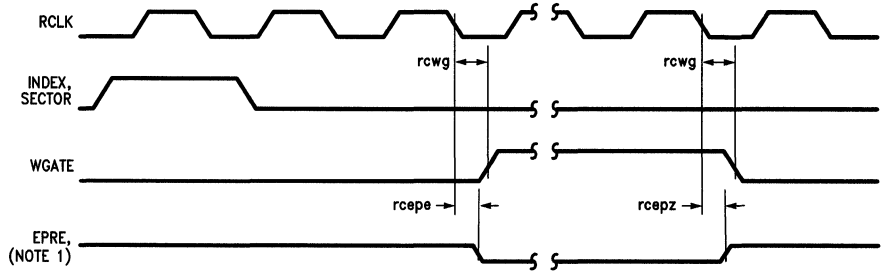


TL/F/5282–40

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rcwch	Read Clock to Write Clock High Delay		25		40	ns
rcwcl	Read Clock to Write Clock Low Delay		25		40	ns
rcwcs	Absolute Value of (rcwcl — rcwch)		6		7	ns
dwds	Drive Write Data Setup to Write Clock	rcl − 10		rcl − 15		ns
dwdh	Drive Write Data Hold to Write Clock	rch − 5		rch − 8		ns
wgs	Write Gate Setup to Write Clock	rcl − 10		rcl − 15		ns
wgh	Write Gate Hold to Write Clock	rch		rch		ns

Note 1: rcl and rch are described in Timing Diagram 13.18.

13.20 WGATE ASSERTION FROM INDEX OR SECTOR PULSE INPUT



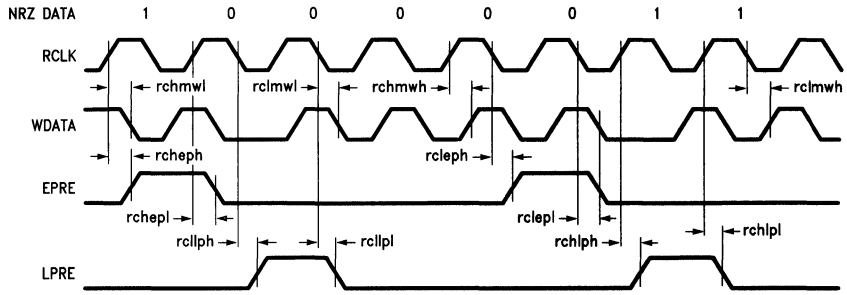
TL/F/5282–41

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rcwg	Read Clock to Write Gate		40		50	ns
rcepe	Read Clock to Early Precomp Enabled		50		60	ns
rcepz	Read Clock to Early Precomp TRI-STATE		50		60	ns

Note 1: Early Precompensation (EPRE) is used as an output only when writing MFM data.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.21 WRITE DATA TIMING FOR MFM TYPE DATA



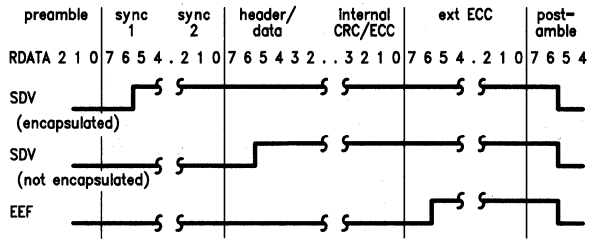
TL/F/5282–42

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rchmwh	RCLK High to MFM WDATA High		40		50	ns
rchmwl	RCLK High to MFM WDATA Low		40		50	ns
rclmwh	RCLK Low to MFM WDATA High		40		50	ns
rclmwl	RCLK Low to MFM WDATA Low		40		50	ns
rcheph	RCLK High to EPRE High		40		50	ns
rchepl	RCLK High to EPRE Low		40		50	ns
rcleph	RCLK Low to EPRE High		40		50	ns
rclepl	RCLK Low to EPRE Low		40		50	ns
rchlph	RCLK High to LPRE High		40		50	ns
rchlpl	RCLK High to LPRE Low		40		50	ns
rcllph	RCLK Low to LPRE High		40		50	ns
rcllpl	RCLK Low to LPRE Low		40		50	ns

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.22 POSITIONAL TIMING FOR SDV AND EEF

Read operation (Compare Header, Read Header, Compare Data or Read Data)

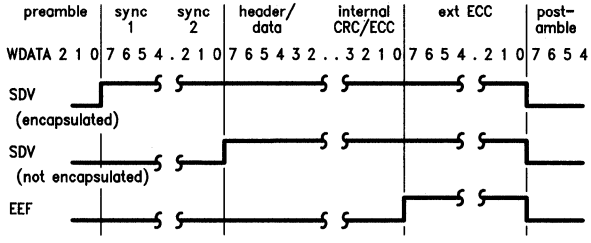


TL/F/5282–43

Note 1: Data should be delayed 2 bit times before entering external ECC circuitry in order for it to properly align correctly with SDC and EEF.

Note 2: Encapsulation is controlled by the HEN and DEN bits in the EC register, and causes the sync patterns to be included in the CRC/ECC calculation.

Write operation (Write Header, Write Data or Format Track)

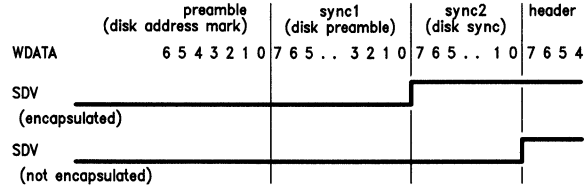


TL/F/5282–44

Note 1: Write operation shown is for NRZ data. For MFM encoding, Write data is delayed two bit times relative to NRZ data.

Note 2: Encapsulation is controlled by the HEN and DEN bits in the EC register, and causes the sync patterns to be included in CRC/ECC calculation.

Write header operation (Start with Address Mark)



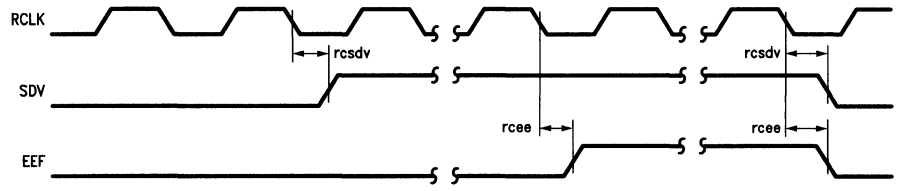
TL/F/5282–45

Note 1: Field names within parenthesis are the names of the fields on disk.

Note 2: Encapsulation is controlled by the HEN bit in the EC register, and causes the sync patterns to be included in CRC/ECC calculation.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.23 FIELD ENVELOPE TIMING



TL/F/5282–46

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
rcsdv	Read Clock to Serial Data Valid		35		50	ns
rcee	Read Clock to External ECC		35		50	ns

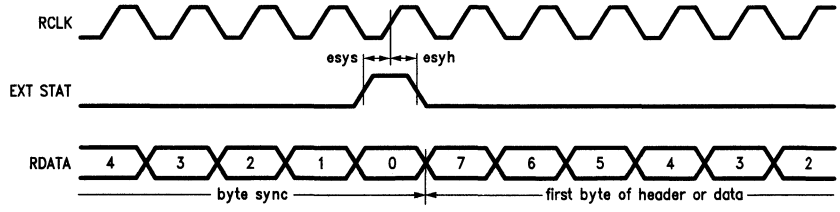
Note 1: SDV is asserted after sync fields, and is deasserted at the start of the postamble field. If sync field encapsulation is enabled, SDV is asserted at the start of the sync fields.

Note 2: EEF is asserted at the start of the external ECC field, and is deasserted at the start of the postamble field.

Note 3: When the DDC is receiving data from the disk, the SDV and EEF are delayed by two bit times from incoming read data due to internal delays.

Note 4: If the external ECC count is set to zero, no EEF output will be generated.

13.24 EXTERNAL STATUS TIMING WHEN USING EXTERNAL BYTE SYNC



TL/F/5282–47

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
esys	External Byte Sync Setup to Rising Edge of Bit Clock 0 of Byte Sync	15		20		ns
esyh	External Byte Sync Hold to Rising Edge of Bit Clock 0 of Byte Sync	10		15		ns

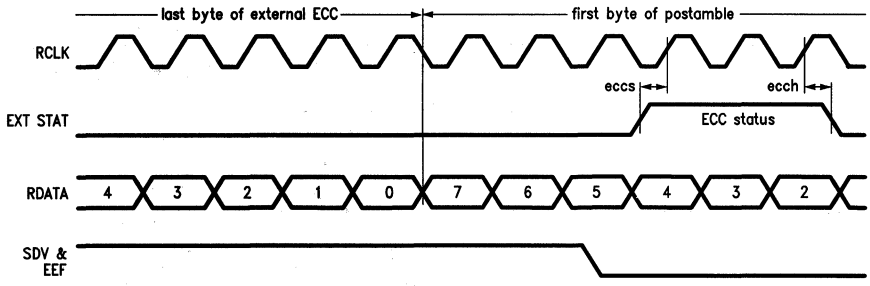
Note 1: The external sync feature can only be used if the Enable External Wait states (EEW) bit of the Remote Transfer (RT) register is not set.

Note 2: External circuitry is needed to feed the DDC with NRZ zeros until the external sync signal has been generated to prevent the DDC from trying to detect sync.

Note 3: If External Sync and External Wait states are not being used, the EXT. STAT. pin must be false during preamble and sync fields.

13.0 AC Electrical Characteristics & Timing Diagrams (Continued)

13.25 EXTERNAL STATUS TIMING WHEN USED FOR EXTERNAL ECC



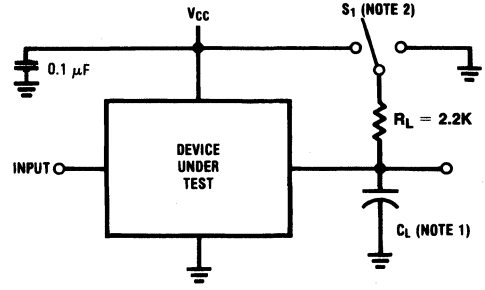
TL/F/5282–48

Symbol	Parameter	DP8466-25/20		DP8466-12		Units
		Min	Max	Min	Max	
eccs	External ECC Status Setup to Rising Edge of Bit Clock 4 of Postamble	15		20		ns
ecch	External ECC Status Hold to Rising Edge of Bit Clock 2 of Postamble	10		5		ns

Note 1: The external ECC error detection feature can only be used if the Enable External Wait states (EEW) bit of the Remote Transfer register (RT) is zero.

14.0 AC Timing Test Conditions

Input Pulse Levels	GND to 3.0V
Input Rise and Fall Times	6 ns
Input and Output Reference Levels	1.3V
TRI-STATE Reference Levels	Float (ΔV) ± 0.5V
Output Load (See Figure 27)	



TL/F/5282–77

FIGURE 27

Note 1: C_L = 50 pF, includes scope and jig capacitance

Note 2: S1 = Open for Push Pull Outputs

S1 = V_{CC} for High Impedance to active low and active low to High Impedance measurements.

S1 = GND for High Impedance to active high and active high to High Impedance measurements.

Capacitance (T_A = 25°C, f = 1MHz)

Parameter	Description	Typ	Max	Unit
C_{IN}	Input Capacitance	7	12	pF
C_{OUT}	Output Capacitance	7	12	pF

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

15.0 Miscellaneous Timing Information

15.1 STATUS REGISTER TIMING

HEADER FAULT: This bit is set at the start of the Header Postamble field of a header with a CRC/ECC error. It is reset at the start of the Header Postamble of the header requested, or upon receipt of a new disk command. No interrupt is generated.

NEXT DISK COMMAND: This bit is set at the start of the Header Postamble of the last sector of an operation, and is reset upon loading the Drive Command register. No interrupt is generated.

HEADER MATCH COMPLETED: This bit is set at the start of the Header Postamble field of the header of interest. This bit is reset when the DDC begins the next header operation. An interrupt is generated if enabled.

LOCAL REQUEST: This bit has the same timing as the Local Request pin. When the FIFO requires servicing, this bit is set. When service is no longer required, this bit is cleared. No interrupt is generated.

REMOTE COMMAND BUSY: In the tracking mode, this bit is set 3–5 RCLK's after receipt of a drive command. In the non-tracking mode, this bit is set when either a Start Remote Input or Start Remote Output command is received in the Operation Command register. This bit is reset and interrupt is generated upon completion of the initiating operation.

LOCAL COMMAND BUSY: This bit is set 3–5 RCLK's after receipt of a drive command which requires the use of the local channel. It is reset after the last transfer of the local channel if in the non-tracking mode or writing the disk, or after the last transfer of the remote channel if in the tracking mode and reading disk. Interrupt is generated upon completion of the initiating operation.

CORRECTION CYCLE ACTIVE: This bit is set upon receipt of the Start Correction Cycle in the Operation Command register, and is reset at the end of the correction operation. An interrupt is generated at the end of the correction cycle.

ERROR DETECTED: This bit is a logical OR function of all the bits in the Error register. An interrupt is generated when an error is detected.

15.2 ERROR REGISTER TIMING

HFASM ERROR: If while in the HFASM mode the sector address matches and another header byte does not, this bit will be set at the start of the Header Postamble field.

DATA FIELD ERROR: If the Data field contains a CRC/ECC error, this bit will be set at the start of the Data Postamble field.

SECTOR NOT FOUND: If the header of the desired sector is not located before two index pulses are received, this bit will be set upon receipt of the second index pulse.

SECTOR OVERRUN: If an index or sector pulse is detected while reading the Header or Data field, or while writing and not in the Gap field, this bit will be set upon receipt of the sector/index pulse.

NO DATA SYNC: If an index or sector pulse is received before data sync is detected, this bit is set upon receipt of the sector/index pulse. If there is a data sync error after the first sync byte has been detected, this bit will be set during the byte following the byte in error.

FIFO DATA LOST: If a transfer between the disk and FIFO causes the FIFO to underrun or overrun, this bit will be set within the next byte time creating a write splice if write gate was on. This is reflected as an ECC error and can be removed if sector is rewritten.

CORRECTION FAILED: This bit is set at the end of the correction cycle if the error is non-correctable.

LATE INTERLOCK: This bit is set at the start of Data Postamble field for Read operations and at the end of the postamble field for non-format Write operations. While formatting, this bit is set at the end of the Gap field.

15.3 GENERAL TIMING FOR READ GATE

Whenever the DDC is reading, comparing, or in some cases, ignoring information, RGATE is asserted. The use of RGATE can be separated into three groups: Header search (soft sectored mode), header examination, and data examination.

SEARCHING FOR HEADERS

When the DDC is searching for a header in the soft-sectored mode, RGATE is asserted in a somewhat random location in the format. After being asserted, if the DDC does not recognize the address mark pattern within eight bit times of detecting a one, RGATE will be de-asserted in 18½ RCLK's. RGATE will then remain low for 17½ RCLK's before another search attempt is made.

In modes where the DDC starts a Read, Compare or Ignore Header operation at an index or sector pulse, RGATE will be asserted 3–4 RCLK cycles from detection of the index or sector pulse.

DATA OPERATIONS

After the header operation has completed, RGATE will be removed two bits after the start of the Header Postamble. If a Read or Check Data operation is to follow, RGATE will be reasserted 11½ bits after the Header Postamble.

At the end of the Data field, RGATE will be removed two bits into the start of the Data Postamble.

15.4 WRITE GATE TIMING

Whenever the DDC is writing information, WGATE is asserted. WGATE can be separated into three uses: Writing header, writing data or track formatting.

WRITING HEADERS

When the DDC writes the header, the write operation does not begin until the receipt of an index or sector pulse. After the pulse is detected, WGATE will be asserted 2½–3½ RCLKs from the detection of the pulse. WGATE will stay true until the end of the Header Postamble, unless the Data field is to be written. If the Data field is to be written, WGATE will not be de-asserted between the Header and Data fields.

WRITING DATA

After a header operation has properly completed, WGATE will be asserted 3 bit times into the Data Preamble. The WGATE will remain active until the end of the Data Postamble. Because of internal delays within the DDC, the Write Data operation is delayed three bit times from the header patterns.

15.0 Miscellaneous Timing Information (Continued)

FORMAT TRACK

In a format track operation, WGATE is asserted 2½–3½ RCLK's from the detection of the index pulse. WGATE will remain active until the next index pulse is detected, and will then be removed.

Note: Detection of an index or sector pulse is defined as the rising edge of the RCLK where index/sector input has met the setup time.

15.5 NORMAL INTERRUPTS

Interrupts are generated by the DDC for a variety of reasons, but they all fall into one of three categories: Either they signal normal completion, a synchronization point, or an error condition. If an interrupt is generated because of an error, the interrupt will have timing as described in the Error register timing section.

The Header Operation Complete interrupt is used for synchronization, and is enabled with the Enable Header Interrupt bit of the Operation Command register. This interrupt will occur when the DDC finishes the header operation, and starts the data operation. For Read, Compare, Write, or Ignore Header operations, the interrupt will be generated at the start of the Header Postamble field.

The normal Operation Complete interrupt is dependent on the operation being performed. If the operation is to Check Data, the interrupt is generated at the start of the Data Postamble field. For Write Data operations, an interrupt will be generated at the end of the Data Postamble. When the DDC is formatting, the interrupt will be delayed by the length of the Header Preamble after the format has finished. The fourth event is further defined by the DMA mode used. For all local channel operations except for tracking mode disk read, the interrupt will be generated during the last transfer of data from the FIFO. In the configuration, tracking mode disk read, the interrupt will be delayed until the last transfer is made by the remote DMA. For all non-tracking remote DMA operations, the interrupt will be generated during the last transfer of the remote DMA.

When a correction operation is being performed, an interrupt is generated at the end of the correction cycle, regardless of the outcome.

15.6 DERATING FACTOR

Output timings are measured with a purely capacitive load for 50 pF. The following correction factor can be used for other loads:

DP8466-25/20 $C_L \geq 50$ pF: +.13 ns/pF (ADS0, ADS1)
+.20 ns/pF (all other outputs)

DP8466-12 $C_L \geq 50$ pF: +.18 ns/pF (ADS0, ADS1)
+.25 ns/pF (all other outputs)

16.0 Appendix

16.1 DDC REGISTERS, INDEX BY HEX ADDRESS

The following is a repeat of what can be found in the DDC INTERNAL REGISTERS Section. This listing is arranged numerically by hex address, and is provided as a quick reference. The section numbers provided indicate where the best description for the particular register can be located. For an explanation of the information contained in the WR and RD columns, refer to the key in the INTERNAL REGISTERS Section.

COLUMN KEY:

HA: *Hex Address* **#B**: *Number of bits* **WR**: *Write* **RD**: *Read* **SC**: *Section*

HA	REGISTER	#B	WR	RD	SC
00	Status Register (S)	8	NO	R	3.1
01	Error Register (E)	8	NO	R	3.1
02	ECC SR Out 0	8	NO	R	3.4
02	Polynomial Preset Byte 0 (PPB0)	8	D	NO	3.4
03	ECC SR Out 1	8	NO	R	3.4
03	Polynomial Preset Byte 1 (PPB1)	8	D	NO	3.4
04	ECC SR Out 2	8	NO	R	3.4
04	Polynomial Preset Byte 2 (PPB2)	8	D	NO	3.4
05	ECC SR Out 3	8	NO	R	3.4
05	Polynomial Preset Byte 3 (PPB3)	8	D	NO	3.4
06	ECC SR Out 4	8	NO	R	3.4
06	Polynomial Preset Byte 4 (PPB4)	8	D	NO	3.4
07	ECC SR Out 5	8	NO	R	3.4
07	Polynomial Preset Byte 5 (PPB5)	8	D	NO	3.4
08	Data Byte Count (0)	8	NO	R	3.4
08	Polynomial Tap Byte 0 (PTB0)	8	D	NO	3.4
09	Data Byte Count (1)	8	NO	R	3.4
09	Polynomial Tap Byte 1 (PTB1)	8	D	NO	3.4
0A	Polynomial Tap Byte 2 (PTB2)	8	D	NO	3.4
0B	Polynomial Tap Byte 3 (PTB3)	8	D	NO	3.4
0C	Polynomial Tap Byte 4 (PTB4)	8	D	NO	3.4
0D	Polynomial Tap Byte 5 (PTB5)	8	D	NO	3.4
0E	ECC CONTROL (EC)	8	D	NO	3.4
0F	Header Byte Count (HBC)/Interlock	3	F	R	3.1
10	Drive Command Register (DC)	8	C	NO	3.1
11	Operation Command Register (OC)	8	C	NO	3.1
12	Sector Counter (SC)	8	C	R	3.1
13	Number of Sector Operations Counter (NSO)	8	C	R	3.1
14	Header Byte 0 Pattern	8	C	R	3.3
15	Header Byte 1 Pattern	8	C	R	3.3
16	Header Byte 2 Pattern	8	C	R	3.3
17	Header Byte 3 Pattern	8	C	R	3.3
18	Header Byte 4 Pattern	8	C	R	3.3
19	Header Byte 5 Pattern	8	C	R	3.3
1A	Remote Data Byte Byte Count (L)	8	C	R	3.2
1B	Remote Data Byte Byte Count (H)	8	C	R	3.2
1C	DMA Address Byte 0	8	C	R	3.2

HA	REGISTER	#B	WR	RD	SC
1D	DMA Address Byte 1	8	C	R	3.2
1E	DMA Address Byte 2	8	C	R	3.2
1F	DMA Address Byte 3	8	C	R	3.2
20	Data Postamble Byte Count	5	D	R	3.3
21	ID Preamble Byte Count	5	C	R	3.3
22	ID Sync #1 (AM) Byte Count	5	D	R	3.3
23	ID Sync #Byte 2 Count	5	D	R	3.3
24	Header Byte 0 Control	5	D	R	3.3
25	Header Byte 1 Control	5	D	R	3.3
26	Header Byte 2 Control	5	D	R	3.3
27	Header Byte 3 Control	5	D	R	3.3
28	Header Byte 4 Control	5	D	R	3.3
29	Header Byte 5 Control	5	D	R	3.3
2A	Data External ECC Byte Count	5	D	R	3.3
2B	ID External ECC Byte Count	5	D	R	3.3
2C	ID Postamble Byte Count	5	D	R	3.3
2D	Data Preamble Byte Count	5	D	R	3.3
2E	Data Sync #1 (AM) Byte Count	5	D	R	3.3
2F	Data Sync #2 Byte Count	5	D	R	3.3
30	Data Postamble Pattern	8	D	R	3.3
31	ID Preamble Pattern	8	D	R	3.3
32	ID Sync #1 (AM) Pattern	8	D	R	3.3
33	ID Sync #2 Pattern	8	D	R	3.3
34	Gap Byte Count	8	F	R	3.3
35	Disk Format Register (DF)	8	D	NO	3.1
36	Header Diagnostic Readback (HDR)	8	NO	R	3.1
36	Local Transfer Register	8	I	NO	3.2
37	DMA Sector Counter (DSC)	8	NO	R	3.2
37	Remote Transfer Register	8	I	NO	3.2
38	Sector Byte Count 0	8	D	R	3.2
39	Sector Byte Count 1	8	D	R	3.2
3A	Gap Pattern	8	F	R	3.3
3B	Data Format Pattern	8	F	R	3.3
3C	ID Postamble Pattern	8	D	R	3.3
3D	Data Preamble Pattern	8	D	R	3.3
3E	Data Sync #2 (AM) Pattern	8	D	R	3.3
3F	Data Sync #2 Pattern	8	D	R	3.3

16.0 Appendix (Continued)

16.2 ALPHABETICAL MNEMONIC GLOSSARY AND INDEX

Listed on the following pages are the majority of the abbreviations used within this data sheet as mnemonics to describe portions or functions of the DDC. The section numbers referenced indicate where the terms are first defined. Mnemonics from the specifications section are not included here.

MNEMONIC	DESCRIPTION	SECTION
AD0–7	Address/Data 0–7 (pins 41–48)	2.0
AD8–15	Address/Data 8–15 (pins 1–8)	2.0
ADS0	Address Strobe 0 (pin 9)	2.0
ADS1	Address Strobe 1 (attached to RRQ, pin 37)	2.0
AME	Address Mark Enable (attached to LPRE, pin 13)	2.0
AMF	Address Mark Found (attached to EPRE, pin 16)	2.0
BCLK	Bus Clock (pin 40)	2.0
CCA	Correction Cycle Active (bit in Status register)	3.1
CF	Correction Failed (bit in Error register)	3.1
$\overline{CS}$	Chip Select (pin 28)	2.0
CS0–3	Correction Span Selection (bits in EC register)	3.4
DC	Drive Command register	3.1
DNE	Data Non-Encapsulation (bit in EC register)	3.4
DF	Disk Format register	3.2
DFE	Data Field Error (bit in Error register)	3.1
D01, 2	Data Operation bits (command in DC register)	3.1
DSC	DMA Sector Counter	3.2
E	Error register	3.1
EC	ECC Control register	3.4
ED	Error Detected (bit in Status register)	3.1
EEF	External ECC Field (pin 26)	2.0
EEW	Enable External Wait (bit in RT register)	3.2
EHF	Enable HFASM Function (bit in HC0–5 registers)	3.3
EHI	Enable Header Interrupts (command in OC register)	3.1
FTF	FIFO Table Format (bit in DF register)	3.1
HBA	Header Byte Active (bit in HC0–5 registers)	3.3
HBC	Header Byte Count register	3.1
HC0–5	Header Byte 0–5 Control registers	3.3
HDR	Header Diagnostic Readback register	3.1
HNE	Header Non-Encapsulation (bit in EC register)	3.4
HF	Header Fault (bit in Status register)	3.1
HFASM	Header Failed Although Sector number Matched (bit in Error register)	2.0
HMC	Header Match Completed (bit in Status register)	3.1
H01, 2	Header Operation bits (command in DC register)	3.1
HSS	Hard or Soft Sectored (bit in DF register)	3.1
ID1, 2	Internal Data Appendage (bits in DF register)	3.1
IDI	Invert Data In (bit in EC register)	3.4
IH1, 2	Internal Header Appendage (bits in DF register)	3.1
$\overline{INT}$	Interrupt (pin 29)	2.0
LA	Long Address (bit in LT register)	3.2
LACK	Local DMA Acknowledge (pin 39)	2.0
LBL1, 2	Local Burst Length (bits in LT register)	3.2
LCB	Local Command Busy (bit in Status register)	3.1
LI	Late Interlock (bit in Error register)	3.1
LPRE	Late Precompensation (attached to AME, pin 13)	2.0
LRQ	Local DMA Request (pin 36)	2.0
LRQ	Local Request (bit in Status register)	3.1
LSRW	Local Slow Read/Write (bit in LT register)	3.2
LT	Local Transfer register	3.2
LTEB	Local Transfer Exact Burst (bit in LT register)	3.2
LWDT	Local Word Data Transfer (bit in LT register)	3.2
MFM	MFM Encode (bit in DF register)	3.1
MSO	Multi-Sector Operation (command in DC register)	3.1
NCP	Not Compare (bit in HC0–5 registers)	3.3
NDC	Next Disk Command (bit in Status register)	3.1
NDS	No Data Synch (bit in Error register)	3.1
NSO	Number of Sector Operations counter	3.1
OC	Operation Command register	3.1
PPB0–5	Polynomial Preset Byte 0–5	3.4
PTB0–5	Polynomial Tap Byte 0–5	3.4
RACK	Remote DMA Acknowledge (pin 38)	2.0
RBL1, 2	Remote Burst Length (bits in RT register)	3.2
RBO	Reverse Byte Order (bit in LT register)	3.2
RCB	Remote Command Busy (bit in Status register)	3.1
RCLK	Read Clock (pin 25)	2.0
$\overline{RD}$	Read (pin 11)	2.0
RDATA	Read Data (pin 15)	2.0
RED	Re-Enable DDC (command in DC register)	3.1
RES	Reset DDC (bit OC register)	3.2
RGATE	Read Gate (pin 19)	2.0
RRQ	Remote Request (attached to ADS1, pin 37)	2.0
RS0–5	Register Select 0–5 (pins 30–35)	2.0
RSRW	Remote Slow Read/Write (bit in RT register)	3.2
RT	Remote Transfer register	3.2
RTEB	Remote Transfer Exact Burst (bit in RT register)	3.2
RWDT	Remote Word Data Transfer (bit in RT register)	3.2
S	Status register	3.1
SAIS	Start At Index or Sector (command in DC register)	3.1
SAM	Start at Address Mark (bit in DF register)	3.1
SC	Sector Counter	3.1
SCC	Start Correction Cycle (command in OC register)	3.1
SDV	Serial Data Valid (pin 27)	2.0
SLD	Select Local DMA (bit in LT register)	3.2
SNF	Sector Not Found (bit in Error register)	3.1
SO	Sector Overrun (bit in Error register)	3.1
SRD	Select Remote DMA (bit in RT register)	3.2
SRI	Start Remote Input (command in OC register)	3.1
SRO	Start Remote Output (command in OC register)	3.1
SSC	Substitute Sector Counter (bit in HC0–5 registers)	3.3
TM	Tracking Mode (bit in RT register)	3.2
WCLK	Write Clock (pin 21)	2.0
WDATA	Write Data (pin 18)	2.0
WGATE	Write Gate (pin 20)	2.0
$\overline{WR}$	Write (pin 10)	2.0

National Semiconductor Corporation

PRELIMINARY

DP8470 Floppy Disk Support Chip Data Separator & Write Precompensation

General Description

This part is a general purpose data separator which can be used to generate a read clock for FM or MFM encoded data. This read clock can be used with many existing floppy disk controllers including the μPD765A, 8272A, and WD179x. It can also be used with National Semiconductor's Hard Disk Controller, DP8466, for a combination hard disk/floppy disk system. The data separator can be used for data rates ranging from 125 kbits/sec up to 1.25 Mbits/sec.

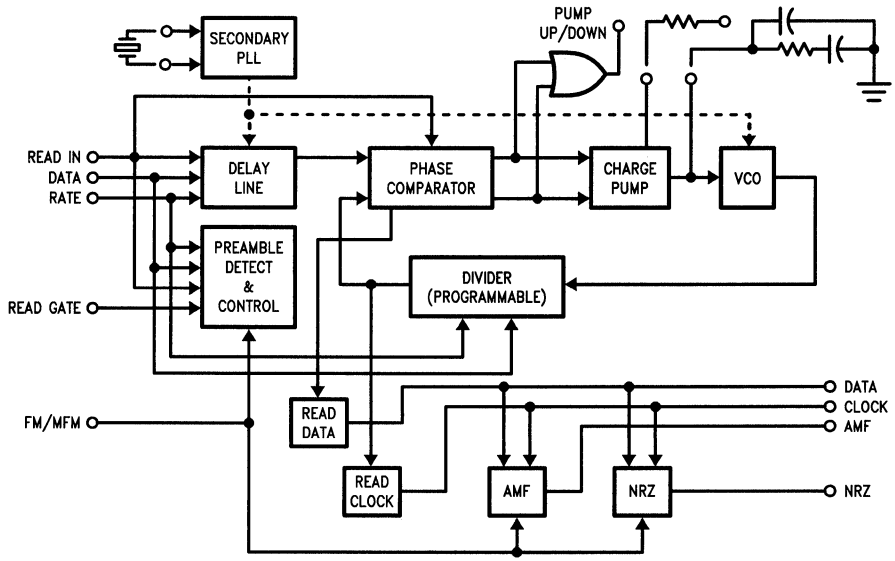
This part also contains a write precompensation circuit. Normally a disk controller will determine whether a bit of data needs to be shifted early, late, or not at all. The controller does not do the actual shifting however. This disk support chip will do the actual shifting that is requested by the controller.

A few other miscellaneous circuits have been included to enable this part to be interfaced to a hard disk controller such as the DP8466. The hard disk controller requires that the data read off from the disk be converted to an NRZ format rather than MFM encoded. Also, the controller needs to know when a valid address mark has been read from the data stream. This disk support chip does both of these functions.

Features

- Analog dual-gain PLL data separator
- Write precompensation (0–393 ns)
- Requires no external trimmable components
- Supports FM/MFM 125 kbits–1.25 Mbits/sec
- Interface to all popular floppy disk controllers.
- Interface to DP8466 hard disk controller
  - — Address Mark Found output
  - — NRZ output
- Pump up/down output for testing
- Low power CMOS
- 24-pin narrow package or 28-pin PCC

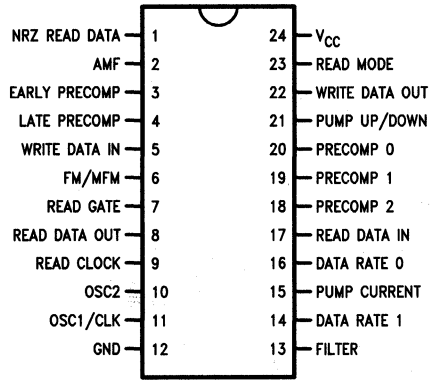
Block Diagram



TL/F/8593–1

Connection Diagram

Dual-In-Line Package

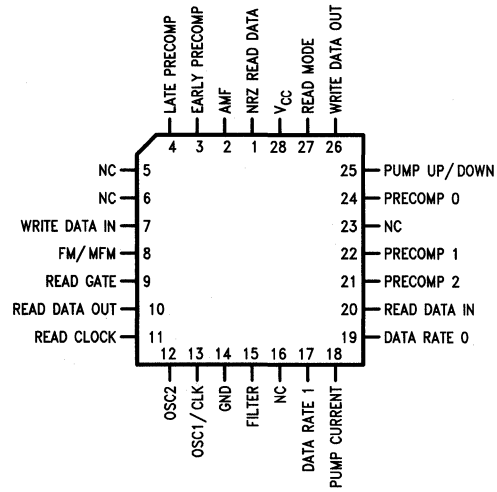


TL/F/8593-2

Top View

Order Number DP8470N or DP8470J
See NS Package Number N24C or J24F

Plastic Chip Carrier



TL/F/8593-9

Top View

Order Number DP8470V
See NS Package Number V28A

Note: Make no corrections to NC pins (No connection).

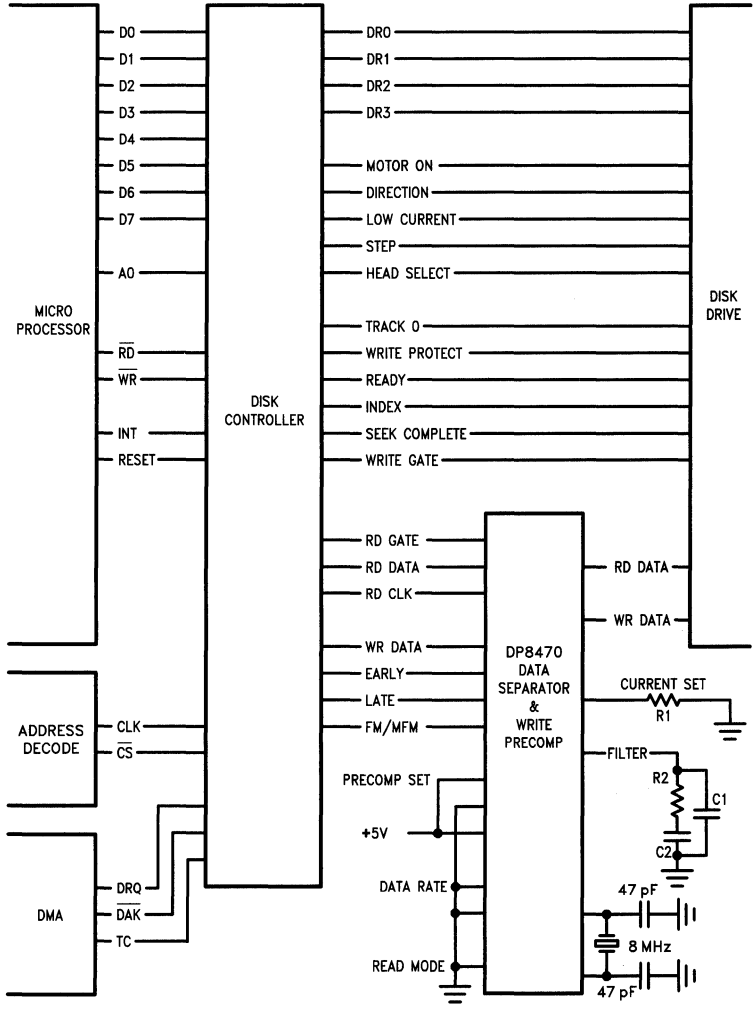
Pin Descriptions

Symbol	DIP Pin No.	PCC Pin No.	Function
NRZ READ DATA	1	1	This output will present data read from the disk in NRZ format based on the read clock. This output is set to a high impedance state when Read Gate is not asserted.
AMF	2	2	This output could be used with any controller that needs an indication of a valid address mark. This pin goes high for one bit period when an address mark is detected. This output is set to a high impedance state when Read Gate is not asserted.
EARLY/PRECOMP LATE/PRECOMP	3 4	3 4	These two active-high inputs determine whether the incoming write-data pulse should be shifted early or late in time.
WRITE DATA IN	5	7	Active-high data input from the disk controller.
FM/MFM	6	8	0 = FM, 1 = MFM. This pin also affects the data rate.
READ GATE	7	9	While this input is low, the PLL will lock to its center frequency. While this input is high, the PLL will lock to signal on the Read Data In pin.
READ DATA OUT	8	10	Active-high data output to the floppy controller. The Read Data Out is synchronized to the Read Clock Output.
READ CLOCK	9	11	This is a clock output that has the same frequency as the data rate. This output is always derived from the output of the VCO. While reading data, the VCO is tracking the data rate. When not reading data, the VCO is locked to its reference frequency.
OSC 1,2	10,11	12,13	These two pins enable the connection of a crystal to form the reference oscillator. Optionally an external clock can be used instead. The clock would drive Osc 1 while Osc 2 would be left open.
FILTER	13	15	This pin is the output of the dual-gain charge pump and is also the input to the VCO. A simple filter is attached to this pin.
DATA RATE 0 DATA RATE 1	16 14	19 17	These two pins select the data rate that this chip will sync to: 00 = 125FM/250MFM 01 = 250FM/500MFM 10 = 500FM/1000MFM 11 = Test Mode
PUMP CURRENT	15	18	A resistor is attached to this pin to set the charge pump current.

Pin Descriptions (Continued)

Symbol	DIP Pin No.	PCC Pin No.	Function
READ DATA IN	17	20	Active-high data input from the disk drive.
PRECOMP 0 PRECOMP 1 PRECOMP 2	20 19 18	24 22 21	These three input pins select the amount of write precompensation.
PUMP UP/DOWN	21	25	This active-high output is the logical OR of Pump Up and Pump Down. This is used for diagnostic purposes.
WRITE DATA OUT	22	26	Active-high data output to the floppy drive. This is the same data as is input on the Write Data In pin, except it has been write-precompensated and delayed.
READ MODE	23	27	This input determines what read algorithm is used to select between the low and the high gain mode. (Low = 4-state algorithm, high = 2-state algorithm.)
V_{CC} GROUND	24 12	28 14	These pins are the power supply pins for both the digital circuitry and the analog circuitry.

Typical Floppy Disk Drive Application



TL/F/8593–3

Functional Description

The data separator consists of a dual gain analog PLL (Phase Locked Loop). This PLL synchronizes a VCO (Voltage Controlled Oscillator) to the raw data signal read from a disk drive. The Read Clock pin is derived from the VCO. The Read Data Out pin mirrors the Read Data In pin except that it is centered with respect to the Read Clock. In addition, NRZ encoded data is available at the Read Clock. In addition, NRZ encoded data is available at the NRZ Read Data pin.

The PLL consists of three main components, a phase comparator, a filter, and a voltage controlled oscillator (VCO), as shown in the Block Diagram. The basic operation of a PLL is fairly straightforward. The phase comparator detects the difference between the phase of the VCO output and the phase of the raw data being read from the disk. This phase difference is converted to a current which either charges or discharges a filter. The resulting voltage of the filter changes the frequency of the VCO in an attempt to reduce the phase difference between the two signals. A PLL is "locked" when the frequency of the VCO is exactly the same as the average frequency of the read data. This is somewhat of a simplified view because it ignores such topics as loop stability, acquisition time, and filter values.

The external filter simply consists of two capacitors and a resistor as shown in the typical application diagram. Another resistor is used to set the charge pump current.

The quarter period delay line is used to determine the center of a bit cell. It is important that this delay line be as accurate as possible. A typical data separator would normally require an external trim to adjust the delay. An external trim is not required for the DP8472/74 however. A secondary PLL is used to automatically calibrate the delay line. The secondary PLL also calibrates the center frequency of the VCO.

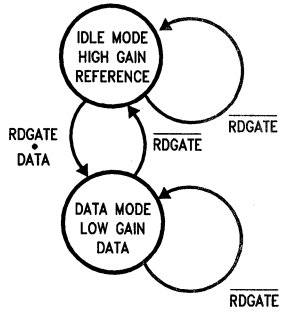
The Preamble Detect circuit is used with the four-state algorithm to determine when the PLL switches from the high gain mode to its low gain mode. This circuit scans the incoming data for the frequency corresponding to a preamble plus or minus 15 percent.

Circuit Operation

READ MODE

There are two read modes to choose from. The Read Mode is selected with the Read Mode pin. The state of this pin should not change during an actual read operation.

Two-State Diagram



Data = first incoming pulse received. TL/F/8593–4

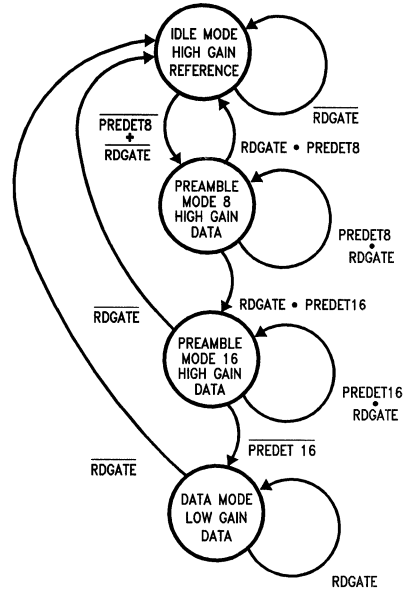
MODE ONE (TWO-STATE DIAGRAM)

When Read Gate is not asserted, the PLL is locked to the crystal frequency in its high-gain mode with a phase/frequency comparator. When Read Gate is asserted, the PLL will remain locked to the crystal until the first data bit arrives.

It will then lock to the incoming data in its low-gain mode with a phase-only comparator. It will stay in this mode until Read Gate is deasserted.

The NRZ Data Output will remain low until 8 bits have been read. This is to guarantee that the clock pulses from the preamble have become stable. The Read Data Out is enabled as soon as the first data bit arrives.

Four-State Diagram



PreDet8 = 8 consecutive bits of preamble frequency. TL/F/8593–5
PreDet16 = 16 consecutive bits of preamble frequency.

MODE TWO (FOUR-STATE DIAGRAM)

When Read Gate is not asserted, the PLL is locked to the crystal frequency in its high-gain mode with a phase/frequency comparator. When Read Gate is asserted, a preamble-detect circuit is enabled. This circuit looks for consecutive bits of the correct preamble frequency. The PLL will stay locked to the crystal frequency until 8 consecutive preamble bits are read. At this point the PLL will lock on to the incoming data (preamble) in its high-gain mode with a phase-only comparator. When the preamble-detect circuit finds 16 consecutive bits of the preamble frequency, the Data Output and the NRZ Data Output logic will be enabled. If at any time before the 16 bits are counted the preamble-detect circuit goes false, the PLL will return to the Idle Mode locked to the crystal. As soon as the preamble-detect circuit goes false after the 16 bits are counted (such as when beginning of the address mark is read) the PLL will switch to its low-gain mode. It will stay in this mode until Read Gate is deasserted. The timing is such that the comparison of the first bit of the Address Mark is done in the low-gain mode.

Between the time in which Read Gate is asserted and the Read Data Output is enabled (states 1 and 2 of the 4-state diagram), the data pattern 4E(hex) or FF(hex) will be output

Circuit Operation (Continued)

for MFM and FM respectively on the Read Data Output pin. The NRZ Data Output will remain low during this time.

WRITE MODE

When writing data, the rising edge of the signal presented to the Write Data Input is delayed before it appears on the Write Data Output. The number of delays is shown in Table I.

The actual value of the delay is determined by the PreComp Set pins. There is also a base delay as specified in the AC timing characteristics.

TABLE I

Early	Late	# of Delays
0	0	1
0	1	2
1	0	0
1	1	illegal

Design Considerations

The operating characteristics of this part are totally pin programmable. The designer needs to set three parameters by tying pins either high or low. These three parameters are the data rate, the amount of write precompensation, and the read mode algorithm.

DATA RATE, FM/MFM

The data rate is determined by three pins (Data Rate 0, Data Rate 1, FM/MFM) and also the clock frequency. The normal clock frequency is 8 MHz. The selectable data rates based on an 8 MHz clock are shown in Table II. If a data rate is needed that is not shown in the 8 MHz column, it may be produced by varying the clock frequency. See the AC Electrical Characteristics for the acceptable range of clock frequencies.

If either of these parameters (data rate or FM/MFM) are subject to change then these pins could be connected to switches, an output port, or through some logic from the controller's drive select output.

TABLE II

Data 1	Rate 0	FM/ MFM	Actual Data Rate (f = 8 MHz)	Actual Data Rate (Variable f)
0	0	0	125 kbits/sec	f/64
0	0	1	250 kbits/sec	f/32
0	1	0	250 kbits/sec	f/32
0	1	1	500 kbits/sec	f/16
1	0	0	500 kbits/sec	f/16
1	0	1	1.00 Mbits/sec	f/8
1	1	0	test mode	
1	1	1	test mode	

f = clock frequency.

The test mode is used by National for testing purposes. It should not normally be used for anything else.

WRITE PRECOMPENSATION

Another parameter to set is the amount of write precompensation needed for the disk drive being used. This value is generally specified by the drive manufacturer. The amount of precompensation used is based on the Precomp Set pins and the Data Rate as shown in Table III.

If the amount of write precompensation is subject to change, then these pins could be connected to switches, an output port, or through some logic from the controller's drive select output.

It is sometimes desirable to enable write precompensation for the inner tracks of a disk only. Some controllers have an output signal that indicates when the head is over a track that needs write precompensation. The easiest way to implement this signal is to choose the amount of write precompensation needed, look up in the table which pins need to be tied high and which need to be tied low. Connect the low pins to ground. Connect the high pins to the controller's write precompensation enable output pin.

CRYSTAL

Normally an 8 MHz crystal is attached in parallel across the two oscillator pins. There should also be a separate 47 pF capacitor attached to each pin with the other side of each capacitor attached to ground. If the system already has an 8 MHz source, this may be used to drive the Osc 1 pin while leaving the Osc 2 pin floating. The frequency at this pin is used to set the center frequency of the VCO and the initial delay of the quarter period delay line. It is also used for the write precompensation circuit timing. See the AC Electrical Characteristics for the acceptable range of the crystal. Varying the frequency will affect many operating parameters as specified in the appropriate sections.

FILTER

The filter is used for the main PLL. The values recommended for the two resistors and the capacitor are given in Table IV based on the data rate needed. If more than one data rate will be used, there are two alternatives. The values can be used that are shown in the table for the multiple data rates. These values are a trade off of PLL characteristics that are not ideal for either data rate. Another alternative is to actually have two separate filters with the capability of switching in one or the other either with a manual switch or an analog switch which can be software controlled.

TABLE III

Precomp Set			Amount of Precompensation					
2	1	0	Data Rt = 00		Data Rt = 01		Data Rt = 10	
			f = 8 MHz	Variable f	f = 8 MHz	Variable f	f = 8 MHz	Variable f
0	0	0	0 ns	0X	0 ns	0X	0 ns	0X
0	0	1	107 ns	3X	36 ns	1X	36 ns	1X
0	1	0	143 ns	4X	71 ns	2X	71 ns	2X
0	1	1	179 ns	5X	107 ns	3X	107 ns	3X
1	0	0	214 ns	6X	143 ns	4X	143 ns	4X
1	0	1	250 ns	7X	179 ns	5X	179 ns	5X
1	1	0	321 ns	9X	214 ns	6X	illegal	
1	1	1	393 ns	11X	250 ns	7X	illegal	

X = 2/7f ns, where f = clock frequency.

TABLE IV

Data Rate	R1	R2	C1	C2
125 kbits/sec FM	kΩ	Ω	nF	μF
250 kbits/sec FM	kΩ	Ω	nF	μF
500 kbits/sec FM	kΩ	Ω	nF	μF
250 kbits/sec MFM	10.0 kΩ	100 Ω	4.7 nF	0.047 μF
500 kbits/sec MFM	kΩ	Ω	nF	μF
1 Mbit/sec MFM	kΩ	Ω	nF	μF
1.25 Mbits/sec MFM	kΩ	Ω	nF	μF
125 FM/250 MFM kbits/sec	kΩ	Ω	nF	μF
250 FM/500 MFM kbits/sec	kΩ	Ω	nF	μF
500 FM/1000 MFM kbits/sec	kΩ	Ω	nF	μF

Interfacing

DISK DRIVE INTERFACE

The connection between the Support Chip and the Disk Drive is very simple. The disk drive's Write Data line connects to the support chip's Write Data Out pin. The disk drive's Read Data line connects to the support chip's Read Data In pin.

FLOPPY CONTROLLER

Simply connect the Write Data and Read Data pins of the controller to the Write Data In and Read Data Out pins of the Support Chip. Connect the Early and Late Precomp outputs from the controller to the Early and Late Precomp inputs of the disk support chip.

The Read Gate input pin of the disk support chip must be connected to the pin of the controller that indicates when the controller is trying to read valid data. On the μPD765A, 8272A this is the VCO pin. On the WD179x this is the VFOE pin.

The Read Clock output pin of the disk support chip must be connected to the pin of the controller that requires a data window (or data clock). This is a window that defines whether an MFM encoded pulse is a data pulse or a clock pulse. The polarity of this pulse is indeterminant. On the μPD765A, 8272A this is the DW pin. On the WD179x this is the RCLK pin.

HARD DISK CONTROLLER

This floppy support chip has been designed to interface directly to the DP8466 Hard Disk Controller. Connect the Write Data lines exactly the same way as for the floppy controller. Connect the Write Precomp lines the same way also.

The hard disk's Read Data line should be connected to the support chip's Read Data In pin. Also, the controller's Read Gate output is connected to the Read Gate input of the support chip. The controller does not use the support chip's Read Data Out pin. The controller needs the data read from the disk to be in NRZ format rather than MFM encoded format. Simply connect the NRZ Read Data pin of the support chip to the Read Data pin of the controller. Connect the Read Clock output of the support chip to the Read Clock input of the controller.

The Address Mark Found output of the support chip gets connected to the Address Mark Found input of the controller.

Note: If write precompensation is used as well as AMF with the DP8466, the signal to indicate early precomp and the signal to indicate AMF must be multiplexed by the Write Gate output of the controller since the DP8466 combines these two functions on the same pin.

PLL Performance

The information in this section is not needed to use this part. It is included for completeness. The performance of the PLL is determined from the following factors:

K_{VCO} — Change in the frequency of the VCO due to a voltage change at the VCO input.

$$K_{VCO} \approx 10 \text{ MRad/s/volt.}$$

I_{CP} — Charge pump current. Set by the external resistor R_1 across the reference voltage set by the chip. $I_{CP} = 1.2\ V/R_1$. This current can be set anywhere between 50 μA and 350 μA. While in the high gain mode, the current is doubled.

C_2 — Filter capacitor.

R_2 — Filter resistor. Determines the PLL damping factor.

C_1 — This filter capacitor improves the performance of the PLL although it has only a secondary effect.

Using second order PLL formulas (i.e. ignoring the effect of C_1) the filter components can be chosen to obtain the required performance.

The bit jitter tolerance of the PLL is given by,

$$\omega_n = (K_{VCO}/2N \times I_{CP}/2\pi \times 1/C_2)^{1/2}$$

where N is the number of VCO cycles between two phase comparisons (N = 2 during the preamble).

The acquisition time (time to lock to the correct phase and frequency) is given by,

$$t_{lk} \approx 6/\omega_n.$$

The trade off, when choosing filter components, is between acquisition time while the PLL is locking and jitter immunity while reading data.

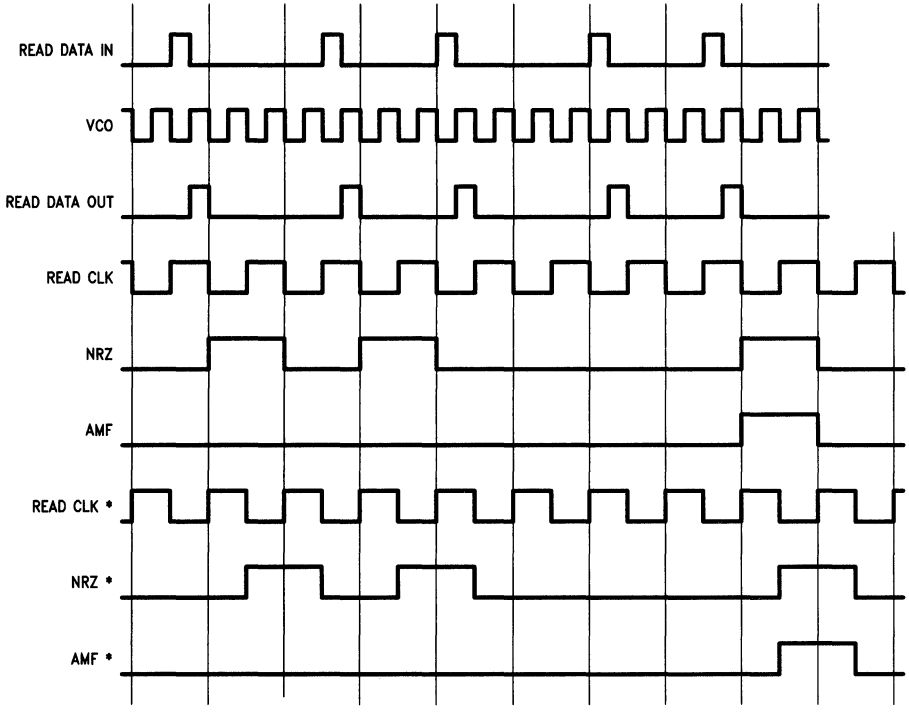
The damping factor is given by,

$$\zeta = \omega_n \times (R_2 \times C_2)/2$$

and is usually set at about 0.7.

Functional Waveform

Read Data Timing



TL/F/8593–6

* = If Read Clock starts out of phase.

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current	±20 mA

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage	0	V_{CC}	V
Operating Temperature Range (T_A):	0	+70	°C

DC Electrical Characteristics V_{CC} = 5V ±10% unless otherwise specified

Symbol	Parameter	Conditions	T_A = 0°C to +70°C Limits	Units
V_{IH}	Minimum High Level Input Voltage		2.0	V
V_{IL}	Maximum Low Level Input Voltage		0.8	V
V_{OH}	Minimum High Level Out	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 2 mA	3.7	V
V_{OL}	Maximum Low Level Out	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 2 mA	0.4	V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND	±1.0	μA
I_{OZ}	Maximum TRI-STATE® Leakage Current	V_{OUT} = V_{CC} or GND	±10.0	μA
I_{CC}	Maximum Supply Current	V_{IN} = V_{CC} or GND F_{IN} = 8 MHz	3.0	mA
I_{CC}	Maximum Supply Current	V_{IN} = 2.4V or 0.5V F_{IN} = 8 Mhz	20.0	mA

AC Electrical Characteristics V_{CC} = 5V ±10%, C = 150 pF, unless otherwise specified

Symbol	Parameter	T_A = 0°C to 70°C				Units
		f = 8 MHz		f = variable		
		Min	Max	Min	Max	
f	Crystal Frequency	4	10			MHz
DR	Data Rate	125	1250			Kbit/s
READ TIMING						
t_{DRS}	Data Rate Setup to Data In	Depends on Filter Used				
t_{RMS}	Read Mode Setup to Read Gate	100				ns
t_{FMS}	FM/MFM Setup to Data In	Depends on Filter Used				
t_{RGS}	Read Gate Setup to Data In	600		$100 + \frac{4\times10^9}{f}$		ns
t_{DRH}	Data Rate Hold from Read Gate	2 Bit Windows				
t_{RMH}	Read Mode Hold from Read Gate	2 Bit Windows				

AC Electrical Characteristics V_{CC} = 5V ±10%, C = 150 pF, unless otherwise specified (Continued)

Symbol	Parameter	T_A = 0°C to 70°C				Units
		f = 8 MHz		f = variable		
		Min	Max	Min	Max	
READ TIMING (Continued)						
t_{FMH}	FM/MFM Hold from Read Gate	2 Bit Windows				
t_{RGH}	Last Data In to Read Gate Disable	2 Bit Windows				
t_{RGF}	Read Gate Off Time between Reads	600				ns
t_{RDO}	Read Data Offset from Center of Read Clock		34			ns
t_{NRZ}	NRZ & AMF Data Offset from Read Clock Edge		20			ns
t_{IN}	Pulse Width of Data In	50				ns
t_{OUT}	Pulse Width of Data Out	110	200	$\frac{8.8\times10^8}{f}$	$\frac{1.6\times10^9}{f}$	ns
WRITE TIMING						
t_{DRWS}	Data Rate Setup to Write Data In	125				ns
t_{PSS}	Precomp. Setup to Write Data In	125				ns
t_{RGS}	Read Gate Setup to Write Data In	600		$100 + \frac{4 \times 10^9}{f}$		ns
t_{PS}	Early/Late Setup to Write Data In	−160 (Note 1)				ns
t_{PH}	Early/Late Hold from Write Data In	200				ns
t_{DRWH}	Data Rate Hold from Write Data In	1.0				µs
t_{PSH}	Precomp. Hold from Write Data In	1.0				µs
t_{RGH}	Read Gate Hold from Write Data In	1.0				µs
t_{WI}	Write In Pulse Width	20				ns
t_{WO}	Write Out Pulse Width	220	350	$\frac{1.76 \times 10^9}{f}$	$\frac{2.8 \times 10^9}{f}$	ns
t_{IO}	Write In to Write Out	280 Typ. (Early Precomp.)		$30 + \frac{2 \times 10^9}{f}$		ns
e_{WP}	Error of Write Precomp.		±10			%

Note 1: The Early and Late pins do not need to be valid until 160 ns after the rising edge of the write data in signal. This is to accommodate interfacing to the µPD765A.

PLL Characteristics

Symbol	Parameter	Value	
$K_{\phi High}$	Phase Comparator & Charge Pump Gain Constant. High Gain Mode. (Note 1)	$\frac{5\ V_{REF}}{2\pi R}$	Typ.
V_{REF}	Voltage at Set Pump Current Pin	1.2V	Typ.
$K_{\phi Low}$	Low Gain Mode (Note 1)	$\frac{2.5\ V_{REF}}{2\pi R}$	Typ.
K_{VCO}	Gain of VCO (Note 2)	5/N MRad/S/V	Typ.
f_{VCO}	Center Frequency of VCO	f/2	Typ.
t_{JITTER}	Maximum Tolerance of Bit Jitter (Note 3)	$\frac{(0.95)}{4 \times DR}$	Typ.
$t_{POWER\ ON}$	Time from Full V_{CC} Power to Guaranteed Functionality	50 ms	Max

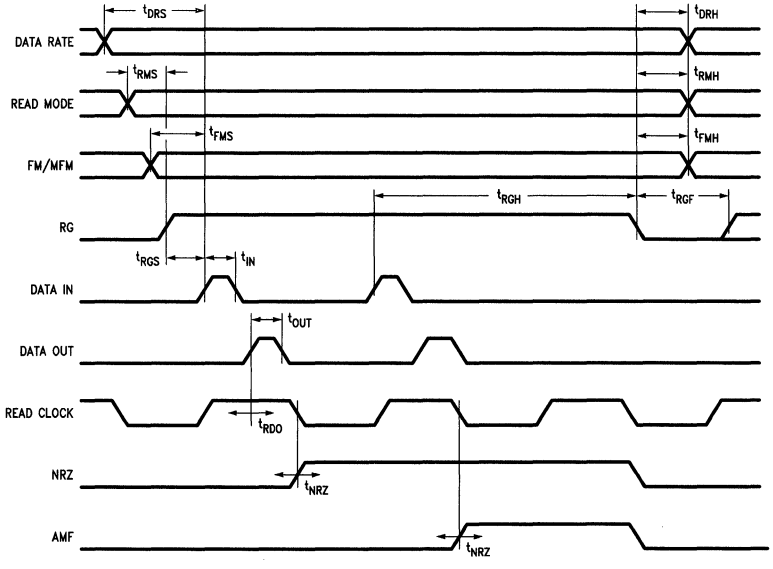
Note 1: R = pump current set resistor (8k–20k).

Note 2: N = # of VCO cycles per bit.

Note 3: DR = Data Rate.

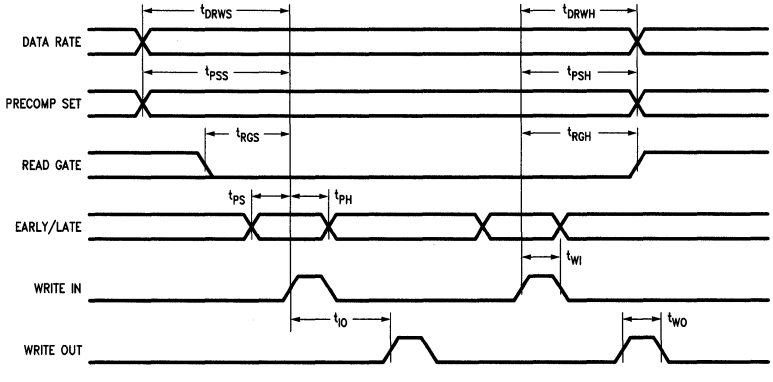
Timing Diagrams

Read Timing



T /F/8593–7

Write Timing



TL/F/8593–8

National Semiconductor Corporation

PRELIMINARY

DP8472, DP8474 Floppy Disk Controller Plus

General Description

This is a full featured Floppy Disk Drive Controller. It is software compatible with the μPD765A but also has many enhancements over the μPD765A. This includes an internal data separator, internal write precompensation, a motor on/off control, internal line drivers, low power mode, and some software enhancements that simplify programming the DP8472, 74.

The internal data separator uses a combination of digital and analog circuits. The analog PLL requires only fixed value external components, no trims are needed.

The internal Write Precompensation can be programmed to shift the outgoing data early or late anywhere between 0 and 464 ns. It uses a standard single level shifting algorithm.

The Head Load and Head Unload timers can be redefined as a Motor On and Motor Off time. This redefinition allows the longer times needed for a disk drive motor to come up to speed (up to four seconds).

The low power feature allows the crystal of the controller to be turned off by software control or it may be programmed to turn off automatically when all drive motors are off. This reduces the power consumption of the controller to less than 100 μA.

The output buffers of the signals to the disk drive can sink up to 8 mA. They can also be active high or active low on the DP8474. If the length of the cable connecting the disk drive to the controller is relatively small, the drive can be connected directly to the controller without any buffers.

Other enhancements that may be enabled or disabled include implied seeks that will automatically move the drive head to the correct position in many commands. This elimi-

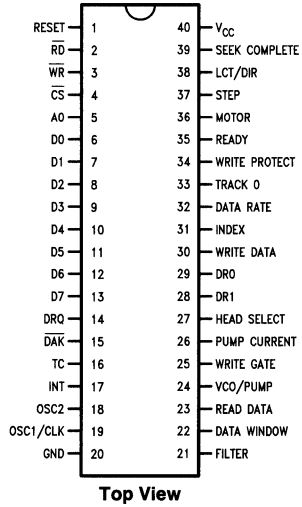
(continued on next page)

Features

- Internal dual gain, analog data separator
- Internal write precompensation (variable starting track #)
- Software compatible with the μPD765A/8272A
- CMOS
- Software selectable data rate (125 kbits–2.5 Mbits)
- No trimmable passive components needed
- Compatible with external data separator
- Low Power mode (I_{CC} < 100 μA)
- Implied seeks on read and write commands
- Disable polling mode
- Can redefine timers for motor on/off
- Seek Complete input for buffered seeks
- Demultiplexed drive select outputs (DP8474)
- Extended track range (up to 4096 tracks)
- Format with or w/o Index Address Mark (μFloppy compatible)

Connection Diagrams

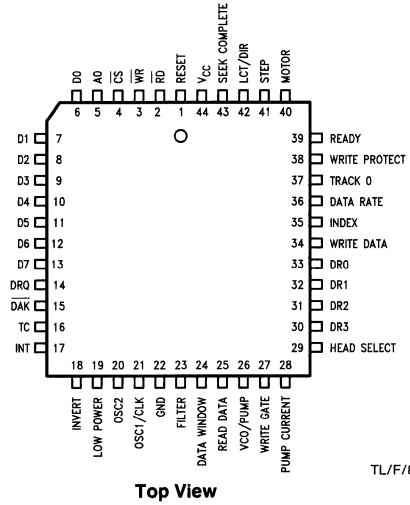
DP8472 Dual-In-Line Package



TL/F/8592-1

Top View
Order Number DP8472J or DP8472N
See NS Package Numbers N40A, J40A

DP8474 Plastic Chip Carrier Package



TL/F/8592-2

Top View
Order Number DP8474V
See NS Package Number V44A

General Description (Continued)

nates the need for issuing the Seek command and the Sense Interrupt command. Another enhancement is the ability to disable the polling mode. Also, the motor multiplexing scheme may be programmed so that the controller knows whether all the drive motors are on at the same time or if only one is on at a time.

If the command used to control these new features is not accessed, the controller will use default modes that are compatible with software written for use with the μPD765A.

There are two different package types. The DP8472 is a 40 pin DIP package. The DP8474 is a 44 pin PCC package.

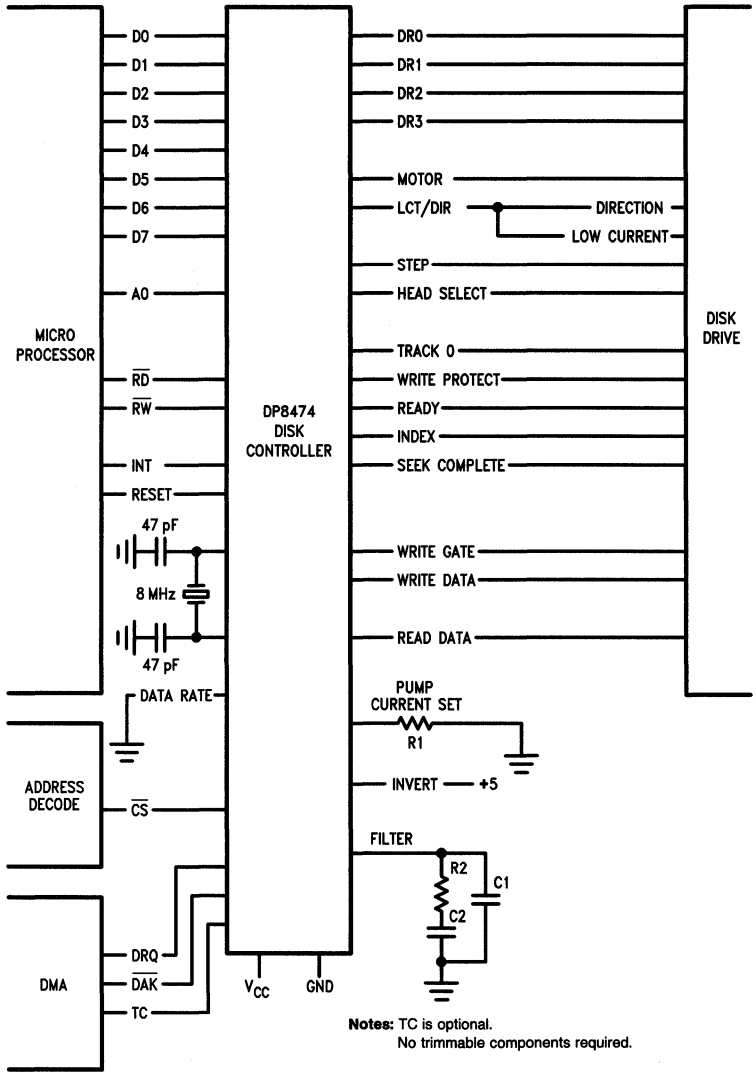
Pin Descriptions

Symbol	DP8472 DIP Pin No.	DP8474 PCC Pin No.	Function
RESET	1	1	Active high input that resets the controller to the idle state and resets all output lines to the disk drive to their disabled state. Does not clear any internal registers except the registers defined in the Mode command. These registers will be set to their default values.
$\overline{RD}$	2	2	Active low input to signal a read from the controller to the microprocessor.
$\overline{WR}$	3	3	Active low input to signal a write from the microprocessor to the controller.
$\overline{CS}$	4	4	Active low input to enable the $\overline{RD}$ and $\overline{WR}$ inputs.
A0	5	5	Address line from the microprocessor. This determines which register the microprocessor is talking to: Data or Status Register.
D0–D7	6–13	6–13	Bi-directional data lines to the microprocessor.
DRQ	14	14	Active high output to signal the DMA controller that a data transfer is needed.
$\overline{DAK}$	15	15	Active low input to acknowledge the DMA request and enable a read or a write.
TC	16	16	Active high input to indicate the termination of a DMA transfer.
INT	17	17	Active high output to signal that an operation requires the attention of the microprocessor. The action required depends on the current function of the controller.
INVERT		18	(DP8474 only) This input determines the polarity of the disk drive interface lines (input & output). Low indicates active high push-pull signals. High indicates active low open drain signals.
LOW POWER		19	(DP8474 only) This active high output indicates when the controller is in its low power mode. This can be connected to a drive that has a low power input signal. Affected by INVERT.
OSC2	18	20	An external crystal is attached here or it is left open if an external clock is used.
CLK/OSC1	19	21	An external crystal or the output of an external clock is attached here. (Usually 8 MHz)
FILTER	21	23	This pin is the output of the dual gain charge pump and is also the input to the VCO. A simple filter is attached to this pin.
DATA WINDOW	22	24	If the internal data separator is disabled, the signal that indicates that the incoming raw data is in the data phase or the clock phase is connected here. This pin is not used if the internal data separator is enabled and should be tied low or high.
READ DATA	23	25	If the internal data separator is enabled, the raw data read from the disk is connected here. If the internal data separator is disabled, the synchronized data signal from an external data separator is connected here. Affected by INVERT.

Pin Descriptions (Continued)

Symbol	DP8472 DIP Pin No.	DP8474 PCC Pin No.	Function
VCO/PUMP	24	26	This active high output enables an external data separator to synchronize to the disk data instead of its center frequency. If the internal data separator is used, this output is the OR of the internal Pump Up and Pump Down signal. This can be used for diagnostic purposes.
WRITE GATE	25	27	This active high output enables the write circuitry of the selected disk drive. Affected by INVERT.
PUMP CURRENT	26	28	A resistor is attached to this pin to set the charge pump current.
HEAD SELECT	27	29	This output determines which disk drive head is active. (low = head 0, high = head 1). Affected by INVERT.
DR0 DR1	29 28		(DP8472) Active high outputs that indicates in binary form which disk drive is active. Affected by INVERT.
DR0 DR1 DR2 DR3		33 32 31 30	(DP8474) Active high outputs to select which disk drive is active. These pins are the demultiplexed DR0, DR1 pins described above. In addition, if no drive is currently selected, no signal will be active. Affected by INVERT.
WRITE DATA	30	34	This is the write precompensated serial data to be written onto the selected disk drive. Affected by INVERT.
INDEX	31	35	This active high input signals the beginning of a track. Affected by INVERT.
DATA RATE	32	36	This input selects the data rate used if the Mode command is not accessed. High indicates 500 kbits/sec (MFM), Low indicates 250 kbits/sec (based on 8 MHz clock). The data rate may be overridden in software through the Mode command. This pin also effects the times programmed with the Specify command. The times are doubled if this pin is low. This doubling effect is not overridden by the Mode Command.
TRACK 0	33	37	This active high input tells the controller that the head is at track zero of the selected disk drive. Affected by INVERT.
WRITE PROTECT	34	38	This active high input tells the controller that the disk is write protected. Any command that writes to the disk drive is not permitted when a disk is write protected. Affected by INVERT.
READY	35	39	This active high input tells the controller that the selected disk drive is ready (i.e. the drive door is closed, may also indicate that the disk is spinning). Affected by INVERT.
MOTOR	36	40	Active high output to turn on the disk drive motor for 5.25″ drives. May also be used to load the head of an 8″ drive. Affected by INVERT.
STEP	37	41	This active high output will produce a pulse at a software programmable rate to move the head during a seek. Affected by INVERT.
LCT/DIR	38	42	When in the seek mode this output will determine the direction of the head movement (high = step in, low = step out). When in the write or read mode this output will go high when the controller detects that the current track number is greater than or equal to a software programmable track number. Affected by INVERT.
SEEK COMPLETE	39	43	This active high input from the disk drive indicates that a buffered seek operation is complete. This input may be tied high if drive does not support buffered seeks. Affected by INVERT.
V_{CC} GROUND	40 20	44 22	These pins are the power supply pins for both the digital circuitry and the analog circuitry.

Typical Application



Notes: TC is optional.
No trimmable components required.

TL/F/8592–3

Functional Description

There are only two registers to access in the floppy disk controller. The read only Main Status Register is used to detect the current status of the controller. The Data Register is used for several purposes. Commands and command parameters are passed to the Data Register. While reading or writing to a disk, the data is transferred through the Data Register. Finally, the result of a command after it is finished is read from the Data Register.

COMMAND SEQUENCE

The disk controller can perform many commands. There are three phases for every command that is executed.

COMMAND PHASE: The μP writes a series of bytes to the Data Register. These bytes indicate the command desired and the particular parameters required for the command.

EXECUTION PHASE: The disk controller performs the desired command. Some commands require the μP to read or write data to or from the Data Register during this time. Reading data from a disk is an example of this.

RESULT PHASE: The μP reads a series of bytes from the Data Register. These bytes indicate whether the command executed properly and other pertinent information.

Each command requires a set of bytes to be written to the disk controller in the Command Phase. All the bytes must be written in the order specified in the Command Description Table. The Execution Phase starts immediately after the last byte in the Command Phase is written.

After the end of the Execution Phase, the Result Phase bytes may be read from the disk controller. The bytes are

Functional Description (Continued)

read in the order specified in the Command Description Table. All the result bytes need not be read, although it is recommended to read them all.

A new command may be initiated by writing the Command Phase bytes after the last bytes needed from the Result Phase have been read.

MAIN STATUS REGISTER (A0 = 0)

The read only Main Status Register indicates the current status of the disk controller. The Main Status Register is always available to be read. One of its functions is to control the flow of data to and from the Data Register. The Main Status Register indicates when the disk controller is ready to send or receive data. It should be read before any byte is transferred to or from the Data Register.

Main Status Register

D7 **Request for Master:** Indicates that the Data Register is ready to send or receive data from the μP. This bit is cleared immediately after a byte transfer and will become set again as soon as the disk controller is ready for the next byte.

D6 **Data Direction:** Indicates whether the controller is expecting a byte to be written to (0) or read from (1) the Data Register. Used in combination with Request for Master (D7).

D5 **Non-DMA Execution:** Bit is set only during the Execution Phase of a command if it is in the non-DMA mode. In other words, if this bit is set, the multiple byte data transfer (in the Execution Phase) must be monitored by the μP either through interrupts, or software polling as described below.

D4 **Command in Progress:** This bit is set after the first byte of the Command Phase is written. This bit is cleared after the last byte of the Result Phase is read. If there is no result phase in a command, the bit is cleared after the last byte of the Command Phase is written.

D3 **Drive 3 Seeking:** Set after the last byte is written in the Command Phase of a Seek or Recalibrate command for drive 3. Cleared after reading the first byte in the Result Phase of the Sense Interrupt Command.

D2 **Drive 2 Seeking:** Same as above for drive 2.

D1 **Drive 1 Seeking:** Same as above for drive 1.

D0 **Drive 0 Seeking:** Same as above for drive 0.

PROCESSOR INTERFACE

Bytes are transferred to and from the disk controller in different ways for the different phases in a command. During the Command Phase and the Result Phase, bytes are transferred using the Main Status Register (A0 = 0) to control the timing and direction of transfer. Bit 6 of the Main Status Register must be clear and bit 7 must be set before a byte can be written to the Data Register (A0 = 1) during the Command Phase. Bits 6 and 7 of the Main Status Register must both be set before a byte can be read from the Data Register during the Result Phase.

If there is information to be transferred during the Execution Phase, there are three methods that can be used. The DMA mode is used if the system has a DMA controller. This allows the μP to do other things during the Execution Phase data transfer. If DMA is not used, an interrupt can be issued for each byte transferred during the Execution Phase. If interrupts are not used, the Main Status Register can be polled to indicate when a byte transfer is required.

DMA MODE

If the DMA mode is selected, a DMA request will be generated in the Execution Phase when each byte is ready to be transferred. The DMA controller should respond to the DMA request with a DMA acknowledge and the $\overline{RD}$ or the $\overline{WR}$ signal. The DMA request will be cleared by the active edge of the DMA acknowledge. After the last byte is transferred, an interrupt is generated. This indicates the beginning of the Result Phase. The interrupt will be cleared by reading the first byte in the Result Phase.

INTERRUPT MODE

If the non-DMA mode is selected, an interrupt will be generated in the Execution Phase when each byte is ready to be transferred. The Main Status Register should be read to verify that the interrupt is for a data transfer. Bits 5 and 7 of the Main Status Register will be set. When the μP reads the data byte from the Data Register, the interrupt will be cleared. The μP should read the byte within the time allotted by the following Time to Service Interrupt Table. If the byte is not transferred within the time allotted, an Overrun Error will be indicated in the Result Phase when the command terminates. An additional interrupt will be generated after the last byte is transferred. This indicates the beginning of the Result Phase. Bits 7 and 6 of the Main Status Register will be set and bit 5 will be clear. This interrupt will be cleared by reading the first byte in the Result Phase.

Time to Service Interrupt

Data Rate	Clk Frequency	Time to Service INT
125 kbits/sec	8 MHz	62.0 μs
250 kbits/sec	8 MHz	30.0 μs
500 kbits/sec	8 MHz	14.0 μs
1000 kbits/sec	8 MHz	6.0 μs
1250 kbits/sec	10 MHz	4.4 μs
2500 kbits/sec	20 MHz	2.2 μs

Time = (8/DR) − (16/f), where DR = Data Rate, f = clock frequency.

SOFTWARE POLLING

If the non-DMA mode is selected and interrupts are not suitable, the μP can poll the Main Status Register during the Execution Phase to determine when a byte is ready to be transferred. During the Execution Phase, in the non-DMA mode, bit 7 of the Main Status Register mirrors the state of the interrupt pin. Otherwise, the data transfer is similar to the Interrupt Mode described above.

DATA RATE

The data rate is determined by three factors; the clock frequency, the data rate pin, and the data rate programmed via the Mode command. Normally an 8 MHz crystal is used as the clock. If this is the case, the data rate pin can be used to select between 250 kbits/sec or 500 kbits/sec (MFM). If a different value clock frequency is used, the data rate is given by the formulas:

data rate = 2×10^{12} / f bits/sec (data rate pin low)

data rate = 4×10^{12} / f bits/sec (data rate pin high)

The Mode command can be used to select the data rate via software. This method gives better flexibility than the data rate pin. With an 8 MHz clock, the following data rates can be selected: 250, 500, or 1000 kbits/sec. With a different clock frequency, the data rate can be calculated by the formulas in the table with the Mode command description.

Functional Description (Continued)

All of the data rates specified in the previous two paragraphs are for MFM encoded data. If FM is used, the data rates are halved.

It is important to note that the internal data separator will not function correctly above a data rate of 1.5 Mbits/sec. The write precompensation logic will also fail above this data rate. Therefore, an external data separator and write precompensation circuit must be used at data rates above 1.5 Mbits/sec.

DATA SEPARATOR

The internal data separator consists of a dual gain analog PLL. This PLL synchronizes the raw data signal read from a disk drive. The synchronized signal is then used to separate the clock and data pulses from the raw signal. The data pulses are grouped into bytes and then sent to the μP.

The PLL consists of three main components, a phase comparator, a filter, and a voltage controlled oscillator (VCO). The basic operation of a PLL is fairly straightforward. The phase comparator detects the difference between the phase of the VCO output and the phase of the raw data being read from the disk. This phase difference is converted to a current which is either charges or discharges a filter. The resulting voltage of the filter changes the frequency of the VCO in an attempt to reduce the phase difference between the two signals. A PLL is "locked" when the frequency of the VCO is exactly the same as the average frequency of the read data. This is somewhat of a simplified view because it ignores such topics as loop stability, acquisition time, and filter values.

The external filter simply consists of two capacitors and a resistor as shown in the typical application diagram. Another resistor is used to set the charge pump current.

The quarter period delay line is used to determine the center of a bit cell. It is important that this delay line be as accurate as possible. A typical data separator would normally require an external trim to adjust the delay. An external trim is not required for the DP8472/74 however. A secondary PLL is used to automatically calibrate the delay line. The secondary PLL also calibrates the center frequency of the VCO.

The quarter period delay line can be programmed to always be set at the ideal delay. It can also be programmed to follow the actual data rate. It does this by following the frequency of the main VCO.

The Preamble Detect circuit is used with an intelligent algorithm to determine when the PLL switches from the high gain mode to its low gain mode. This circuit scans the incoming data for the frequency corresponding to a preamble plus or minus 15 percent.

PLL PERFORMANCE

The information in this section is not needed to use this part. It is included for completeness. The performance of the PLL is determined from the following factors:

K_{VCO}— Change in the frequency of the VCO due to a voltage change at the VCO input.

$$K_{VCO} \approx 10 \text{ MRad/s/volt.}$$

I_{CP}— Charge pump current. Set by the external resistor R_1 across the reference voltage set by the chip. $I_{CP} = 1.2 \text{ V}/R_1$. This current can be set anywhere between 50 μA and 350 μA. While in the high gain mode, the current is doubled.

C_2— Filter capacitor.

R_2— Filter resistor. Determines the PLL damping factor.

C_1— This filter capacitor improves the performance of the PLL although it has only a secondary effect.

Using second order PLL formulas (i.e., ignoring the effect of C_1) the filter components can be chosen to obtain the required performance.

The bit jitter tolerance of the PLL is given by,

$$\omega_n = (K_{VCO}/2N \times I_{CP}/2\pi \times 1/C_2)^{1/2}$$

where N is the number of VCO cycles between two phase comparisons (N = 2 during the preamble).

The acquisition time (time to lock to the correct phase and frequency) is given by,

$$t_{lk} \approx 6/\omega_n.$$

The trade off, when choosing filter components is between acquisition time while the PLL is locking and jitter immunity while reading data.

The damping factor is given by,

$$\zeta = \omega_n \times (R_2 \times C_2)/2$$

and is usually set at about 0.7.

Other Features

DRIVE POLLING

If the Polling Mode is enabled, the disk controller will poll the drives continuously while it is in the idle state. The idle state is after the last byte of the Result Phase has been read and before the first byte of the Command Phase has been written. The disk controller will select each drive and check to see if the ready signal has changed states since the last time it checked. If a drive has changed its ready state, an interrupt is generated by the controller. The Sense Interrupt command should be used to identify and clear this interrupt. The Polling Mode can be enabled and disabled through the Mode Command.

LOW POWER MODE

In the Low Power Mode the crystal oscillator is turned off. When the oscillator is turned off the controller will draw less than 100 μA. Also, the internal circuitry is disabled from doing anything when the oscillator is off, since the internal circuitry is driven from this oscillator. The oscillator will turn back on automatically after it detects $\overline{CS}$ & $\overline{RD}$ or $\overline{CS}$ & $\overline{WR}$ being activated. It may take a few milliseconds for the oscillator to return to full frequency, and the μP will be prevented from trying to access the Data Register during this time through the normal Main Status Register protocol. The Controller will go back to low power mode any time it is idle for more than 500 ms (based on 8 MHz clock).

There are two ways to go into the low power mode. One is to command the controller to switch to low power immediately through a software command. The other method is to set the controller to automatically go into the low power mode whenever all the disk drive motors are off (after the Motor Off time expires). This would be invisible to the software. The low power mode is programmed through the Mode Command.

SEEK COMPLETE

The seek complete signal is available on any disk drive that supports buffered seeks. If the drive used does not have this signal available, simply tie this input high.

Result Phase Status Registers

The Result Phase of a command usually contains bytes that hold status information. The format of these bytes are the same for each command and are described below. Do not confuse these bytes with the Main Status Register which is a read only register that is always available. The Result Phase status registers are read from the Data Register only during the Result Phase.

STATUS REGISTER 0 (ST0)

D7 Interrupt Code:

D6 00 = Normal termination of command. Command was completed and properly executed.

01 = Abnormal termination of command. Execution of command was started, but was not successfully completed.

10 = Invalid command issue. Command issued was not recognized as a valid command.

11 = Ready changed state during the polling mode.

D5 Seek End: Seek or Recalibrate command completed by the controller. (Used during Sense Interrupt command.)

D4 Equipment Check: After a Recalibrate command, Track 0 signal failed to occur. (Used during Sense Interrupt command.)

D3 Not Ready: Drive is not ready by 5 disk revolutions after a Read or Write Command. Inverse of current state of ready signal during Sense Interrupt command.

D2 Head Address: (at end of Execution Phase).

D1 Unit Select: (at end of Execution Phase).

D0 00 = Drive 0 selected.
01 = Drive 1 selected.
10 = Drive 2 selected.
11 = Drive 3 selected.

STATUS REGISTER 1 (ST1)

D7 **End of Track:** Controller attempted to access a sector number greater than that programmed by the End of Track (EOT) byte in Command Phase.

D6 **Not Used:** 0.

D5 **CRC Error:** Controller detected a CRC error in the Address Field or the Data Field, depending on the state of bit 5 of ST2.

D4 **Over Run:** Controller was not serviced by the μP soon enough during a data transfer in the Execution Phase.

D3 **Not Used:** 0.

D2 **No Data:** Three possible problems: 1) Controller cannot find the sector specified in the Command Phase during the execution of a Read, Write, or Scan command. An address mark was found however, so it is not a blank disk. 2) Controller cannot read any Address Fields without a CRC error during Read ID command. 3) Controller cannot find starting sector during execution of Read A Track command.

D1 **Not Writable:** Controller detected a write protect signal from the drive during execution of Write Data, Write Deleted Data, or Format A Track commands.

D0 **Missing Address Mark:** If bit 0 of ST2 is clear, then the disk controller cannot detect any Address Field Address Mark after encountering the index hole twice. If bit 0 of ST2 is set, then the disk controller cannot detect the Data Address Mark or Deleted Data Address Mark of the Data Field.

STATUS REGISTER 2 (ST2)

D7 **Not Used:** 0.

D6 **Control Mark:** Controller tried to read a sector which contained a deleted data address mark during execution of Read Data or Scan commands. Or, if a Read Deleted Data command was executed, a regular address mark was detected.

D5 **CRC Error in Data Field:** Controller detected a CRC error in the Data Field. Bit 5 of ST1 is also set.

D4 **Wrong Track:** Only set if desired sector not found. The track number recorded on any sector on the track is different from that stored in the Track Register.

D3 **Scan Equal Hit:** "Equal" condition satisfied during any Scan Command.

D2 **Scan Not Satisfied:** Controller cannot find a sector on the track which meets the desired condition during Scan Command.

D1 **Bad Track:** Only set if the desired sector is not found. The track number recorded on any sector on the track is different from that stored in the Track Register and the recorded track number is FF.

D0 **Missing Address Mark in Data Field:** Controller cannot find a Data Address Mark during a read command. Bit 0 of ST1 is also set.

STATUS REGISTER 3 (ST3)

D7 **Not used:** 0.

D6 **Write Protect Signal.**

D5 **Ready.**

D4 **Track 0**

D3 **Not used:** 0.

D2 **Head Address.**

D1 00 = Drive 0 selected.
D0 01 = Drive 1 selected.
10 = Drive 2 selected.
11 = Drive 3 selected.

COMMAND DESCRIPTION TABLE

READ DATA

Command Phase:

MT	MFM	SK	0	0	1	1	0
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Data Length							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

READ DELETED DATA

Command Phase:

MT	MFM	SK	0	1	1	0	0
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Data Length							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

READ A TRACK

Command Phase:

0	MFM	SK	0	0	0	1	0
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Data Length							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

WRITE DATA

Command Phase:

MT	MFM	0	0	0	1	0	1
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Data Length							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

WRITE DELETED DATA

Command Phase:

MT	MFM	0	0	1	0	0	1
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Data Length							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

READ ID

Command Phase:

0	MFM	0	0	1	0	1	0
0	0	0	0	0	HD	DR1	DR0

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

Notes: MT = Multi-Track
SK = Skip
IPS = Implied Seek
HD = Head #
DR = Drive Select

COMMAND DESCRIPTION TABLE (Continued)

SCAN EQUAL

Command Phase:

MT	MFM	SK	1	0	0	0	1
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Sector Step Process							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

SCAN LOW OR EQUAL

Command Phase:

MT	MFM	SK	1	1	0	0	1
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Sector Step Process							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

SCAN HIGH OR EQUAL

Command Phase:

MT	MFM	SK	1	1	1	0	1
IPS	0	0	0	0	HD	DR1	DR0
Track Number							
Head Number							
Sector Number							
Number Data Bytes/Sector							
End of Track							
Gap Length							
Sector Step Process							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

FORMAT A TRACK

Command Phase:

0	MFM	0	0	1	1	0	1
0	0	0	0	0	HD	DR1	DR0
Number Data Bytes/Sector							
Sectors per Track							
Gap Length							
Data Pattern							

Result Phase:

Status Register 0
Status Register 1
Status Register 2
Track Number
Head Number
Sector Number
Number Data Bytes/Sector

Notes: *—This byte only written or read if the extended track range mode is enabled. (12 bit head #)

TMR = Timer mode

IDX = No index address mark

IPS = Implied Seek

MMX = All motors assumed on if any on

POL = Polling mode

HDR = Extended track range

DRE = Software data rate enable

ANR = Abort Not Ready

EL = Early/Late output enable

WLD = No Wildcard in scan

DTS = Internal data separator

¼ = ¼ period delay tracks data

SEEK

Command Phase:

0	0	0	0	1	1	1	1
0	0	0	0	0	HD	DR1	DR0
New Track Number (NTN)							
*MSB of NTN			0	0	0	0	

No Result Phase

SENSE INTERRUPT STATUS

Command Phase:

0	0	0	0	1	0	0	0

Result Phase:

Status Register 0							
Present Track Number (PTN)							
*MSB of PTN			0	0	0	0	

SPECIFY

Command Phase:

0	0	0	0	0	0	1	1
Step Rate Time				Motor Off Time			
Motor On Time							DMA

No Result Phase

SET TRACK

Command Phase:

0	R/W	1	0	0	0	0	1
Internal Register #						DR1	DR0
New Value (or dummy)							

Result Phase:

Value

RECALIBRATE

Command Phase:

0	0	0	0	0	1	1	1
0	0	0	0	0	0	DR1	DR0

No Result Phase

SENSE DRIVE STATUS

Command Phase:

0	0	0	0	0	1	0	0
0	0	0	0	0	HD	DR1	DR0

Result Phase:

Status Register 3

MODE

Command Phase:

0	0	0	0	0	0	0	1
TMR	IDX	IPS	MMX	LOW PWR		POL	HDR
Low Current & Precomp Track #							
DRE	ANR	EL	WLD	Head Settle			
DTS	1/4	Data Rt		Write PreComp			

No Result Phase

INVALID COMMAND

Command Phase:

Invalid Codes

Result Phase:

Status Register 0

Command Descriptions

READ DATA

The Read Data op-code is written to the data register followed by 8 bytes as specified in the Command Description table. After the last byte is written, the controller turns on the correct drive and starts looking for the sector specified. Once the sector is found the controller sends the data to the μP. After one sector is finished, the Sector Number is incremented by one and this new sector is searched for. If MT (Multi-Track) is set, both sides of one track can be read. Starting on side zero, the sectors are read until the sector number specified by End of Track is reached. Then, side one is read starting with sector number one.

The Read Data command continues to read until the TC pin is set. This means that the DMA controller should be programmed to transfer the correct number of bytes. TC could be controlled by the μP and be asserted when enough bytes are received. An alternative to these methods of stopping the Read Data command is to program the End of Track to be the last sector number that needs to be read. The controller will stop reading the disk with an error indicating that it tried to access a sector number beyond the end of the track.

The Number of Data Bytes per Sector parameter is defined in Table I. If this is set to zero, the Data Length parameter determines the number of bytes that the controller transfers to the μP. If the data length specified is smaller than 128, the controller still reads the entire 128 byte sector and checks the CRC though only the number of bytes specified by the Data Length parameter are transferred to the μP. If the Number of Bytes per Sector parameter is not zero, the Data Length parameter has no meaning and should be set to FF(hex).

TABLE I

# Bytes/Sector Code	Actual # Bytes
0	128
1	256
2	512
3	1024
4	2048
5	4096
6	8192

After the last byte of the Command Phase is written by the μP, the controller will turn on the correct drive. If the drive was previously off, the Motor On Timer will be enabled.

If the Implied Seek Mode is enabled by both the Mode command and the IPS bit in this command, a Seek is performed to the track number specified in the Command Phase.

The controller then waits for all four of the following conditions to occur. 1) The Motor On time must expire, 2) the Ready line must be true, 3) the Seek Complete input must be true and, 4) if an Implied Seek was performed, the Head Settle Time programmed in the Mode Command must expire. The controller will wait up to 5 disk revolutions (counted by Index Pulses) after the Motor On Timer times out for the Ready and the Seek Complete to become true. If 5 revolutions pass, bit 3 of ST0 (Not Ready) is set and an abnormal termination is indicated (bit 7 and 6 of ST0 is 0 and 1 respectively). If a byte is written by the μP during this waiting time, the command is aborted and an abnormal termination is indicated. This way, if the index signal is not active (no disk in drive), the controller will not hang up forever.

After all these conditions are true, the controller searches for the specified sector by comparing the track #, head #, sector #, and number of bytes/sector given in the Command Phase with the appropriate bytes read off the disk in the Address Fields.

If the correct sector is found, but there is a CRC error in the Address Field, bit 5 of ST1 (CRC Error) is set and an abnormal termination is indicated. If the correct sector is not found, bit 2 of ST1 (No Data) is set and an abnormal termination is indicated. In addition to this, if any Address Field track # is FF, bit 1 of ST2 (Bad Track) is set or if any Address Field track # is different from that specified in the Command Phase, bit 4 of ST2 (Wrong Track) is set.

After finding the correct sector, the controller reads that Data Field. If a Deleted Data Mark is found and the SK bit is set, the sector is not read, bit 6 of ST2 (Control Mark) is set, and the next sector is searched for. If a deleted data mark is found and the SK bit is not set, the sector is read, bit 6 of ST2 (Control Mark) is set, and the read terminates with a normal termination. If a CRC error is detected in the Data Field, bit 5 is set in both ST1 and ST2 (CRC Error) and an abnormal termination is indicated.

If no problems occur in the read command, the read will continue from one sector to the next in logical order (not physical order) until either TC is set or an error occurs.

An interrupt will be generated when the Execution Phase of the Read Data command terminates. The values that will be read back in the Result Phase are shown in Table II. If an error occurs, the result bytes will indicate the sector being read when the error occurred.

Command Descriptions (Continued)

TABLE II

MT	HD	Final Sector Xfered to μP	ID Information at Result Phase			
			Track	Head	Sector	Bytes
0	0	< EOT	NC	NC	S+1	NC
		= EOT	T+1	NC	1	NC
	1	< EOT	NC	NC	S+1	NC
		= EOT	T+1	NC	1	NC
1	0	< EOT	NC	NC	S+1	NC
		= EOT	NC	LSB	1	NC
	1	< EOT	NC	NC	S+1	NC
		= EOT	T+1	LSB	1	NC

EOT = End of Track
NC = No Change
LSB = Least Significant Bit = 1
T = Track # Programmed
S = Last Sector # Read

READ DELETED DATA

This command is the same as the Read Data command except for its treatment of a Deleted Data Mark. If a Deleted Data Mark is read, the sector is read normally. If a regular Data Mark is found and the SK bit is set, the sector is not read, bit 6 of ST2 (Control Mark) is set, and the next sector is searched for. If a regular Data Mark is found and the SK bit is not set, the sector is read, bit 6 of ST2 (Control Mark) is set, and the read terminates with a normal termination.

WRITE DATA

The Write Data command is very similar to the Read Data command except that data is transferred from the μP to the disk rather than the other way around. If the controller detects the Write Protect signal, bit 1 of ST1 (Not Writable) is set and an abnormal termination is indicated.

WRITE DELETED DATA

This command is the same as the Write Data command except a Deleted Data Mark is written at the beginning of the Data Field instead of the normal Data Mark.

READ A TRACK

This command is similar to the Read Data command except for the following. The controller starts at the index hole and reads the Data Fields in their physical order, not their logical order. The controller still does a comparison of the Address Field information with the data programmed in the Command Phase and will set bit 2 of ST1 (No Data) if the comparison fails. If there is a CRC error in the Address Field or the Data Field, the read will continue.

The command will terminate when it has read the number of sectors programmed in the EOT parameter.

READ ID

This command will cause the controller to read the first Address Field that it finds. The Result Phase will contain the header bytes that are read. There is no data transfer during the Execution Phase of this command.

FORMAT A TRACK

This command will format one track on the disk. After the index hole is detected, data patterns are written on the disk including all gaps, address marks, Address Fields, and Data Fields. The exact details of the number of bytes for each field is controlled by the parameters given in the Command Phase. The Data Field consists of the Fill Byte specified in the Command Phase repeated to fill the entire sector.

To allow for flexible formatting, the μP must supply the four Address Field bytes (track, head, sector, size) for each sector formatted during the Execution Phase. In other words, as the controller formats each sector, it will request four bytes through either DMA requests or interrupts. This allows for non-sequential sector interleaving. Some typical values for the Address Field bytes are shown in Table III.

The Format command terminates when the index hole is detected a second time.

Command Descriptions (Continued)

TABLE III

Mode	Actual Sector Size	# Bytes per Sector Code	EOT (Hex)	Gap (Hex)	Format Gap (Hex)
8" DRIVES					
FM	128 bytes/sec	0	1A	07	1B
	256	1	0F	0E	2A
	512	2	08	1B	3A
	1024	3	04	47	8A
	2048	4	02	C8	FF
	4096	5	01	C8	FF
MFM	256	1	1A	0E	36
	512	2	0F	1B	54
	1024	3	08	35	74
	2048	4	04	99	FF
	4096	5	02	C8	FF
	8192	6	01	C8	FF
5.25" DRIVES					
FM	128	0	12	07	09
	128	0	10	10	19
	256	1	08	18	30
	512	2	04	46	87
	1024	3	02	C8	FF
	2048	4	01	C8	FF
MFM	256	1	12	0A	0C
	256	1	10	20	32
	512	2	08	2A	50
	1024	3	04	80	F0
	2048	4	02	C8	FF
	4096	5	01	C8	FF
3.5" DRIVES					
FM	128	0	0F	07	1B
	256	1	09	0E	2A
	512	2	05	1B	3A
MFM	256	1	0F	0E	36
	512	2	09	1B	54
	1024	3	05	35	74

Note: Format Gap is the gap length used only for the Format Command.

SCAN COMMANDS

The Scan commands allow data read from the disk to be compared against data sent from the μP. There are three conditions to choose from: Equal, Less than or Equal, Greater than or Equal. An FF(hex) from either the disk or the μP is used as a don't care byte that will always match true. After each sector is read, if the desired condition has not been met, the next sector is read. The next sector is defined as the current logical sector number plus Sector Step Process (SSP). The Scan command will continue until the scan condition has been met, or if the End of Track has been reached, or if TC is asserted.

It is important to program the End of Track to be a multiple of the Sector Step Process. Otherwise, the end of the track will not be detected.

The result of the command is shown in Table IV.

TABLE IV

Status Reg. 2			
Command	**Bit 2**	**Bit 3**	**Conditions**
Scan Equal	0	1	Disk = μP
	1	0	Disk ≠ μP
Scan Low or Equal	0	1	Disk = μP
	0	0	Disk < μP
	1	0	Disk > μP
Scan High or Equal	0	1	Disk = μP
	0	0	Disk > μP
	1	0	Disk < μP

SEEK

There are two ways to move the disk drive head to the desired track number. Method One is to enable the Implied Seek Mode. This way each individual Read or Write command will automatically move the head to the track specified in the command.

Method Two is using the Seek command. During the Execution Phase of the Seek command, the track number to seek to is compared with the present track number and a step pulse is produced to move the head one track closer to the desired track number. This is repeated at the rate specified by the Specify command until the head reaches the correct track. At this point, an interrupt is generated and a Sense Interrupt command is required to clear the interrupt.

During the Execution Phase of the Seek command the only indication via software that a Seek command is in progress is bits 0–3 (Drive Busy) of the Main Status register. Bit 4 of the Main Status register (Controller Busy) is not set. This allows a Seek command to be issued for another drive even while the first drive is still seeking. This is called a Multiple Seek. All four drives may be seeking at the same time. No other command except the Seek command or the Sense Interrupt command should be issued while a Seek command is in progress.

RECALIBRATE

The Recalibrate command is very similar to the Seek command. It is used to step a drive head out to track zero. Step pulses will be produced until the track zero signal from the drive becomes true. If the track zero signal does not go true before 256 step pulses are issued, an error is generated. If the extended track range mode is enabled, an error is not generated until 4096 pulses are issued.

Multiple recalibrations may be issued just like the Seek command for more than one drive. No other command except the Recalibrate command or the Sense Interrupt command should be issued while a Recalibrate Command is in progress.

Command Descriptions (Continued)

SENSE INTERRUPT STATUS

An interrupt is generated by the controller when any of the following conditions occur:

1. Upon entering the Result Phase of:
   a. Read Data command
   b. Read Deleted Data command
   c. Write Data command
   d. Write Deleted Data command
   e. Read a Track command
   f. Read ID command
   g. Format command
   h. Scan commands
2. During data transfers in the Execution Phase while in the Non-DMA mode
3. Ready Line from a drive changes state
4. Seek or Recalibrate command termination

An interrupt generated for reasons 1 or 2 above occurs during normal command operations and is easily discernible by the μP. During an execution phase in Non-DMA Mode, bit 5 (Execution Mode) in the Main Status register is set to 1. Upon entering Result Phase this bit is set to 0. Reasons 1 and 2 do not require the Sense Interrupt Status command. The interrupt is cleared by reading or writing data to the Controller.

Interrupts caused by reasons 3 and 4 are identified with the aid of the Sense Interrupt Status command. This command resets the interrupt when the command byte is written. Use bits 5, 6, and 7 of ST0 to identify the cause of the interrupt as shown in Table V.

TABLE V

Status Register 0			Cause
Seek End	Interrupt Code		
Bit 5	Bit 6	Bit 7	
0	1	1	Ready Line Changed State
1	0	0	Normal Seek Termination
1	1	0	Abnormal Seek Termination

Issuing a Sense Interrupt Status command without an interrupt pending is treated as an invalid command.

If the extended track range mode is enabled, a third byte should be read in the Result Phase which will indicate the four most significant bits of the Present Track Number. Otherwise, only two bytes should be read.

SPECIFY

The Specify command sets the initial values for three internal timers. The timers have two modes as shown in Table VI. The timer modes are programmed from the Mode command. Mode One should be used if the controller is being interfaced to 8" drives. This mode could be interpreted as defining the timers to be Head Load and Head Unload timers. Mode Two should be used if a drive motor is being turned off and on as in a 5¼" drive. The Motor On Time defines the time between when the Motor On signal going high and the start of the Read/Write operation starts. The Motor Off Time defines the time from the end of the Execution Phase of one of the Read/Write commands to the Motor Off state. The Step Rate Time defines the time interval between adjacent step pulses during a Seek, Implied Seek, or Recalibrate command.

The times stated in the table are affected by the Data Rate pin. If the pin is high, the table is correct. If the pin is low, the times in the table should be doubled.

The choice of DMA or Non-DMA operation is made by the Non-DMA bit. When this bit = 1 then Non-DMA mode is selected, and when this bit = 0, the DMA mode is selected.

TABLE VI

Timer	Mode 1		Mode 2	
	Value	Min/Max	Value	Min/Max
Step Rate Time	(16-N) ms	1–16 ms	(16-N) ms	1–16 ms
Motor Off Time	N × 16 ms	0–240 ms	N × 512 ms	0–7.68 Sec
Motor On Time	N × 2 ms	0–254 ms	N × 32 ms	0–4.064 Sec

Note 1: Double all times if Data Rate pin is low

Note 2: Based on 8 MHz clock

SENSE DRIVE STATUS

This two-byte command obtains the status of the Drives. Status Register 3 is returned in the result phase and contains the drive status.

MODE

This command is used to select the special features of the controller. The "*" indicates the default which is used after any reset. This reset default has been chosen to be compatible with the μPD765A.

* **TMR = 0**	Timers for motor on and motor off are defined for Mode 1 (see Specify command).
TMR = 1	Timers for motor on and motor off are defined for Mode 2 (see Specify command).
* **IDX = 0**	The controller will format tracks with the Index Address Mark included. (Exact IBM standard.)
IDX = 1	The controller will format tracks without including the Index Address Mark. This may increase the storage capability of each track. (Sony standard.)
* **IPS = 0**	The implied seek bit in the commands is ignored.
IPS = 1	The implied seek bit in the commands is enabled so that if the bit is set in the command, the Seek will be performed automatically.
* **MMX = 0**	Only the motor of the drive selected is assumed on. This means that whenever a new drive is selected, the Motor On time is enabled.
MMX = 1	All drive motors are assumed to be on when any motor is turned on. This eliminates the Motor On time when switching from one drive to another before the Motor Off time expires.
LOW = 00*	Completely disables the low power mode.
01	Go into low power mode automatically after all drive motors are off.
10	Not used.
11	Go into low power mode now.

Command Descriptions (Continued)

POL = 0 Disable polling mode.

*** POL = 1** Enable polling mode.

*** HDR = 0** The standard header format is used with 8 bits for the track number.

HDR = 1 Header format is the same as above but there are 12 bits of track number. The most significant bits of the track number are in the upper four bits of the head number byte.

Low Current and Precomp Track #

Track number to enable the low current output pin and to enable write precompensation. When the controller is writing to track numbers with a value which is less than this value, write precompensation is disabled and the low current output pin is disabled. Default is track zero.

*** DRE = 0** The data rate is determined by the Data Rate pin.

DRE = 1 The data rate is determined by the bits set in the DATA RT positions.

ANR Abort Not Ready. The state of this bit, in conjunction with the POL bit, determines how the controller treats the drive ready signal. Table VII describes how the controller responds to the ready pin depending on the state of POL and ANR. Default is ANR = 1.

*** EL = 0** Early/Late. The drive select signals are demultiplexed onto four output signals.

EL = 1 The drive select signals are multiplexed onto DR0 and DR1, while precomp Early appears on DR2, and precomp Late appears on DR3. This is only needed if the clock frequency is greater than 10 MHz which is above the range of the internal write precompensation circuit. This is only used in the DP8474 and should always be set low in the DP8472.

*** WLD = 0** Wildcard character. An FF(hex) from either the μP or the disk is interpreted as a wildcard character that will always match true.

WLD = 1 The Scan commands do not recognize FF(hex) as a wildcard character.

Head Settle Time allowed for head to settle after an Implied Seek. Time = N $\times$ 16 ms, (0–240 ms). Default is 64 ms. (Based on 8 MHz clock).

DTS = 0 Disable internal Data Separator. Decoded clock signal goes to the Data Window input. Decoded data goes to the Read Data input.

*** DTS = 1** Enable internal Data Separator. The encoded data read from the disk goes to the Read Data input.

*** 1/4 = 0** Quarter period delay line is always set to the reference period through the secondary PLL.

1/4 = 1 Quarter period delay line follows the changes in frequency of the data by following the same bias voltage as the Main VCO.

DATA RT Data Rate. After a Reset, the data rate is determined by the Data Rate pin. If the DRE bit is set, the data rate is determined by these two bits as shown in Table VIII.

Write Pre-Comp The value of these four bits determines the amount of write precompensation used when writing to the disk drive as shown in Table IX. The default value is based on the data rate used and can be found in Table VIII.

TABLE VII. Ready Polling Mode Table

POL	ANR	Comments
0	0	Do not check ready ever. All commands execute whether ready is true or not.
0	1	Wait up to 5 revs for ready. If ready is not true, wait up to 5 disk revolutions and if still not ready, abort the command.
1	0	Poll, but do not abort. All commands execute whether ready is true or not, but polling continues and ready change INTs are still issued.
1	1	Poll, and abort immediately command if drive not ready (default).

TABLE VIII. Data Rate Table

DT RT	FM		MFM		Default Precomp	
	f = 8 MHz	Variable f	f = 8 MHz	Variable f	f = 8 MHz	Variable f
00	125 kbits/sec	1X	250 kbits/sec	2X	250 ns	0111
01	250 kbits/sec	2X	500 kbits/sec	4X	143 ns	0100
10	500 kbits/sec	4X	1000 kbits/sec	8X	71 ns	0010

Note: X = 10^{12} / f bits/sec, where f = clock frequency. (See Write Precompensation Table for default precomp with variable f.)

Command Descriptions (Continued)

TABLE IX. Write Precompensation Table

Write Pre-Comp	Amount of Pre-Compensation	
	f = 8 MHz	Variable f
0 0 0 0	none	0X
0 0 0 1	36 ns	1X
0 0 1 0	71 ns	2X
0 0 1 1	107 ns	3X
0 1 0 0	143 ns	4X
0 1 0 1	179 ns	5X
0 1 1 0	214 ns	6X
0 1 1 1	250 ns	7X
1 0 0 0	286 ns	8X
1 0 0 1	321 ns	9X
1 0 1 0	357 ns	10X
1 0 1 1	393 ns	11X
1 1 0 0	429 ns	12X
1 1 0 1	464 ns	13X
1 1 1 0	Illegal	
1 1 1 1	Default	

Note: X = 2/7f ns, where f = clock frequency.

SET TRACK

This command can be used to read or write any value to or from any internal register. For the typical application this command could be used to inspect or change the value of the internal Present Track Register. This could be useful for disk mistracking errors, where the real current track could be read through the Read ID command and then the Set Track Command can set the internal present track register to the correct value. The internal register # for the least significant byte of the Present Track Register is 0C (hex). If more than 8 bits are being used for the track counter, the upper four bits can be accessed through internal register # 0D (hex).

INVALID COMMAND

If an invalid command (i.e., a command not defined) is received by the controller, then the controller terminates the command. The controller does not generate an interrupt during this condition. Bits 6 and 7 in the Main Status Register are both set to 1's indicating to the processor that the Controller is in the Result Phase and the contents of ST0 must be read. When the system reads ST0 it will find a hex 80 indicating an invalid command was received.

In some applications the user may use this command as a No-Op command to place the controller in a standby or no operation state. Simply issue an illegal command and delay reading the result phase until the controller is needed for another command. During this time, the Controller will not poll the drives.

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN})	−1.5V to V_{CC} +1.5V
DC Output voltage (V_{OUT})	−0.5V to V_{CC} +0.5V
Clamp Diode Current	±20 mA

Operating Conditions

	Min	Max	
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage	0	V_{CC}	V
Operating Temperature Range (T_A)	−40	+85	°C

DC Electrical Characteristics V_{CC} = 5V ±10%, unless otherwise specified.

Symbol	Parameter	Conditions	T_A = −40 to +85°C Limits	Units
V_{IH}	Minimum High Level Input Voltage		2.0	V
V_{IL}	Maximum Low Level Input Voltage		0.8	V
I_{OH}	Max Leakage Current Disk Interface	V_{IN} = V_{IH} or V_{IL} Invert = 1	10	μA
V_{OH}	Min High Level Out Disk Interface	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 2 mA Invert = 0	3.7	V
V_{OL}	Max Low Level Out Disk Interface	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 8 mA	0.4	V
V_{OH}	Min High Level Out All Other	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 2 mA	3.7	V
V_{OL}	Max Low Level Out All Other	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 2 mA	0.4	V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND	±1.0	μA
I_{OZ}	Maximum TRI-STATE® Leakage Current	V_{OUT} = V_{CC} or GND	±5.0	μA
I_{CC}	Maximum Supply Current	V_{IN} = 3.4 or 0.8V F_{IN} = 8 MHz	40	mA
I_{CC}	Maximum Supply Current	V_{IN} = V_{CC} or GND F_{IN} = 8 MHz	20	mA
I_{CC}	Maximum Supply Current in Low Power Mode	V_{IN} = V_{CC} or GND	100	μA

AC Characteristics V_{CC} = 5V ±10%, C = 150 pF, unless otherwise specified.

Symbol	Parameter	T = −40° to +85°C		Units
		Min	Max	
f	Crystal Frequency	4	10	MHz
f	External Clock Frequency	1	20	MHz
D Rate	Data Rate	125	2500	Kbit/s
μP READ TIMING				
t_{AR}	Address to Read Strobe	0		ns
t_{RA}	Address Hold from Read Strobe	0		ns
t_{RR}	Read Strobe Width	130		ns
t_{RD}	Read Strobe to Data		130	ns
t_{DF}	Data Hold from Read Strobe	5	45	ns
t_{RI}	Clear INT from Read Strobe		130	ns

AC Characteristics V_{CC} = 5V ±10%, C = 150 pF, unless otherwise specified. (Continued)

Symbol	Parameter	T = −40° to +85°C		Units
		Min	Max	
μP WRITE TIMING				
t_{AW}	Address to Write Strobe	0		ns
t_{WA}	Address Hold from Write Strobe	0		ns
t_{WW}	Write Strobe Width	80		ns
t_{DW}	Data Setup to End of Write Strobe	65		ns
t_{WD}	Data Hold from Write Strobe	5		ns
t_{WI}	Clear INT from Write Strobe		130	ns
DMA TIMING				
t_{AM}	End of DRQ from DAK		75	ns
t_{MA}	DAK from DRQ	0		ns
t_{AA}	DAK Pulse Width	130		ns
t_{MRW}	DRQ to End of Read Strobe	130		ns
t_{MW}	DRQ to Read Strobe	0		ns
t_{MR}	DRQ to Write Strobe	0		ns
DRIVE SEEK TIMING				
t_{DIR}	Direction from Drive Select	6		μs
t_{DRH}	Drive Select Hold from End of Step	3		μs
t_{DST}	Step from Direction	6		μs
t_{DH}	Direction Hold from End of Step	3		μs
t_{STR}	Step Rate	Programmable		
t_{STP}	Step Pulse Width	4		μs

AC Characteristics V_{CC} = 5V ±10%, C = 150 pF, unless otherwise specified. (Continued)

Symbol	Parameter	T = −40° to +85°C		Units
		Min	Max	
DISK READ TIMING				
t_{DRS}	Drive Select from Motor On	15		µs
t_{SR}	Read Data from Last Step Pulse	75		µs
t_{RD}	Read Data Width	20		ns
t_{RDS}	Data Window Setup Time	10		ns
t_{RDH}	Data Window Hold Time			ns
DISK WRITE TIMING				
t_{WGS}	Write Data from Write Gate	200 ns	2 bit times	
t_{WGH}	Write Gate Hold from Write Data	1 bit time	2 bit times	

PLL Characteristics

Symbol	Parameter	Valve	
$K\phi_{High}$	Phase Comparator & Charge Pump Gain Constant. High Gain Mode. (Note 1)	$\frac{5\ V_{REF}}{2\pi R}$	Typ
V_{REF}	Voltage at Set Pump Current Pin	1.2V	Typ
$K\phi_{Low}$	Low Gain Mode (Note 1)	$\frac{2.5\ V_{REF}}{2\pi R}$	Typ
K_{VCO}	Gain of VCO (Note 2)	5/N MRad/S/V	Typ
f_{VCO}	Center Frequency of VCO	f/2	Typ
t_{JITTER}	Maximum Tolerance of Bit Jitter (Note 3)	$\frac{(0.95)}{4 \times DR}$	Typ
$t_{POWER\ ON}$	Time from Full V_{CC} Power to Guaranteed Functionally	50 ms	Max

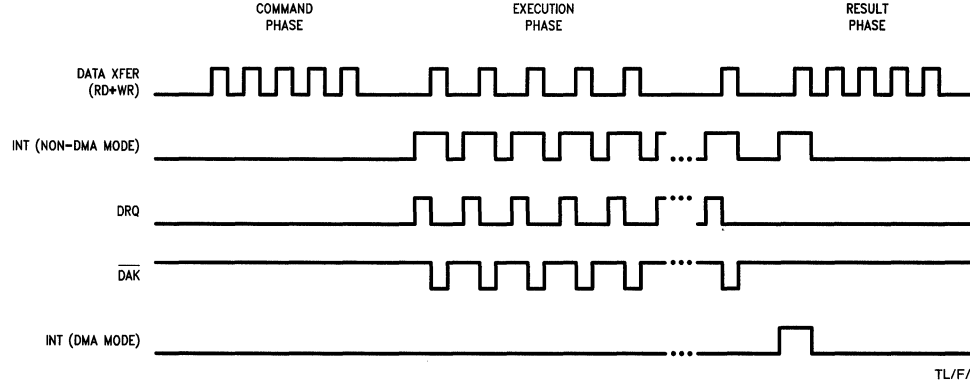
Note 1: R = Pump current set resistor (8k–20k).

Note 2: N = # of VCO cycles per bit.

Note 3: DR = Data Rate.

Timing Waveforms

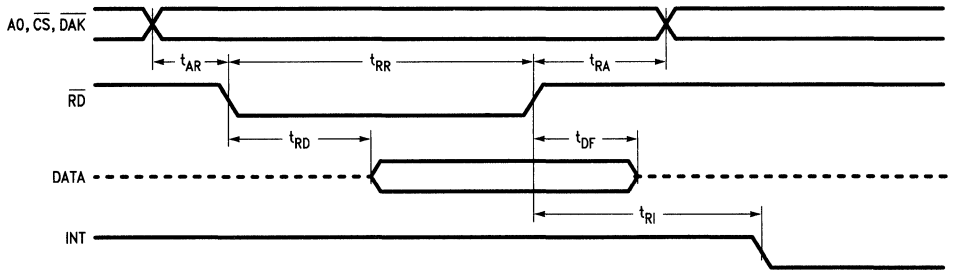
Interrupt and DMA Timing



TL/F/8592–4

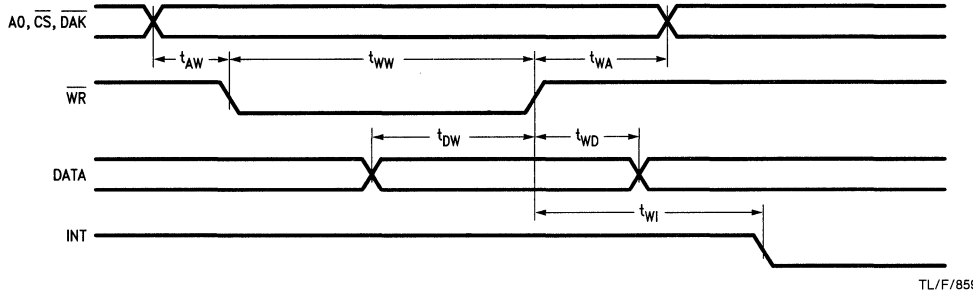
Timing Waveforms (Continued)

μP Read Timing



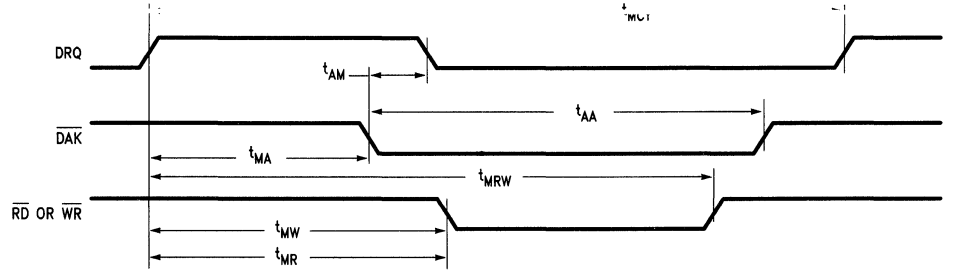
TL/F/8592–5

μP Write Timing



TL/F/8592–6

DMA Timing

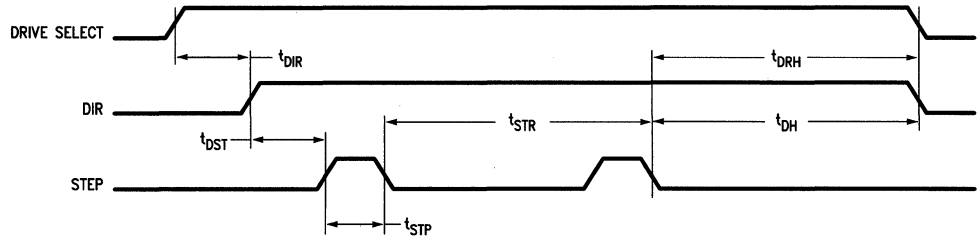


TL/F/8592–7

6

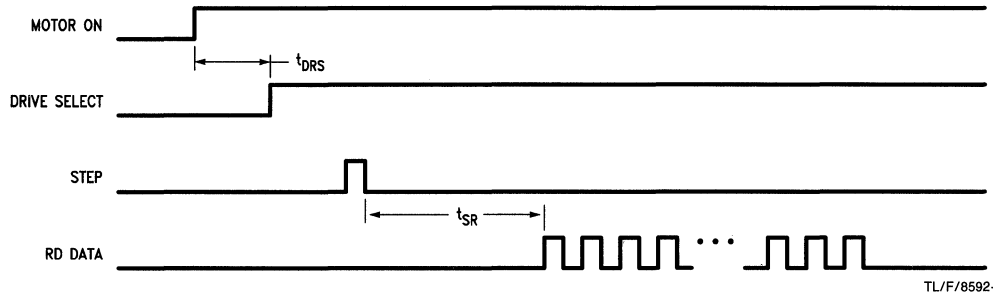
Timing Waveforms (Continued)

Drive Seek



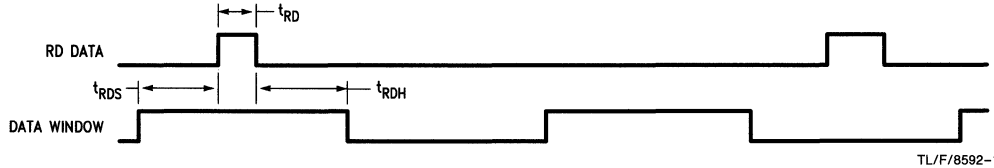
TL/F/8592–8

Disk Read



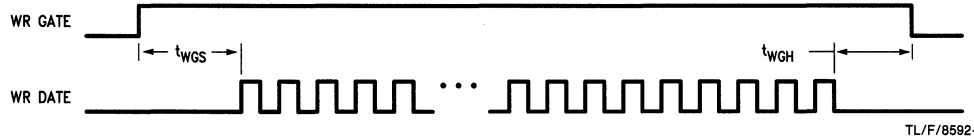
TL/F/8592–9

Read Data



TL/F/8592–11

Disk Write



TL/F/8592–10

Section 7
DRAM Interface

Section 7 Contents

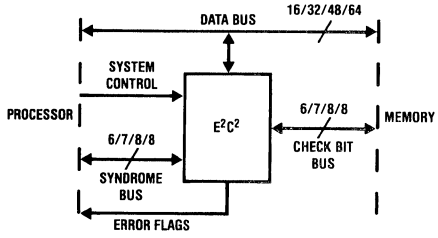
DP8400-2/NS32800-2 E^2C^2 Expandable Error Checker/Corrector 7-3
DP8402A/NS32802A 32-bit Parallel Error Detection and Correction Circuits (EDAC's) 7-37
DP8409A/NS32809A Multi-Mode Dynamic RAM Controller/Driver 7-53
DP84412/NS32812 Dynamic RAM Controller Interface Series Circuit for the Series 32000 CPU 7-75
DP8417/NS32817/DP8418/NS32818/DP8419/NS32819/DP8419X/NS32819X 64k, 256k Dynamic RAM Controller/Drivers 7-88
DP84512/NS32813 Dynamic RAM Controller Interface Circuit for the NS32332 7-113
DP8428/NS32828/DP8429/NS32829 1 Megabit High Speed Dynamic RAM Controller/Driver 7-114

National Semiconductor Corporation

DP8400-2—E²C² Expandable Error Checker/Corrector

General Description

The DP8400-2 Expandable Error Checker and Corrector (E^2C^2) aids system reliability and integrity by detecting errors in memory data and correcting single or double-bit errors. The E^2C^2 data I/O port sits across the processor-memory data bus as shown, and the check bit I/O port connects to the memory check bits. Error flags are provided, and a syndrome I/O port is available. Fabricated using high speed Schottky technology in a 48-pin dual-in-line package, the DP8400-2 has been designed such that its internal delay times are minimal, maintaining maximum memory performance.



TL/F/6899–1

For a 16-bit word, the DP8400-2 monitors data between the processor and memory, with its 16-bit bidirectional data bus connected to the memory data bus. The DP8400-2 uses an encoding matrix to generate 6 check bits from the 16 bits of data. In a WRITE cycle, the data word and the corresponding check bits are written into memory. When the same location of memory is subsequently read, the E^2C^2 generates 6 new check bits from the memory data and compares them with the 6 check bits read from memory to create 6 syndrome bits. If there is a difference (causing some syndrome bits to go high), then that memory location contains an error and the DP8400-2 indicates the type of error with 3 error flags. If the error is a single data-bit error, the DP8400-2 will automatically correct it.

The DP8400-2 is easily expandable to other data configurations. For a 32-bit data bus with 7 check bits, two DP8400-2s can be used in cascade with no other ICs. Three DP8400-2s can be used for 48 bits, and four DP8400-2s for 64 data bits, both with 8 check bits. In all these configurations, single and double-error detection and single-error correction are easy to implement.

When the memory is more unreliable, or better system integrity is preferred, then in any of these configurations, double-error correction can be performed. One approach requires a further memory WRITE-READ cycle using complemented data and check bits from the DP8400-2. If at least one of the two errors is a hard error, the DP8400-2 will correct both errors. This implementation requires no more memory check bits or DP8400-2s than the single-error correct configurations.

The DP8400-2 has a separate syndrome I/O bus which can be used for error logging or error management. In addition, the DP8400-2 can be used in BYTE-WRITE applications (for up to 72 data bits) because it has separate byte controls for the data buffers. In 16 or 32-bit systems, the DP8400-2 will generate and check system byte parity, if required, for integrity of the data supplied from or to the processor. There are three latch controls to enable latching of data in various modes and configurations.

Operational Features

- Fast single and double-error detection
- Fast single-error correction
- Double-error correction after catastrophic failure with no additional ICs or check bits
- Functionally expandable to 100% double-error correct capability
- Functionally expandable to triple-error detect
- Directly expandable to 32 bits using 2 DP8400-2s only
- Directly expandable to 48 bits using 3 DP8400-2s only
- Directly expandable to 64 bits using 4 DP8400-2s only
- Expandable to and beyond 64 bits in fast configuration with extra ICs
- 3 error flags for complete error recording
- 3 latch enable inputs for versatile control
- Byte parity generating and checking
- Separate byte controls for outputting data in BYTE-WRITE operation
- Separate syndrome I/O port accessible for error logging and management
- On-chip input and output latches for data bus, check bit bus and syndrome bus
- Diagnostic capability for simulating check bits
- Memory check bit bus, syndrome bus, error flags and internally generated syndromes available on the data bus
- Self-test of E^2C^2 on the memory card under processor control
- Full diagnostic check of memory with the E^2C^2
- Complete memory failure detectable
- Power-on clears data and syndrome input latches

Timing Features

16-BIT CONFIGURATION

WRITE Time: 29 ns from data-in to check bits valid

DETECT Time: 21 ns from data-in to Any Error (AE) flag set

CORRECT Time: 44 ns from data-in to correct data out

7

Timing Features (Continued)

32-BIT CONFIGURATION

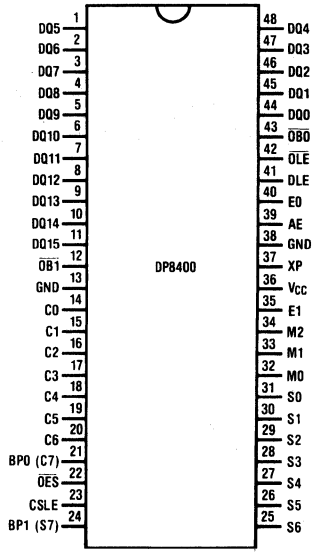
WRITE Time: 49 ns from data-in to check bits valid

DETECT Time: 46 ns from data-in to Any Error (AE) flag set

CORRECT Time: 84 ns from data-in to correct data out

DP8400-2 Connection Diagram

Dual-In-Line Package

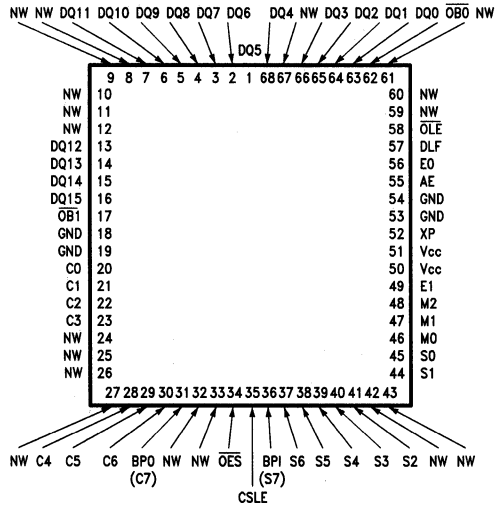


TL/F/6899–2

Top View

Order Number DP8400V-2, DP8400N-2, or DP8400D-2
See NS Package V68, N48A or D48A

Chip Carrier Package



TL/F/6899–36

Top View

Pin Descriptions

Pin #	Description
1	DQ5
2	DQ6
3	DQ7
4	DQ8
5	DQ9
6	DQ10
7	DQ11
13	DQ12
14	DQ13
15	DQ14
16	DQ15
17	$\overline{OBI}$
18	GND
19	GND
20	C0
21	C1
22	C2
23	C3
28	C4
29	C5
30	C6
31	BP0 (C7)
34	$\overline{OES}$
35	CSLE
36	BPI (S7)
37	S6
38	S5
39	S4
40	S3
41	S2
44	S1
45	S0
46	M0
47	M1
48	M2
49	E1
50	V_{CC}
51	V_{CC}
52	XP
53	GND
54	GND
55	AE
56	E0
57	DLF
58	$\overline{OLE}$
62	$\overline{OB0}$
63	DQ0
64	DQ1
65	DQ2
66	DQ3
68	DQ4

Note: Pins 8, 9, 10, 11, 12, 24, 25, 26, 27, 32, 33, 42, 43, 59, 60, 61, and 67 are all **NW**.

Pin Definitions See *Figure 1* for abbreviations

V_{CC}, GND, GND: 5.0V ±5%. The 3 supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. Also there are two ground pins to reduce the low-level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 16 data bits change in the same direction simultaneously. A recommended solution would be a 1 μF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected close to pins 36 and 38 to reduce lead inductance.

DQ0–DQ15: Data I/O port. 16-bit bidirectional data bus which is connected to the input of DIL0 and DIL1 and the output of DOB0 and DOB1, with DQ8–DQ15 also to CIL.

C0–C6: Check-bit I/O port. 7-bit bidirectional bus which is connected to the input of the CIL and the output of the COB. COB is enabled whenever M2 is low.

S0–S6: Syndrome I/O port. 7-bit bidirectional bus which is connected to the input of the SIL and the output of the SOB.

DLE: Input data latch enable. When high, DIL0 and DIL1 outputs follow the input data bus. When low, DIL0 and DIL1 latch the input data.

CSLE: Input check bit and syndrome latch enable. When high, CIL and SIL follow the input check and syndrome bits. When low, CIL and SIL latch the input check and syndrome bits. If $\overline{OES}$ is low, SIL remains latched.

$\overline{OLE}$: Output latch enable. $\overline{OLE}$ enables the internally generated data to DOL0, and DOL1, COL and SOL when low, and latches when high.

XP: Multi-expansion, which feeds into a three-level comparator. With XP at 0V, only 6 or 7 check bits are available for expansion up to 40 bits, allowing byte parity capability. With XP open or at V_{CC}, expansion beyond 40 bits is possible, but byte parity capability is no longer available. When XP is at V_{CC}, CG6 and CG7, the internally generated upper two check bits, are set low. When XP is open, CG6 and CG7 are set to word parity.

BP0 (C7): When XP is at 0V, this pin is byte-0 parity I/O. In the Normal WRITE mode, BP0 receives system byte-0 parity, and in the Normal READ mode outputs system byte-0 parity. When XP is open or at V_{CC}, this pin becomes C7 I/O, the eighth check bit for the memory check bits, for 48-bit expansion and beyond.

BP1 (S7): When XP is at 0V, this pin is byte-1 parity I/O. In the Normal WRITE mode, BP1 receives system byte-1 parity, and in the Normal READ mode outputs system byte-1 parity. When XP is open or at V_{CC}, this pin becomes S7 I/O, the eight syndrome bit for 48-bit expansion and beyond.

AE: Any error. In the Normal READ mode, when low, AE indicates no error and when high, indicates that an error has occurred. In any WRITE mode, AE is permanently low.

E0: In the Normal READ mode, E0 is high for a single-data error, and low for other conditions. In the Normal WRITE mode, E0 becomes $\overline{PE0}$ and is low if a parity error exists in byte-0 as transmitted from the processor.

E1: In the Normal READ mode, E1 is high for a single-data error or a single check bit error, and low for no error and double-error. In the Normal WRITE mode, E1 becomes $\overline{PE1}$ and is low if a parity error exists in byte-1 as transmitted from the processor.

$\overline{OB0}$, $\overline{OB1}$: Output byte-0 and output byte-1 enables. These inputs, when low, enable DOL0 and DOL1 through DOB0 and DOB1 onto the data bus pins DQ0–DQ7 and DQ8–DQ15. When $\overline{OB0}$ and $\overline{OB1}$ are high the DOB0, DOB1 outputs are TRI-STATE®.

$\overline{OES}$: Output enables syndromes. I/O control of the syndrome latches. When high, SOB is TRI-STATE and external syndromes pass through the syndrome input latch with CSLE high. When $\overline{OES}$ is low, SOB is enabled and the generated syndromes appear on the syndrome bus, also CSLE is inhibited internally to SIL.

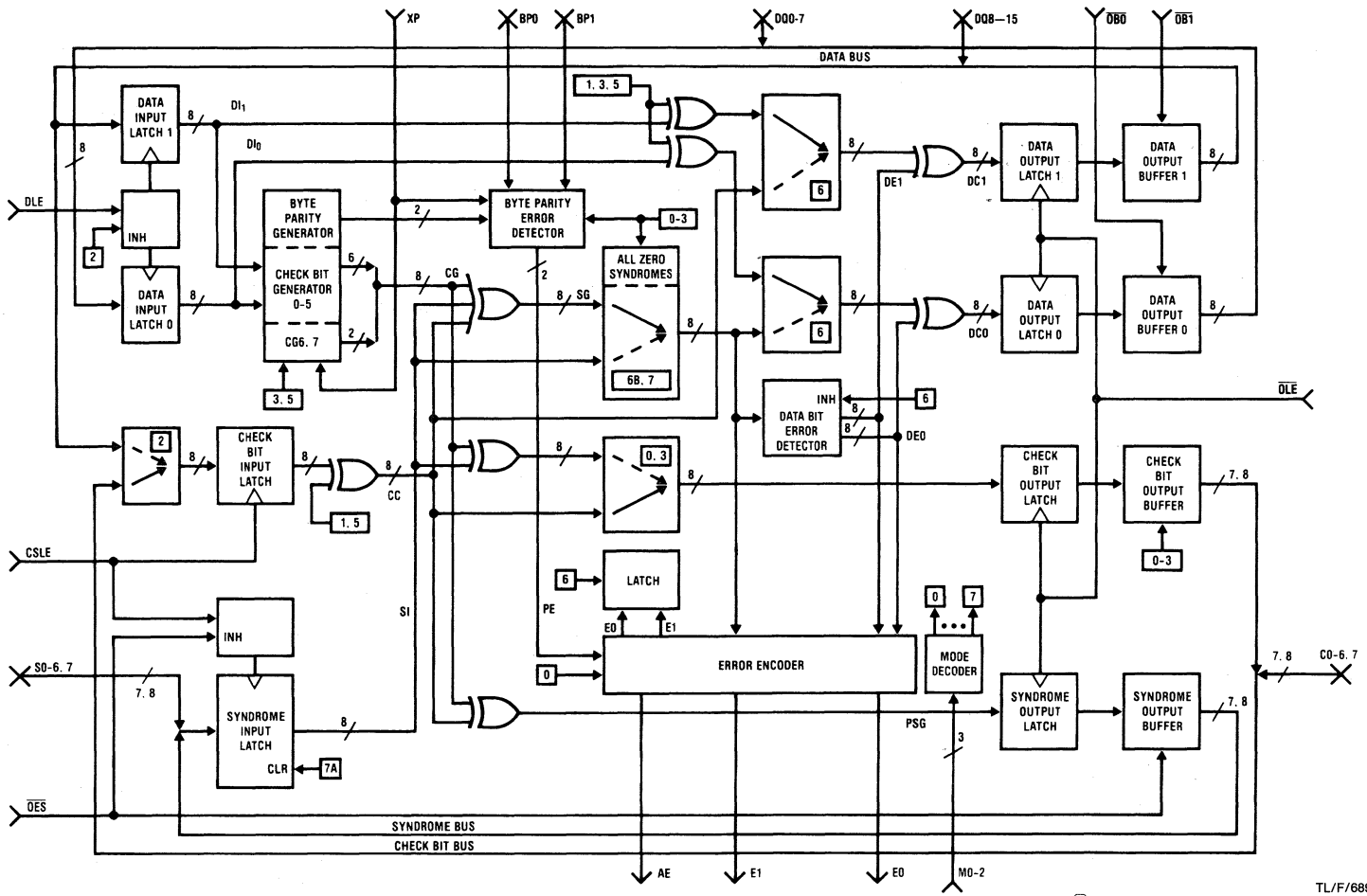
M0, M1, M2: Mode control inputs. These three controls define the eight major operational modes of the DP8400-2. Table III depicts the modes.

System Write (*Figure 2a*)

The Normal WRITE mode is mode 0 of Table III. Referring to the block diagram in *Figure 9a* and the timing diagram of *Figure 9b*, the 16 bits of data from the processor are enabled into the data input latches, DIL0 and DIL1, when the input data latch enable (DLE) is high. When this goes low, the input data is latched. The check bit generator (CG) then produces 6 parity bits, called check bits. Each parity bit monitors different combinations of the input data-bits. In the 16-bit configuration, assuming no syndrome bits are being fed in from the syndrome bus into the syndrome input latch, the 6 check bits enter the check bit output latch (COL), when the output latch enable $\overline{OLE}$ is low, and are latched in when $\overline{OLE}$ goes high. Whenever M2 (READ/$\overline{WRITE}$) is low, the check bit output buffer COB always enables the COL contents onto the external check bit bus. Also the data error decoder (DED) is inhibited during $\overline{WRITE}$ so no correction can take place. Data output latches DOL0 and DOL1, when enabled with $\overline{OLE}$, will therefore see the contents of DIL0 and DIL1. If valid system data is still on the data bus, a memory WRITE will write to memory the data on the data bus and the check bits output from COB. If the system has vacated the data bus, output enables ($\overline{OB0}$ and $\overline{OB1}$) must be set low so that the original data word with its 6 check bits can be written to memory.

System Read

There are two methods of reading data: the error monitoring method (*Figure 2b*), and the always correct method (*Figure 2c*). Both require fast error detection, and the second, fast correction. With the first method, the memory data is only monitored by the E^2C^2, and is assumed to be correct. If there is an error, the Any Error flag (AE) goes high, requiring further action from the system to correct the data. With the always correct method, the memory data is assumed to be possibly in error. Memory data is removed and the corrected, or already correct, data is output from the E^2C^2 by enabling $\overline{OB1}$ and $\overline{OB0}$. To detect an error (referring to *Figures 10a* and *10b*) first DLE and CSLE go high to enter data bits and check bits from memory into DIL0, DOL1 and CIL. The 6 check bits generated in CG from DIL0 and DOL1 are then compared with CIL to generate syndromes on the internal syndrome bus (SG). Any bit or bits of SG that go high indicate an error to the error encoder (EE).



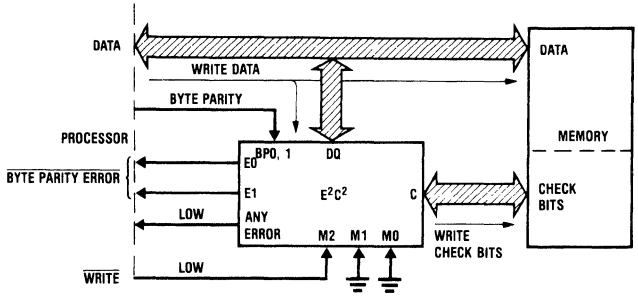
TL/F/6899–3

DIL	Data Input Latch	SG	Syndrome Generator	SOL	Syndrome Output Latch
CG	Check Bit Generator	DED	Data Error Detector	DOB0, 1	Data Output Buffer Bytes 0, 1
CIL	Check Bit Input Latch	DE0, 1	Data Error Bytes 0, 1	COB	Check Bit Output Buffer
CC	Check Bit Complementor	PE	Parity Error	SOB	Syndrome Output Buffer
SIL	Syndrome Input Latch	DOL0, 1	Data Output Latch Bytes 0, 1	EE	Error Encoder
PSG	Partial Syndrome Generator	COL	Check Bit Output Latch	DC0, 1	Data Corrector Bytes 0, 1

[n] mode of operation signifies active signal

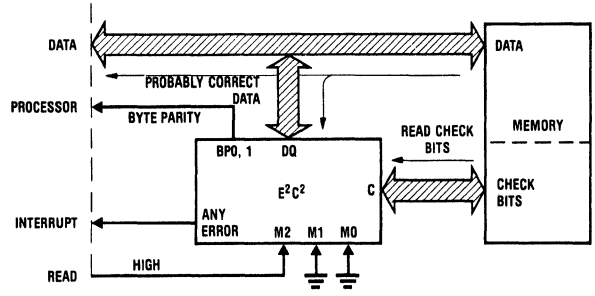
FIGURE 1. DP8400-2 Block Diagram

System Diagrams—Modes of Operation



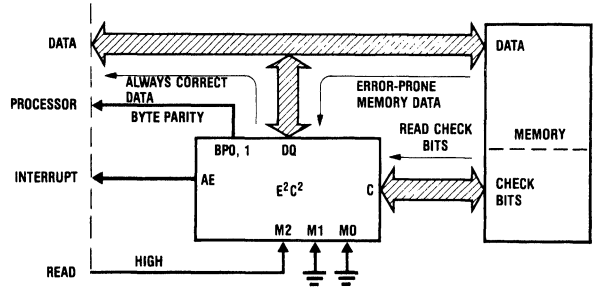
TL/F/6899–4

FIGURE 2a. Normal WRITE Mode with E^2C^2



TL/F/6899–5

FIGURE 2b. Normal READ Mode, Error Monitoring Method with E^2C^2



TL/F/6899–6

FIGURE 2c. Normal READ Mode, Always Correct Method with E^2C^2

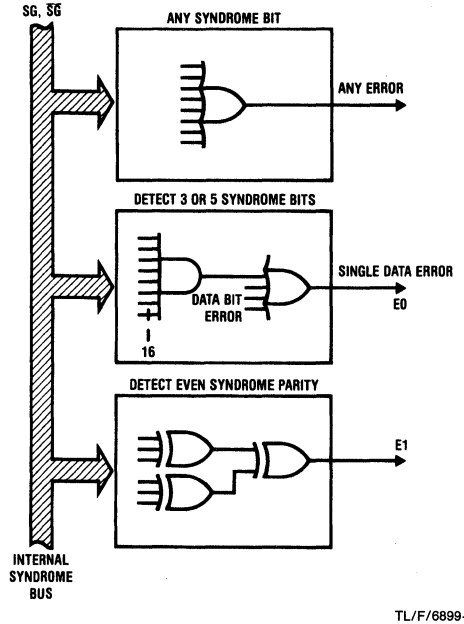
System Read (Continued)

If data correction is required $\overline{OB0}$ and $\overline{OB1}$ must be set low (after memory data has been disabled) to enable data output buffers DOB0 and DOB1. The location of any data bit error is determined by the data error decoder (DED), from the syndrome bits. The bit in error is complemented in the DOL for correction. The other 15 bits from DED pass the DIL contents directly to the DOL, so that DOL now contains corrected data.

Error Determination

The three error flags, for a 16-bit example, are decoded from the internally generated syndromes as shown in *Figure 3*. First, if any error has occurred, the generated check bits will be different from the memory check bits, causing some of the syndrome bits to go high. By OR-ing the syndrome bits, the output will be an indication of any error.

If there is a single-data error, then (from the matrix in Table IV) it can be seen that any data error causes either 3 or 5 syndrome bits to go high. 16 AND gates decode which bit is in error and the bit in error is XOR-ed with the corresponding bit of the DIL to correct it, whereas the other 15 decoder outputs are low, causing the corresponding 15 bits in DIL to transfer to DOL directly. DOL now contains corrected data. The 16 AND gate outputs are OR-ed together causing E0 to go high, so that E0 is the single-data-error indication. If the error is a double-error, then either 2, 4 or 6 of the syndrome bits will be high. The syndromes for two errors (including one or two check bit errors) are the two sets of syndromes for each individual error bit, XOR-ed together. By performing a parity check on the syndrome bits, flag E1 will indicate even/odd parity. If there is still an error, but it is not one of these errors, then it is a detectable triple-bit error. Some triple-bit errors are not detectable as such and may be interpreted as single-bit errors and falsely corrected as single-data errors. This is true for all standard ECC circuits using a Modified Hamming-code matrix. The DP8400-2 is capable, with its Rotational Syndrome Word Generator matrix, of determining all triple-bit errors using twice as many DP8400-2s and twice as many check bits.



TL/F/6899–7

FIGURE 3. Error Encoder

Error Flags

Three error flags are provided to allow full error determination. Table I shows the error flag outputs for the different error types in Normal READ mode. If there is an error, then ANY ERROR will go high, at a time t_{DEV} (*Figure 10b*) after data and check bits are presented to the DP8400-2. The other two error flags E0 and E1 become valid t_{DE0} and t_{DE1} later.

The error flags differentiate between no error, single check bit error, single data-bit error, double-bit error. Because the DP8400-2 can correct double errors, it is important to know that two errors have occurred, and not just a multiple-error indication. The error flags will remain valid as long as DLE and CSLE are low, or if DLE is high, and data and check bits remain valid.

Byte Parity Support

Some systems require extra integrity for transmission of data between the different cards. To achieve this, individual byte parity bits are transmitted with the data bits in both directions. The DP8400-2 offers byte parity support for up to 40 data bits. If the processor generates byte parity when transferring information to the memory, during the WRITE cycle, then each byte parity bit can be connected to the corresponding byte parity I/O pin on the DP8400-2, either BP0 or BP1. The DP8400-2 develops its own internal byte parity bits from the two bytes of data from the processor, and compares them with BP0 and BP1 using an exclusive-OR for both parities. The output of each exclusive-OR is fed to the error flags E0 and E1 as $\overline{PE0}$ and $\overline{PE1}$, so that a byte parity error forces its respective error flag low, as in Table II. These flags are only valid for the Normal WRITE (mode 0) and XP at 0V. The DP8400-2 checks and generates even byte parity.

When transferring information from the memory to the processor, the DP8400-2 receives the memory data, and outputs the corresponding byte parity bits on BP0 and BP1 to the processor. The processor block can then check data integrity with its own byte parity generator. If in fact memory data was in error, the DP8400-2 derives BP0 and BP1 from the memory input data, and not the corrected data, so when corrected data is output from the DP8400-2, the processor will detect a byte parity error.

During the read mode, DP8400-2 corrects single data bit error and also its parity.

TABLE I. Error Flags After Normal Read (Mode 4)

AE	E1	E0	Error Type
0	0	0	No error
1	1	0	Single check bit error
1	1	1	Single-data error
1	0	0	Double-bit error
All Others			Invalid conditions

TABLE II. Error Flags after Normal Write (Mode 0)

AE	E1 ($\overline{PE1}$)	E0 ($\overline{PE0}$)	Error Type
0	1	1	No parity error
0	1	0	Parity error, byte 0
0	0	1	Parity error, byte 1
0	0	0	Parity error, bytes 0, 1

TABLE III. DP8400-2 Modes of Operation

Mode	M2 (R/W)	M1	M0	$\overline{OES}$	Operation
0	0	0	0	X	Normal WRITE DIL → DOL, CG → COL → COB
1	0	0	1	X	Complement WRITE $\overline{DIL}$ → DOL, $\overline{CIL}$ → COL → COB
2	0	1	0	X	Diagnostic WRITE, DLE inhibited DQ8–DQ15 ⊕ CG → SOL → SOB DQ8–DQ15 → CIL → COL → COB
3	0	1	1	X	Complement data-only WRITE $\overline{DIL}$ → DOL, (CG0, 1, 4, 5, $\overline{CG2}$, $\overline{CG3}$) → COL → COB
4	1	0	0	X	Normal READ DIL ⊕ DE → DOL, CIL → COL
5	1	0	1	X	Complement READ $\overline{DIL}$ ⊕ DE → DOL, $\overline{CIL}$ → COL
6A	1	1	0	0	READ generated syndromes, check bit bus, error flags, SG0–SG6 → DQ0–DQ6, CIL0–CIL6 → DQ8–DQ14, E1 → DQ7, E0 → DQ15
6B	1	1	0	1	READ syndrome bus, check bit bus, error flags, SIL0–SIL6 → DQ0–DQ6, CIL0–CIL6 → DQ8–DQ14, E1 → DQ7, E0 → DQ15
7A	1	1	1	0	Generated syndromes replace with zero 0 → SIL → SG, CIL → COL, DIL ⊕ DE → DOL
7B	1	1	1	1	Generated syndromes replace SIL → SG, CIL → COL, DIL ⊕ DE → DOL

TABLE IV. Data-In To Check Bit Generate, Or Data Bit Error To Syndrome-Generate Matrix (16-Bit Configuration)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	1	1	1	1	1	1	0	1	2	3	4	5	6	7	8	9	DQ0-15
GENERATE CHECK BITS →																	
0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0
1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	2*
3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	3*
4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	4
5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	5
GENERATED SYNDROMES																	GENERATED CHECK BITS
	4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	F	0
	3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1	1

HEXADECIMAL EQUIVALENT OF SYNDROME BITS

*C2, C3 generate odd parity

TL/F/6899–8

Modes of Operation

There are three mode-control pins, M2, M1 and M0, offering 8 major modes of operation, according to Table III.

M2 is the READ/$\overline{\text{WRITE}}$ control. In normal operation, mode 0 is Normal WRITE and mode 4 is Normal READ. By clamping M0 and M1 low, and setting M2 low during WRITE and high during READ, the DP8400-2 is very easy to use for normal operation. The other modes will be covered in later sections.

16-BIT CONFIGURATION

The first two rows on top of the check bit generate matrix (Table IV) indicate the data position of DQ0 to DQ15. The left side of the matrix, listed 0 to 5, corresponds to syndromes S0 to S5. S0 is the least significant syndrome bit. There are two rows of hexadecimal numbers below the matrix. They are the hex equivalent of the syndrome patterns. For example, syndrome pattern in the first column of the matrix is 001011. Its least significant four bits (0010) equal hexadecimal 4, and the remaining two bits (11) equal hexadecimal 3.

Check bit generation is done by selecting different combinations of data bits and generating parities from them. Each row of the check bit generate matrix corresponds to the generation of a check bit numbered on the right hand side of the matrix, and the ones in that row indicate the selection of data bits.

The following are the check bit generate equations for 16-bit wide data words:

$$CG0 = DQ2 \oplus DQ3 \oplus DQ4 \oplus DQ5 \oplus DQ6 \oplus DQ7 \oplus DQ9 \oplus DQ10 \oplus DQ11 \oplus DQ13 \oplus DQ14 \oplus DQ15$$

$$CG1 = DQ3 \oplus DQ6 \oplus DQ8 \oplus DQ9 \oplus DQ11 \oplus DQ13 \oplus DQ14 \oplus DQ15$$

$$*CG2 = DQ0 \oplus DQ3 \oplus DQ4 \oplus DQ8 \oplus DQ10 \oplus DQ12 \oplus DQ13 \oplus DQ14 \oplus DQ15 \oplus 1$$

$$*CG3 = DQ1 \oplus DQ2 \oplus DQ7 \oplus DQ8 \oplus DQ9 \oplus DQ10 \oplus DQ12 \oplus DQ14 \oplus DQ15 \oplus 1$$

$$CG4 = DQ0 \oplus DQ1 \oplus DQ5 \oplus DQ7 \oplus DQ8 \oplus DQ11 \oplus DQ13 \oplus DQ15$$

$$CG5 = DQ0 \oplus DQ1 \oplus DQ2 \oplus DQ4 \oplus DQ5 \oplus DQ6 \oplus DQ8 \oplus DQ12 \oplus DQ13 \oplus DQ14$$

*CG2 and CG3 are odd parities.

The following error map (Table V) depicts the relationship between all possible error conditions and their associated syndrome patterns. For example, if a syndrome pattern is S0–5 = 111101, data bit 14 is in error.

Figure 4 shows how to connect one DP8400-2 in a 16-bit configuration, in order to detect and correct single or double-bit errors. For a Normal WRITE, processor data is presented to the DP8400-2, where it is fed through DIL0 and DIL1 to the check bit generator. This generates 6 parity bits from different combinations of data bits, according to Table IV. The numbers in the row below the table are the hexadecimal equivalent of the column bits (with bits 6, 7 low). A "1" in any row indicates that the data bit in that column is connected to the parity generator for that row. For example, check bit 1 generates parity from data bits 3, 6, 8, 9, 11, 13, 14, and 15.

Check bits 0, 1, 4, 5, and 6 generate even parity, and check bits 2 and 3 generate odd parity. This is done to insure that a total memory failure is detected. If all check bits were even parity, then all zeroes in the data word would generate all check bits zero and a total memory failure would not be detected when a memory READ was performed. Now all-zero-data bits produce C2 and C3 high and a total memory failure will be detected. When reading back from the same location, the memory data bits (possibly in error) are fed to the same check bit generator, where they are compared to the memory check bits (also possibly in error) using 6 exclusive-OR gates. The outputs of the XORs are the syndrome bits, and these can be determined according to Table IV for one data bit error. For example, an error in bit 2 will produce the syndrome word 101001 (for S5 to S0 respectively). The syndrome word is decoded by the error encoder to the error flags, and the data-error decoder to correct a single data bit error. Assuming the memory data has been latched in the DIL, by making DLE go low, memory data can be disabled. Then by setting $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ low, corrected data will appear on the data bus. The syndromes are available as outputs on pins S0–5 when $\overline{\text{OES}}$ is low. It is also possible to feed in syndromes to SIL when $\overline{\text{OES}}$ is high and CSLE goes high. This can be useful when using the Error Management Unit shown in *Figure 4*. C6 and S6 are not used for 16 bits. It is safe therefore to make C6 appear low, through a 2.7 kΩ resistor to ground. The same applies for S6 if syndromes are input to the DP8400-2. If $\overline{\text{OES}}$ is permanently low, S6 may be left open.

Any 16-bit memory correct system using the DP8400-2 without syndrome inputs must keep the $\overline{\text{OES}}$ pin grounded, then all the syndrome I/O pins may be left open. The reason for this is that the DP8400-2 resets the syndrome input latch at power up. If the $\overline{\text{OES}}$ pin is grounded, the syndrome input latch will remain reset for normal operations.

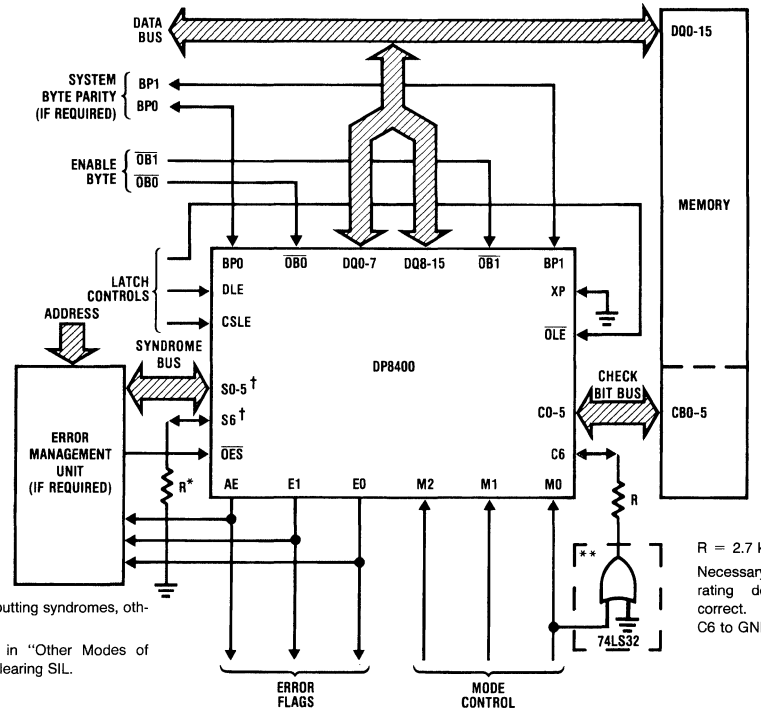
The parameter t_{NMR} (see *Figure 10b*), new mode recognized time, is measured from M2 (changing from READ to WRITE) to the valid check bits appearing on the check bit bus, provided the $\overline{\text{OLE}}$ was held low.

The parameter t_{MCR} (see *Figure 10b*), mode change recognized time, is measured from M2 (changing from WRITE to

TABLE V. Syndrome Decode To Bit In Error For 16-Bit Data Word

	S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Syndrome	**S1**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Bits	**S2**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S5	**S4**																
0	0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0	1	C4	D	D	11	D	T	T	D	D	7	T	D	T	D	D	15
1	0	C5	D	D	6	D	4	T	D	D	2	T	D	12	D	D	14
1	1	D	5	T	D	0	D	D	13	1	D	D	T	D	T	8	D

NE = no error Cn = check bit n in error T = three errors detected Number = single data bit in error D = two bits in error



* Necessary when inputting syndromes, otherwise leave open.

† Refer-to-discussion in "Other Modes of Operation" under Clearing SIL.

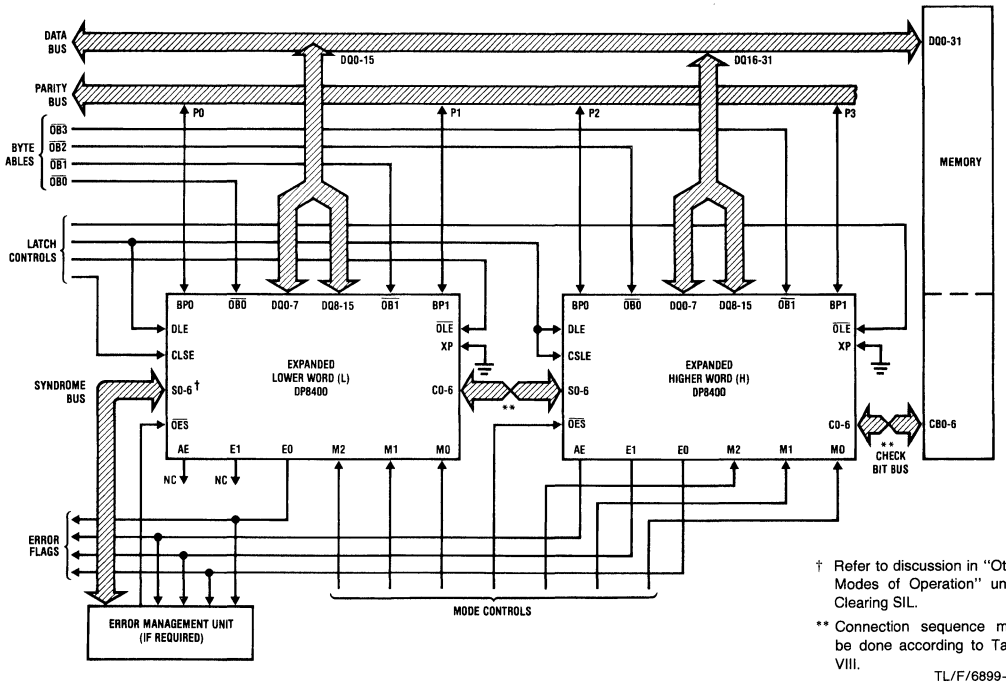
**20 ns max tpd1, 0

R = 2.7 kΩ

Necessary only when incorporating double complement correct. Otherwise connect C6 to GND through R.

TL/F/6899–9

FIGURE 4. 16-Bit Configuration Using One DP8400-2



† Refer to discussion in "Other Modes of Operation" under Clearing SIL.

** Connection sequence must be done according to Table VIII.

TL/F/6899–10

FIGURE 5. 32-Bit Error Detection and Correction

Modes of Operation (Continued)

READ) when both E1 and E2 become invalid. This is required when a memory correcting system employs the DP8400-2 with byte parity checking. The E1 and E2 pins flag the byte parity error in a memory WRITE cycle. When the DP8400-2 switches to a subsequent memory READ cycle, it requires t_{MCR} for E1 and E2 to be switched to flag any READ error(s).

Expanded Operation

32-BIT CONFIGURATION

Figure 5 shows how to connect two DP8400-2s in cascade to detect single and double-bit errors, and to correct single-data errors. The same circuit will also correct double-bit errors once a double-error has been detected, provided at least one error is a hard error. The lower chip L is in effect a slave to the higher chip H, which controls the memory check bits and error reporting. The check bit bus of L is reordered and connected to the syndrome bus of H, as shown in *Figure 5*.

In a Normal WRITE mode, referring to *Figures 13a, 13b,* and *13c*, the 7 check bits generated from the lower 16 bits (CGL) are transferred via the COL to the COB of L, provided $\overline{OLE}$ is high and M2 (R/$\overline{W}$) of L is low. These partial check bits from L then appear at SIL of H, so that with CSLE high, they combine with the 6 check bits generated in H with an overlap of one bit, to produce 7 check bits. With M2 (R/$\overline{W}$) of H low, these 7 check bits are output from COB to memory.

A READ cycle may consist of DETECT ONLY or DETECT THEN CORRECT, depending on the system approach. In both approaches, L writes its partial check bits, CGL, to H as in WRITE mode. H develops the syndrome bits from CGL, CGH and the 7 check bits read from memory in CIL. H then outputs from its error encoder (EE) if there is an error. If corrected data is required, H already knows if it has a single-data error from its syndrome bits, but if not, it must transfer partial syndromes back to L. These partial syndromes PSH, (CGH XOR-ed with CIL), are stored in SOL of H. L must therefore change modes from WRITE to READ, while H outputs the partial syndromes from its SOB by setting $\overline{OES}$ low. The partial syndromes are fed into CIL of L and XOR-ed with CGL to produce syndrome bits at SGL. The data error decoder, DED, then corrects the error in L. The DED of H will already have corrected an error in the higher 16 bits. Only one error in 32 bits can be corrected as a single-data error, the chip with no error does not change the contents of its DIL when it is enabled in DOL. Table VI shows the 3 error flags of H, which become valid during the DETECT cycle. E0 of L becomes valid during the CORRECT cycle, so that the 4 flags provide complete error reporting.

TABLE VI. Error Flags After Normal READ (32-Bit Configuration)

AE (H)	E1 (H)	E0 (H)	E0 (L)*	Error Type
0	0	0	0	No error
1	1	0	0	Single-check bit error
1	1	1	0	Single-data bit error (H)
1	1	0	1	Single-data bit error (L)
1	0	0	0	Double-bit error
All Others				Invalid conditions

*E0 (L) is valid after transfer of partial syndromes from higher to lower

Equations for 32-bit expansion:

$$t_{DCB32} = t_{DCB16} + t_{SCB16}$$

$$t_{DEV32} = t_{DCB16} + t_{SEV16}$$

$$t_{DCD32}\ (\text{High Chip}) = t_{DCB16} + t_{SCD16}$$

$$t_{DCD32}\ (\text{Low Chip}) = t_{DCB16} + t_{BR}{}^{*} + t_{CCD16}$$

*t_{BR}: Bus reversing time (25 ns)

32-BIT MATRIX

Table VII shows a 32-bit matrix using two DP8400-2s in cascade as in *Figure 5*. This is one of 12 matrices that work for 32 bits. The matrix for bits 0 to 15 (lower chip) is the matrix of Table IV for 16-bit configuration, with row 6 always "0". The matrix for bits 16 to 31 (higher chip) uses the same row combinations but interchanged, for example, the 3rd row (row 2) of L matrix is the same as the 6th row (row 5) of the H matrix. This means row 5 of H is in fact check bit 2 of H. Thus, the 6th row (row 5) combines generated check bit 5 (CG5) of L and generated check bit 2 of H. Check bit 5 of L therefore connects to the syndrome bit 2 (CG2) of H, and the composite generated check bit is written to check bit 2 of memory. Thus C2 performs a parity check on bits 0, 1, 2, 4, 5, 6, 8, 12, 13, 14, of L, and bits 16, 19, 20, 24, 26, 28, 29, 30, 31, of H. CG2 and CG3 generate odd parity, so that CG5 of L generates even parity which combines with CG2 of H generating odd parity. CG3 of L and CG3 of H both generate odd parity causing C3 to memory to represent even parity. Only 6 check bits are generated in each chip, the 7th (CG6) is always zero with XP grounded. Thus CG6 of L combines with CG0 of H so that C0 to memory is the parity of bits 18, 19, 20, 21, 22, 23, 25, 26, 27, 29, 30, 31. Similarly C6 to memory is only CG2 of L. The 7 composite generated check bits of H can now be written to memory.

When reading data and check bits from memory, CG6–CG0 of L are combined with CG6–CG0 of H in the same combination as WRITE. Memory check bits are fed into C6–C0 of H and compared with the 7 combined parity bits in H, to

TABLE VII. Data Bit Error To Syndrome-Generate Matrix (32-Bit Configuration)

	L																H																
	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	DQ0-31
0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	5
*2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
SYNDROMES *3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	3
4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	4
5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	2
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0
	4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	F	2	A	A	1	2	2	3	8	B	9	8	1	A	3	B	9	0	HEX
	3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1	3	1	4	6	6	5	4	5	3	4	6	5	2	7	6	7	1	

GENERATED CHECK BITS

TL/F/6899–11

*CG2, CG3 generate odd parity

Expanded Operation (Continued)

TABLE VIII. Check Bit Port To Syndrome Port Interconnections For Expansion To 32 Bits

		LS		LC	HS		HC		
Syndrome I/O to Management	S0	0		0	1		1	C0	Check Bit I/O to Memory
	S1	1		1	5		5	C1	
	S2	2		2	6		6	C2	
	S3	3		3	3		3	C3	
	S4	4		4	4		4	C4	
	S5	5		5	2		2	C5	
	S6	6		6	0		0	C6	

TABLE IX. Syndrome Decode To Bit In Error For 32-Bit Data Word

Syndrome Bits																	
		S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		S1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
		S2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
		S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S6	S5	S4																
0	0	0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0	0	1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0	1	0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0	1	1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
1	0	0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
1	0	1	D	27	21	D	T	D	D	T	23	D	D	T	D	T	T	D
1	1	0	D	19	20	D	T	D	D	T	26	D	D	30	D	T	T	D
1	1	1	T	D	D	29	D	T	T	D	D	31	T	D	T	D	D	T

NE = no error Cn = check bit n in error T = three errors detected
Number = single data bit in error D = two bits in error

produce 7 syndrome bits S6–S0. H can now determine if there is any error, and if it has a single-data error, it can locate it and correct it without transferring partial syndromes to L. As an example of a DETECT cycle, CG5 of L combines with CG2 of H and is compared in H with memory check bit 2.

If L is now set to mode 4, Normal READ, and $\overline{OES}$ of H is set low, the partial syndromes of H (CG6–CG0 of H XOR-ed with C6–C0 of H) are transferred and shifted to L. L receives these partial syndromes (S6–S0 of H) as check bit inputs C2, C1, C4, C3, C5, C0, C6 respectively, and compares them with CG6–CG0 respectively, to produce syndrome bits S6–S0. L now decodes these syndromes to correct any single-data error in data bits 0 to 15. For example, partial syndrome bit 2 of H combines with generated check bit 5 of L to produce syndrome bit 5 in L. An error in data bit 10 will create syndrome bits in L as 0001101 from S6–S0, and these will appear on S6–S0 of L with $\overline{OES}$ low. An error in H will appear as per the H matrix. For example, an error in bit 16 will cause S6–S0 of L to be 0110010.

If $\overline{OES}$ of L is set low, this syndrome combination appears on pins S6 to S0. For errors in bits 0 to 15, the syndrome outputs will be according to Table VII. For errors in bits 16 to 31, the syndrome outputs from L will still be according to Table VII due to the shifting of partial syndrome bits from H to L. The syndrome outputs from L are unique for each of the possible 32 bits in error.

If there is a check bit error, only one syndrome bit will be high. For example, if C5 is in error, then S1 of L will be high. For double-errors, an even number of syndrome bits will be high, derived from XOR-ing the two single-bit error syndromes. As mentioned previously, this is only one of the 12 approaches to connecting two chips for 32 bits, 6 of which are mirror images.

Table VIII depicts the exact connection for 32-bit expansion. LS equals syndrome bits of L. LC equals check bits of L. HS equals syndrome bits of H. HC equals check bits of H. Syndrome bits S0 to S6 of L are connected to system syndrome bits S0 to S6. LC and HS columns are lined together showing the check bit port of L connected to the syndrome port of H in the exact sequence as shown in Table VIII. For example, check bit C0 of L is connected to the syndrome bit S1 of H, and check bit C6 of L is connected to the syndrome bit S0 of H. Check bits of H are connected to the system check bits in the order shown. Check bit C1 of H is connected to the system check bit C0.

EXPANSION FOR DATA WORDS REQUIRING 8 CHECK BITS

For 16-bit and 32-bit configurations, XP is set permanently low. In 48-bit or 64-bit configurations, XP is either set permanently to V_{CC} or left open, according to Table X, to provide 8 check bits and syndrome bits.

TABLE X. XP: Expansion Status

XP	Status	Data Bus
0V	BP0 and BP1 are byte parity I/O CG6 = 0	< 40 Bits
Open	No byte parity I/O, CG6 and CG7 = word parity	≥ 40 Bits
V_{CC}	No byte parity I/O, CG6 and CG7 = 0	≥ 40 Bits

48-BIT EXPANSION

Three DP8400-2s are required for 48 bits, with the higher chip using all 8 of its check bits to the memory. No byte parity is available for 48 to 64 bits. XP of all three chips must be at V_{CC}. The three chips are connected in cascade as in

Expanded Operation (Continued)

TABLE XI. Check Bit Port To Syndrome Port Interconnections For Expansion To 48 Bits

		LL S		LL C	LH S		LH C	HL S		HL C		
Syndrome I/O to Management	S0	0		0	1		1	6		6	C0	Check Bit I/O to Memory
	S1	1		1	5		5	1		1	C1	
	S2	2		2	6		6	4		4	C2	
	S3	3		3	3		3	7		7	C3	
	S4	4		4	4		4	2		2	C4	
	S5	5		5	2		2	3		3	C5	
	S6	6		6	0		0	5		5	C6	
	S7	7		7	7		7	0		0	C7	

For example: S0 of LL is connected to system syndrome S0. C0 of LL is connected to S1 of LH. C1 of LH is connected to S6 of HL. C6 of HL is connected to system check bit C0.

TABLE XII. Syndrome Decode To Bit In Error For 48-Bit Data Word

			S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Syndrome			S1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Bits			S2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
			S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S7	S6	S5	S4																
0	0	0	0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0	0	0	1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0	0	1	0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0	0	1	1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
0	1	0	0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
0	1	0	1	D	27	21	D	32	D	D	T	23	D	D	T	D	T	T	D
0	1	1	0	D	19	20	D	33	D	D	T	26	D	D	30	D	T	T	D
0	1	1	1	44	D	D	29	D	T	40	D	D	31	T	D	T	D	D	T
1	0	0	0	C7	D	D	T	D	T	43	D	D	T	T	D	T	D	D	T
1	0	0	1	D	T	35	D	T	D	D	T	T	D	D	T	D	T	T	D
1	0	1	0	D	T	41	D	39	D	D	T	T	D	D	T	D	T	T	D
1	0	1	1	42	D	D	T	D	T	47	D	D	T	T	D	T	D	D	T
1	1	0	0	D	T	38	D	37	D	D	T	T	D	D	T	D	T	T	D
1	1	0	1	36	D	D	T	D	T	45	D	D	T	T	D	T	D	D	T
1	1	1	0	34	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
1	1	1	1	D	T	46	D	T	D	D	T	T	D	D	T	D	T	T	D

NE = no error Cn = check bit n in error T = three errors detected
Number = single data bit in error D = two bits in error

Figure 6, but with the HH chip removed. The error flags are as Table XV, but with AE (HH) and E1 (HH) becoming AE (HL) and E1 (HL), and E0 (HH) removed.

48-BIT MATRIX

The matrix for 48 bits is that for 64 bits shown (in Table XVI) but only using bits 0 to 47. This is one of many matrices for 48-bit expansion using the basic 16-bit matrix. The matrix shown uses 2 zeroes for CG6 and CG7, for all three chips, with XP set to V_{CC}. Other matrices may use CG6 and CG7 as word parity with XP open.

64-BIT EXPANSION

There are two basic methods of expansion to 64 bits, both requiring 8 check bits to memory, and four DP8400-2s. One is the cascade method of *Figure 6*, requiring no extra ICs. With this method partial check bits have to be transferred through three chips in the WRITE or DETECT mode, and partial syndrome bits transferred back through three chips in CORRECT mode. This method is similar to *Figure 5*, 32-bit approach. The connections between the check bit bus and syndrome bus for each of the chip pairs are shown in Table XIII.

The error flags of HH are valid during the DETECT cycle as in Table XV, and the other error flags are valid during the CORRECT cycle.

A faster method of 64-bit expansion shown in *Figure 7* requires a few extra ICs, but can WRITE in 50 ns, DETECT in 42 ns or DETECT THEN CORRECT in 90 ns. In the WRITE mode, all four sets of check bits are combined externally in the 8 74S280 parity generators. These generate 8 composite check bits from the system data, which are then enabled to memory. In the DETECT mode, again 8 composite check bits are generated, from the memory data this time, and compared with the memory check bits to produce 8 external syndrome bits. These syndrome bits may be OR-ed to determine if there is any error. By making the 74S280 outputs $\overline{\text{SYNDROMES}}$, then any bit low causes the 74S30 NAND gate to go high, giving any error indication. To correct the error, these syndrome bits are fed re-ordered into SIL of each DP8400-2 now set to mode 7B. This enables the syndromes directly to SG and then DED of each chip. One chip

Expanded Operation (Continued)

will output corrected data, while the other three output non-modified data (but still correct).

Equations for fast 64-bit expansion:

$t_{DCB64} = t_{DCB16} + t_{pd}$ (74S280) + t_{pd} (74S240)

$t_{DEV64} = t_{DCB16} + t_{pd}$ (74S280) + t_{pd} (74S30)

$t_{DCB64} = t_{DCB16} + t_{pd}$ (74S280) + t_{pd} (74ALS533) + t_{SCD16}

64-BIT MATRIX

With the 64-bit matrix shown in Table XVI, it is necessary to set at least one chip with CG6, CG7 non-zero. The highest chip, connected to data bits 48 to 63, has XP set open, so that its CG6 and CG7 are word parity. The syndrome word of the highest chip will now have either 5 or 7 syndrome bits high, but inside the chip CG6 and CG7 remove two of these in a READ so that the chip sees the normal 3 or 5 syndrome bits.

TABLE XIII. Check Bit Port To Syndrome Port Interconnections For Expansion To 64 Bits

		LL S		LL C	LH S		LH C	HL S		HL C	HH S		HH C		
Syndrome I/O to Management	S0	0		0	1		1	6		6	7		7	C0	Check Bit I/O to Memory
	S1	1		1	5		5	1		1	0		0	C1	
	S2	2		2	6		6	4		4	1		1	C2	
	S3	3		3	3		3	7		7	2		2	C3	
	S4	4		4	4		4	2		2	3		3	C4	
	S5	5		5	2		2	3		3	4		4	C5	
	S6	6		6	0		0	5		5	5		5	C6	
	S7	7		7	7		7	0		0	6		6	C7	

For example: S0 of LL is connected to system syndrome S0. C0 of LL is connected to S1 of LH. C1 of LH is connected to S6 of HL. C6 of HL is connected to S7 of HH. C7 of HH is connected to system check bit C0.

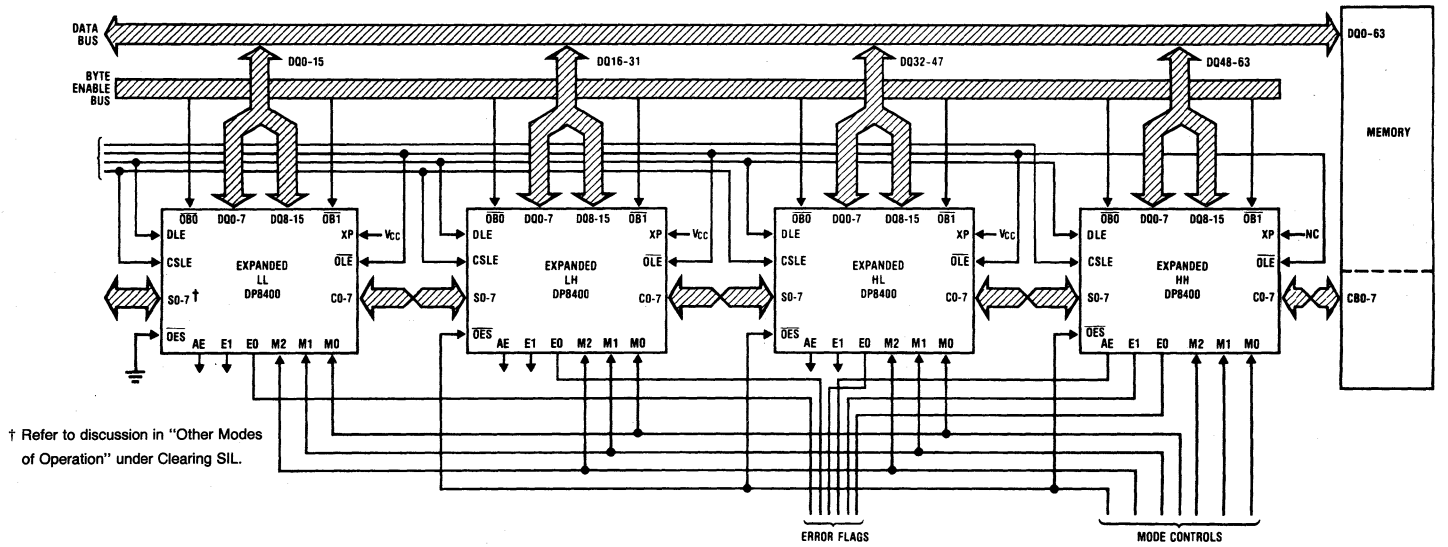
TABLE XIV. Syndrome Decode To Bit In Error For 64-Bit Data Word

Syndrome Bits																			
			S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			S1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
			S2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
			S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S7	S6	S5	S4																
0	0	0	0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0	0	0	1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0	0	1	0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0	0	1	1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
0	1	0	0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
0	1	0	1	D	27	21	D	32	D	D	T	23	D	D	T	D	T	T	D
0	1	1	0	D	19	20	D	33	D	D	T	26	D	D	30	D	T	T	D
0	1	1	1	44	D	D	29	D	T	40	D	D	31	T	D	T	D	D	T
1	0	0	0	C7	D	D	T	D	T	43	D	D	T	T	D	T	D	D	51
1	0	0	1	D	T	35	D	T	D	D	57	T	D	D	58	D	T	T	D
1	0	1	0	D	T	41	D	39	D	D	59	T	D	D	T	D	T	T	D
1	0	1	1	42	D	D	55	D	T	47	D	D	T	T	D	T	D	D	63
1	1	0	0	D	T	38	D	37	D	D	54	T	D	D	52	D	T	T	D
1	1	0	1	36	D	D	50	D	T	45	D	D	60	T	D	T	D	D	62
1	1	1	0	34	D	D	53	D	T	T	D	D	48	T	D	T	D	D	61
1	1	1	1	D	49	46	D	T	D	D	T	T	D	D	T	D	56	T	D

NE = no error
Cn = check bit n in error
T = three errors detected
Number = single data bit in error
D = two bits in error

TABLE XV. Error Flags After Normal READ (Any 64-Bit Configuration)

AE (HH)	E1 (HH)	E0 (HH)	E0 (HL)	E0 (LH)	E0 (LL)	Error Type
0	0	0	0	0	0	No error
1	1	0	0	0	0	Single-check bit error
1	1	1	0	0	0	Single-data bit error in HH
1	1	0	1	0	0	Single-data bit error in HL
1	1	0	0	1	0	Single-data bit error in LH
1	1	0	0	0	1	Single-data bit error in LL
1	0	0	0	0	0	Double-error



TL/F/6899–12

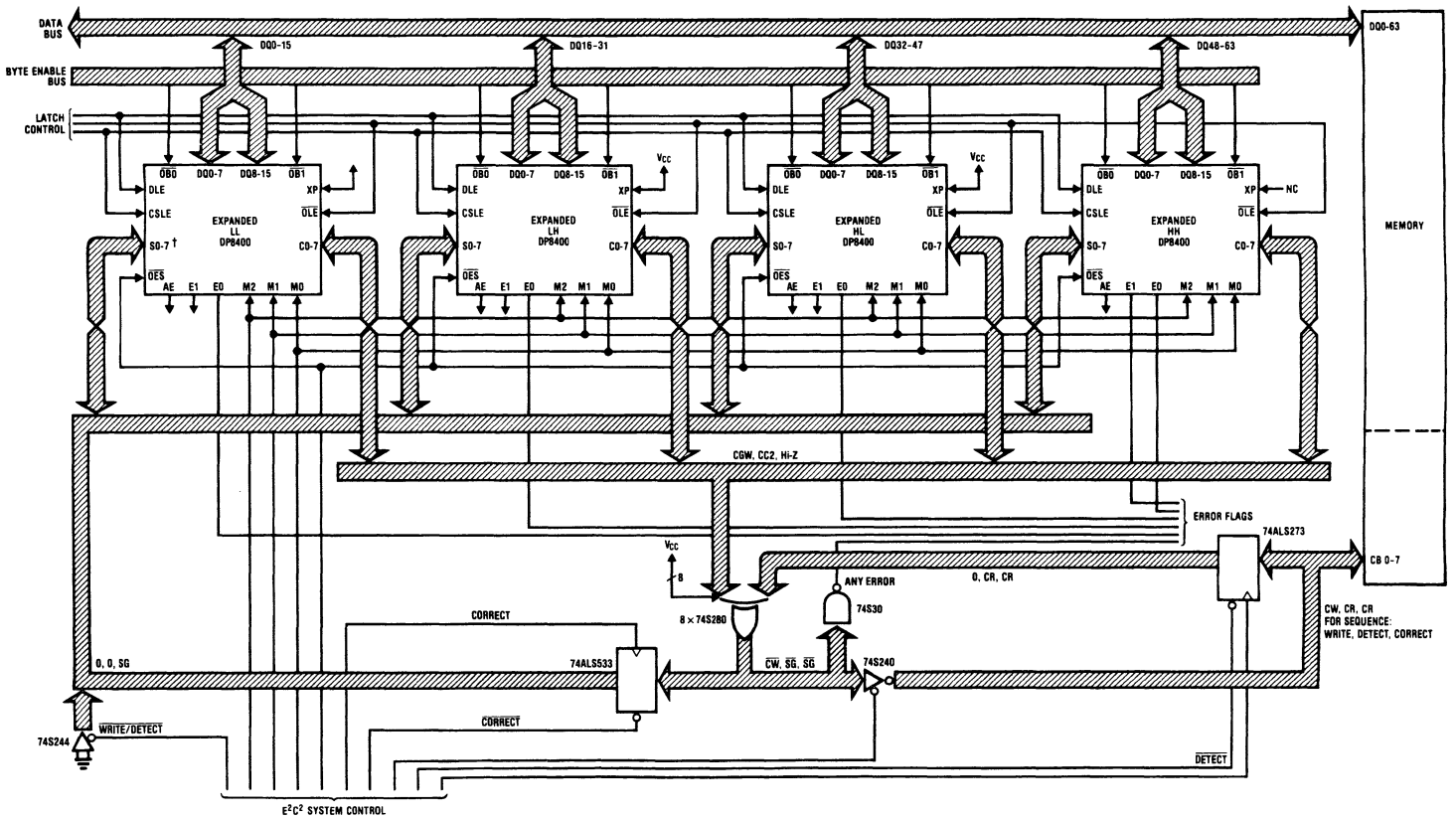
FIGURE 6. Cascade Expansion Using No Extra ICs (64-Bit Configuration)

TABLE XVI. Data Bit Error To Syndrome-Generate Matrix (64-Bit Configuration)

SYNDROMES	LL	LH	HL	HH	
0	0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1	0 0 0 1 0 0 1 0 1 1 1 0 1 0 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	7
1	0 0 0 1 0 0 1 0 1 1 1 0 1 0 1 1 1	1 1 1 0 1 1 1 1 0 1 0 0 0 1 1 1 0	0 0 0 1 0 0 1 0 1 1 1 0 1 0 1 1 1	0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1	0
2	1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 1	0 0 0 1 0 0 1 0 1 1 1 0 1 0 1 1 1	1
3	0 1 1 0 0 0 0 1 1 1 1 0 1 0 1 1	0 1 1 0 0 0 0 1 1 1 1 0 1 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 1	2
4	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 1	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 1	1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 1	0 1 1 0 0 0 0 1 1 1 1 0 1 0 1 1	3
5	1 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0	1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 1	0 1 1 0 0 0 0 1 1 1 1 0 1 0 1 1	1 1 0 0 0 1 0 1 1 0 0 1 0 1 0 1	4
6	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1	1 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0	1 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0	5
7	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	6
	4 8 9 7 5 1 3 9 E B D 3 C 7 F F	2 A A 1 2 2 3 8 B 9 8 1 A 3 B 9	4 4 0 2 0 4 2 4 6 2 0 6 0 6 2 6	9 1 3 F B 3 7 3 D 7 B 7 9 F F F	0 HEX
	3 3 2 0 2 3 2 1 3 0 0 1 2 3 2 1	3 1 4 6 6 5 4 5 3 4 6 5 2 7 6 7	5 6 E 9 D C C A 7 A B 8 7 D F B	E F D 8 C E C B F 9 9 A D E D B	1

DQ0-63 | GENERATED CHECK BITS

TL/F/6899–13



TL/F/6899–14

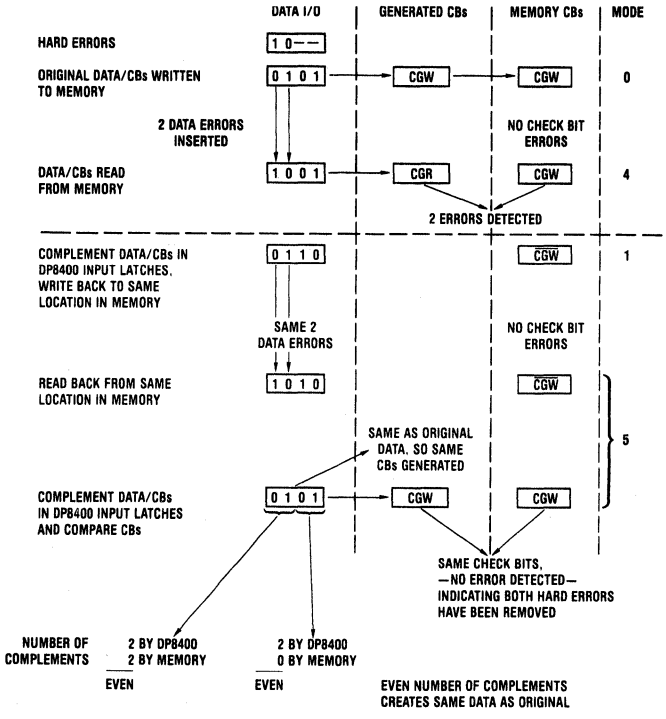
FIGURE 7. Parallel Expansion (Fast 64-Bit Configuration)

Other Modes Of Operation

DOUBLE ERROR CORRECTION, USING THE DOUBLE-COMPLEMENT APPROACH

The DP8400-2 can be made to correct two errors, using no extra ICs or check bits, if at least one of the two errors detected is a hard error. This does require an extra memory WRITE and READ. Nevertheless, if a permanent failure exists, and an additional error occurs (creating two errors), both errors can be corrected, thereby saving a system crash.

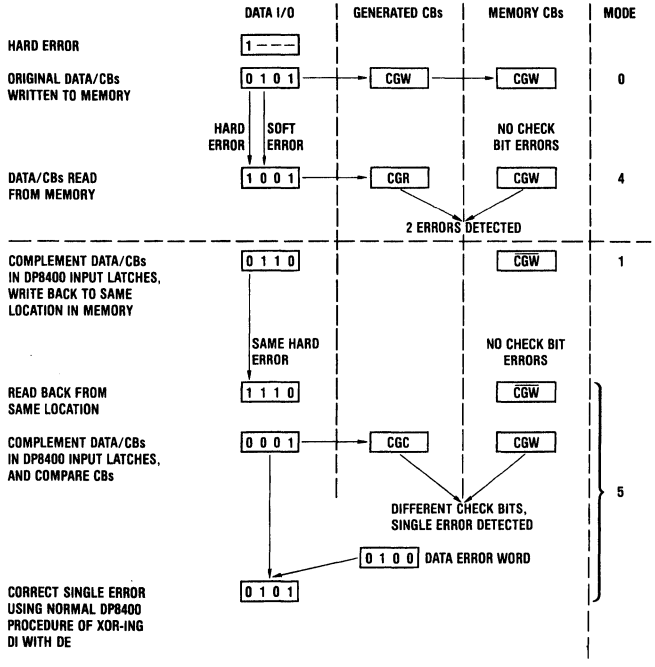
Once a double error has been detected, the system puts the DP8400-2 in COMPLEMENT mode by setting M0 high. First a WRITE cycle is required and M2 is set low, putting the chip in mode 1, Table III, (COMPLEMENT WRITE), so that the contents of DIL are complemented into DOL, and the contents of CIL complemented into COL. $\overline{OB0}$ and $\overline{OB1}$ are set low so that complemented data and check bits can be written back to the same location of memory. Writing back complemented data to a location with a hard error forces the error to repeat itself. For example, if cell N of a particular location is jammed permanently high, and a low is written to it, a high will be read. However, when the data is complemented a low is again written, so that a high is read back for the second time. After a second READ (this second READ is a COMPLEMENT READ) of the location, data and check bits from the memory are recomplemented, so that bit N now contains a low. In other words, the error in bit N has corrected itself, while the other bits are true again. If there are two hard errors in a location, both are automatically corrected and the DP8400-2 detects no error on COMPLEMENT READ, as in *Figure 8a*. *Figure 8b* also shows that if one error is soft, the hard error will disappear on the second READ and the DP8400-2 corrects the soft error as a single-error. Therefore, in both cases, the DOL contains corrected data, ready to be enabled by $\overline{OB0}$ and $\overline{OB1}$. A WRITE to memory at this stage removes the complemented data written at the start of the sequence.



TL/F/6899–15

FIGURE 8a. Double Error Correct Complement Hard Error Method — 2 Hard Errors In Data Bits

Other Modes of Operation (Continued)



TL/F/6899–16

FIGURE 8b. Double Error Correct Complement Hard Error Method — 1 Hard Error, 1 Soft Error In Data Bits

The examples shown in *Figures 8a* and *8b* are for 4 data bits. This approach will work for any number of data bits, but for simplicity these examples show how complementing twice corrects two errors in the data bits. The double COMPLEMENT approach also works for any two errors providing at least one is hard. In other words, one data-bit error and one check bit error, or two check bit errors are also corrected if one or both are hard. At the end of the COMPLEMENT READ cycle, the error flags indicate whether the data was correctable or not, as shown in Table XVII. If both the errors were soft, then the data was not correctable and the error flags indicate this.

This approach is ideal where double errors are rare but may occur. To avoid a system crash, a double-error detect now causes the system to enter a subroutine to set the DP8400-2 in COMPLEMENT mode. This method is also useful in bulk-memory applications, where RAMs are used with known cell failures, and is applicable in 16, 32, 48 or 64-bit configuration. In the 16-bit configuration, modes 1 and 5 of Table III are used. In the 32-bit expanded configuration, modes 1, 5 and 5 are used for the highest chip, and modes 3, 3 and 4 for the lower chip for WRITE, DETECT, and CORRECT. With the lower chip it is necessary to wrap around DOL (after latching its contents in mode 3), back to DIL and perform a Normal READ in mode 4 in the lower chip.

TABLE XVII. ERROR FLAGS AFTER COMPLEMENT READ (MODE 5)

AE	E1	E0	Error Type
0	0	0	Two hard errors
1	1	0	One hard error, one soft check bit error
1	1	1	One hard error, one soft data bit error
1	0	0	Two soft errors, not corrected

Other Modes of Operation (Continued)

DOUBLE-ERROR CORRECT WITH ERROR LOGGING

Figures 4 and *5* show the E²C² syndrome port connected to an error management unit (EMU). This scheme stores syndromes and the address of locations that fail, thereby logging the errors. Subsequent errors in a memory location that has already stored syndromes in the EMU, can then be removed by injecting the stored syndromes of the first error. To save the addresses and syndromes when power to the EMU is removed, it is necessary to be able to transfer information via the E²C² syndrome port to the processor data bus. This is also useful for logging the errors in the processor. Transfer in the opposite direction is also necessary.

DATA BUS TO SYNDROME BUS TRANSFER

This is necessary when transferring syndrome information to the error management unit, which is connected to the external syndrome bus. First, data to make CG = 0 (all data bits high) must be latched in DIL. Then in mode 2, data is fed to CIL, XOR-ed with 0, and output via SOL with $\overline{OES}$ low to the syndrome bus. Data is therefore fed directly from the system to the syndrome bus, and this cycle may be repeated as long as DLE is kept low, forcing CG to remain zero.

SYNDROME BUS TO DATA BUS TRANSFER

This is important when information in the error logger or error management unit has to be read. The DP8400-2 is set to mode 6B with $\overline{OES}$ high, and with $\overline{OB0}$, $\overline{OB1}$ and $\overline{OLE}$ low. If CSLE is high, the syndrome bus and check bit bus data appear on the lower and upper bytes of the data bus to be read by the system. Also E1 and E0 values that were valid when mode 6 was entered, appear on DQ7 and DQ15.

FULL DIAGNOSTIC CHECK OF MEMORY

Using mode 2, it is possible to transfer the upper byte of the data bus directly to the CIL, with CSLE high, without affecting DIL. These simulated check bits then appear on the check bit bus with $\overline{OLE}$ low, which also causes the previously latched contents of DIL to transfer to DOL. By enabling $\overline{OB0}$ and $\overline{OB1}$ data can be written to memory with the simulated check bits. A Normal READ cycle can then aid the system in determining that the memory bits are functioning correctly, since the processor knows the check bits and data it sent to the E²C². Another solution is to put the E²C² in mode 6 and read the memory check bits directly back to the processor.

SELF-TEST OF THE E²C² ON-CARD

Again using mode 2, data written from the processor data bus upper byte to CIL may be stored in CIL, by taking CSLE low. Next, a mode 0 WRITE can be performed and the user generated data can be latched in the DP8400-2 input latches (DLE held low). Now the user may perform a normal mode 4 READ. This will in effect be a Diagnostic READ of the user generated data and check bits without using the external memory. Thus by reading corrected data in mode 4, and by reading the generated syndromes, and error flags E0 and E1, the DP8400-2 can be tested completely on-card without involving memory.

MONITORING GENERATED SYNDROMES AND MEMORY CHECK BITS

Mode 6A enables SG0–SG6 onto DQ0–DQ6, and CIL0–CIL6 onto DQ8–DQ14, provided $\overline{OLE}$, $\overline{OB0}$ and $\overline{OB1}$ are low. Also the two error flags, E1 and E0 (latched from the previous READ mode), appear on DQ7 and DQ15. This may be used for checking the internal syndromes, for reading the memory check bits, or for diagnostics by checking the latched error flags.

CLEARING SIL

In the 16-bit only configuration, or the lower chip of expanded configurations, and in various modes of operation in the higher expanded chips, it is required that SIL be maintained at zero. At power-up initialization, both SIL and DIL are reset to all low. If $\overline{OES}$ is kept low, SIL will remain reset because CSLE is inhibited to SIL. Another method is to keep $\overline{OES}$ always high and the syndrome bus externally set low, or set low whenever CSLE can be used to clear SIL.

Mode 7A also forces the SIL to be cleared whenever CSLE occurs, and also these zero syndromes go to the internal syndrome bus SG. This puts the DP8400-2 in a PASS-THROUGH mode where the DIL contents pass to DOL and CIL contents to COL, if $\overline{OLE}$ is low.

POWER-UP INITIALIZATION OF MEMORY

Both SIL and DIL are reset low at power-up initialization. This facilitates writing all zeroes to the memory data bits to set up the memory. The check bits corresponding to all-zero data will appear on the check bit bus if the DP8400-2 is set to mode 0 and $\overline{OLE}$ is set low. All-zero data appears on the data bus when $\overline{OB0}$ and $\overline{OB1}$ are also set low. The system can now write zero-data and corresponding check bits to every memory location.

BYTE WRITING

Figure 14a shows the block diagram of a 16-bit memory correction system consisting of a DP8400-2 error correction chip and a DP8409A DRAM controller chip. There are 12 control signals associated with the interface. Six of the signals are standard DP8400-2 input signals, three are standard DP8409A input signals, and three are buffer control signals. The buffer control signals, $\overline{PBUF0}$ and $\overline{PBUF1}$, control when data words or bytes from the DP8400-2/memory data bus are gated to the processor bus and when data words or bytes from the processor are gated to the DP8400-2/memory data bus.

When the processor is reading or writing bytes to memory, words will always be read or written by the DP8400-2 and DP8409A error correction and DRAM controller section. The High Byte Enable and Address Data Bit Zero signals from the processor should control the byte transfers via the ocal bus transceiver signals $\overline{PBUF0}$ and $\overline{PBUF1}$. The buffer control signal, $\overline{DOUTB}$, controls when data from memory is gated onto the DP8400-2/memory data bus.

Figure 14b shows the timing relationships of the 12 control signals, along with the DP8400-2/memory data bus and some of the DRAM control signals ($\overline{RAS}$ and $\overline{CAS}$). RGCK is the $\overline{RAS}$ generator clock of the DP8409A which is used in Mode 1 (Auto Refresh mode), along with being the system clock.

Other Modes of Operation (Continued)

Having two separate byte enable pins, $\overline{OB0}$ and $\overline{OB1}$, it is easy to implement byte writing using the DP8400-2. First it is necessary to read from the location to which the byte is to be written. To do this the DP8400-2 is put in normal Read mode (Mode 4), which will detect and correct a single bit error. $\overline{WIN}$ is kept high and $\overline{RASIN}$ is pulled low, causing the DP8409A, now in Mode 5 (Auto Access mode), to start a read memory cycle. The data word and check bits from memory are then enabled onto the DP8400-2/memory data bus by pulling $\overline{DOUTB}$ low. The data and check bits are valid on the bus after the $\overline{RASIN}$ to $\overline{CAS}$ time (t_{RAC}) plus the column access time (t_{CAC}) of the particular memories used. DLE, CSLE can then be pulled low in order to latch the memory data into the input latches of the DP8400-2. $\overline{OLE}$ can be pulled low to enable the corrected memory word, or the original memory word if no error was present, into the data output latches. Following this, $\overline{DOUTB}$ can be pulled high to disable the memory data from the DP8400-2/memory data bus. The corrected memory word will be available at the data output latches "t_{DCD16}" after the memory word was available at the data input latches. Once the corrected data is available at the output latches $\overline{OLE}$ can be pulled high to latch the corrected data. Also DLE and CSLE can be pulled high in order to enable the input data latches again.

Now the DP8400-2 can be put into a write cycle (Mode 0 = M2 = Low). At this time the byte to be written to memory and the other byte from memory can be enabled onto the DP8400-2/memory data bus ($\overline{OB0}$, $\overline{PBUF1}$ or $\overline{OB1}$, $\overline{PBUF0}$ go low). DLE, CSLE can now transition low to latch the new memory word into the data input latch. $\overline{OLE}$ is pulled low to enable the output latches. When the new checkbits are valid, t_{DCB16} after the data word is valid on the DP8400-2/memory data bus, $\overline{OLE}$ and $\overline{DLE}$ can be pulled high to latch the new memory word into the output latches, and then $\overline{WIN}$ can be pulled low to write the data into memory. $\overline{RASIN}$ should be held low long enough to cause the new data and check bits to be stored into memory ($\overline{WIN}$ data hold time).

Also a READ-MODIFY-WRITE cycle was performed, taking approximately 40% longer than a normal memory WRITE cycle. A READ and then a WRITE memory cycle could have been used in the above example but it would have taken longer.

Buffers are used in this system (74ALS244) to keep the Data Out and Data In of the memory IC's from conflicting with each other during Read-Modify-Write cycles.

A byte READ from memory is no different from a normal READ. This approach may be used for a 16-bit processor using byte writing, or an 8-bit processor using a 16-bit memory to reduce the memory percentage of check bits, or with memory word sizes greater than two bytes.

An APP NOTE (App Note 387) has been written detailing an Error Correcting Memory System using the DP8409A or DP8419 (Dynamic RAM Controller) and the DP8400-2 interfaced to a National Semiconductor Series 32000 CPU. See this App Note for further system details and considerations.

BEYOND SINGLE-ERROR CORRECT

With the advent of larger semiconductor memories, the frequency of the soft errors will increase. Also some memory system designers may prefer to buy less expensive memories with known cell, row or column failures, thus, more hard errors. All this means that double-error correct, triple-error detect capability, and beyond will become increasingly important. The DP8400-2 can correct two errors, provided one or both are hard errors, with no extra components, using the double complement approach. There are two other approaches to enhance reliability and integrity. One is to use the error management unit to log errors using the syndrome bus, and then to output these syndromes, when required, back to the DP8400-2.

DOUBLE SYNDROME DECODING

The other approach takes advantage of the Rotational Syndrome Word Generator matrix. This matrix is an improvement of the Modified Hamming-code, so that if, on a second DP8400-2, the data bus is shifted or rotated by one bit, and 2 errors occur, the syndromes for this second chip will be different from the first for any 2 bits in error. Both chips together output a unique set of syndromes for any 2 bits in error. This can be decoded to correct any 2-bit error. This is not possible with other Modified Hamming-code matrices.

Absolute Maximum Ratings (Note 1)

Storage Temperature Range	−65°C to +150°C
Supply Voltage, V_{CC}	7V
Input Voltage	5.5V
Output Sink Current	50 mA
Maximum Power Dissipation at 25°C	
Molded Package	3269 mW
Lead Temperature (Soldering, 10 seconds)	300°C

*Derate molded package 26.2 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
V_{CC}, Supply Voltage	4.75	5.25	V
T_A, Ambient Temperature	0	70	°C

Electrical Characteristics (Note 2) V_{CC} = 5V ±5%, T_A = 0°C to 70°C unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_C	Input Clamp Voltage	V_{CC} = Min, I_C = −18 mA		−0.8	−1.5	V
I_{IH}	Input High Current	V_{IN} = 2.7V		1	160	μA
I_{IH} (XP)	Input High Current	V_{CC} = Max, XP = 5.25V		2.5	4.5	mA
I_{IL} (XP)	Input Low Current	V_{CC} = Max, XP = 0V		−2.5	−4.5	mA
I_{IL} (BP0/C7)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−100.0	−500	μA
I_{IL} (BP1/S7)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−100.0	−500	μA
I_{IL} (CSLE)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−150.0	−750	μA
I_{IL} (DLE)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−200.0	−1000	μA
I_{IL}	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−50.0	−250	μA
I_I	Input High Current (Max)	V_{IN} = 5.5V (Except XP Pin)			1.0	mA
V_{OL}	Output Low Voltage	I_{OL} = 8 mA (Except BP0, BP1) I_{OL} = 4 mA (BP0, BP1 Only)		0.3 0.3	0.5 0.5	V V
V_{OH}	Output High Voltage	I_{OH} = −100 μA I_{OH} = −1 mA	2.7 2.4	3.2 3.0		V V
I_{OS}	Output Short Current (Note 3)	V_{CC} = Max		−150	−250	mA
I_{CC}	Supply Current	V_{CC} = Max		220	300	mA
C_{IN} (I/O)	Input Capacitance All Bidirectional Pins	Note 4		8.0		pF
C_{IN}	Input Capacitance All Unidirectional Input Pins	Note 4		5.0		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for T_A = 25°C and V_{CC} = 5.0V

Note 3: Only one output at a time should be shorted.

Note 4: Input capacitance is guaranteed by periodic testing. F test = 10 kHz at 300 mV, T_A = 25°C.

Note 5: All switching parameters measured from 1.5V of input to 1.5V of output. Input pulse amplitude 0V to 3V, $t_r = t_f$ = 2.5 ns.

DP8400-2 Switching Characteristics (Note 5)

V_{CC} = 5.0V ±5%, T_A = 0°C to 70°C, C_L = 50 pF

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{DCB16}	Data Input Valid to Check Bit Valid	*Figure 9b*		29	40	ns
t_{DEV16}	Data Input to Any Error Valid	*Figures 10b, 11b*		21	31	ns
t_{DCD16}	Data Input Valid to Corrected Data Valid	*Figure 10b*, $\overline{OB0}$, $\overline{OB1}$ Low		44	61	ns
t_{DSI}	Data Input Set-Up Time Before DLE, CSLE H to L	*Figures 10b, 13d*	10	5		ns
t_{DHI}	Data Input Hold Time After DLE, CSLE H to L	*Figures 10b, 13d*	10	5		ns
t_{DSO}	Data Input Set-Up Time Before $\overline{OLE}$ L to H	*Figure 10b*	10	5		ns
t_{DHO}	Data Input Hold Time After $\overline{OLE}$ L to H	*Figure 10b*	10	5		ns
t_{DE0}	Data in Valid to E0 Valid	*Figures 9b, 10b, 13d*		36	55	ns
t_{DE1}	Data in Valid to E1 Valid	*Figures 9b, 10b, 13d*		43	55	ns
t_{IEV}	DLE, CSLE High to Any Error Flag Valid (Input Data Previously Valid)	*Figure 10b*		28	45	ns
t_{IEX}	DLE, CSLE High to Any Error Flag Invalid	*Figures 9b, 10b*		38	60	ns
t_{ILE}	DLE, CSLE High Width to Guarantee Valid Data Latched	*Figures 10b, 13d*	20			ns
t_{OLE}	$\overline{OLE}$ Low Width to Guarantee Valid Data Latched	*Figure 13d*	20			ns
t_{ZH}	High Impedance to Logic 1 from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$ M2 H to L	*Figures 9b, 10b, 13d*		22	36	ns
t_{HZ}	Logic 1 to High Impedance from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$, M2 L to H	*Figures 9b, 10b, 13d,*		38	55	ns
t_{ZL}	High Impedance to Logic 0 from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$ M2 H to L	*Figures 9b, 10b, 13d*		19	35	ns
t_{LZ}	Logic 0 to High Impedance from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$, M2 H to L	*Figures 9b, 10b, 13d*		15	25	ns
t_{PPE}	Byte Parity Input Valid to Parity Error Flags Valid	*Figure 9b*		16	27	ns
t_{DPE}	Data In Valid to Parity Error Flags Valid	*Figures 9b, 13d*		27	55	ns
t_{DCP}	Data in Valid to Corrected Byte Parity Output Valid	*Figure 9b*		44	61	ns

DP8400-2 Switching Characteristics (Continued) (Note 5)

V_{CC} = 5.0V ±5%, T_A = 0°C to 70°C, C_L = 50 pF

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{NMR}	New Mode Recognize Time	*Figure 10b*		22	35	ns
t_{CDV}	Mode Valid to Complement Data Valid	*Figure 11b*		34	50	ns
t_{CCV}	Mode Valid to Complement Check Bit Valid	*Figure 11b*		30	45	ns
t_{SCB}	Syndrome Input Valid to Check Bit Valid	*Figure 13d*		20	35	ns
t_{SEV}	Syndrome Input Valid (CGL) to Any Error Valid	*Figure 13d*		17	27	ns
t_{SCD}	Syndrome Inputs Valid to Corrected Data Valid	*Figure 13d*		35	50	ns
t_{DSB}	Data Input Valid to Syndrome Bus Valid	*Figure 13d*, $\overline{OES}$ Low		28	46	ns
t_{CSB}	Check Bit Inputs Valid to Syndrome Bus Valid	*Figure 13d*, $\overline{OES}$ Low		19	32	ns
t_{CEV}	Check Bit Inputs Valid (PSH) to Any Error Valid	*Figure 13d*		17	30	ns
t_{CCD}	Check Bit Input Valid (PSH) to Corrected Data Valid	*Figure 13d*		30	45	ns
t_{DCB32}	Data Input Valid to Check Bit Valid	*Figure 13d*		49	75	ns
t_{DEV32}	Data Input Valid to Any Error Valid	*Figure 13d*		46	67	ns
t_{DCD32}	Data Input Valid to Corrected Data Out	*Figure 13d*, $\overline{OB0}$, $\overline{OB1}$ Low		84	110	ns

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

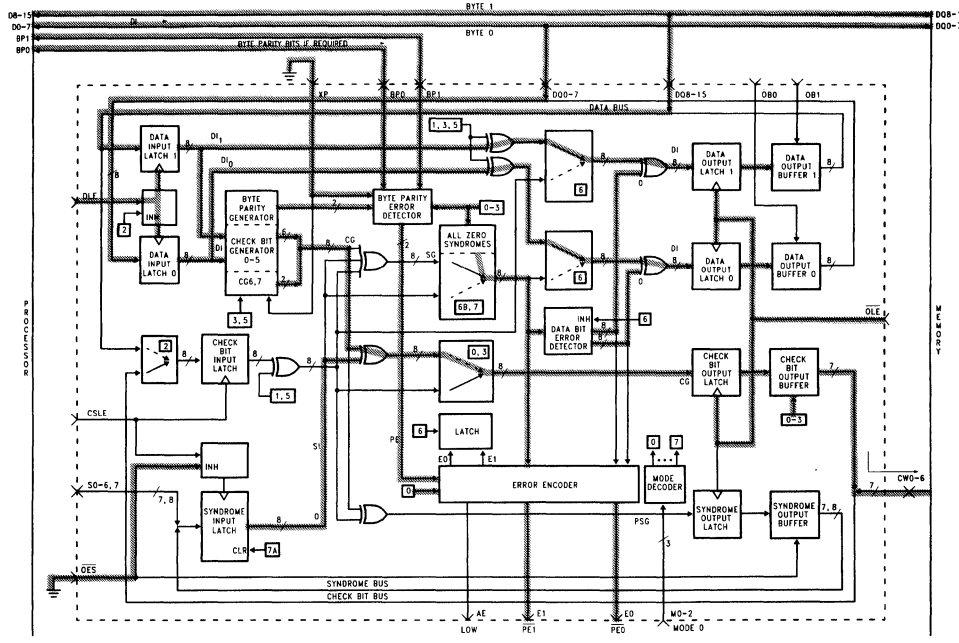
Note 2: All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: Only one output at a time should be shorted.

Note 4: Input capacitance is guaranteed by periodic testing. F test = 10 kHz at 300 mV, $T_A = 25°C$.

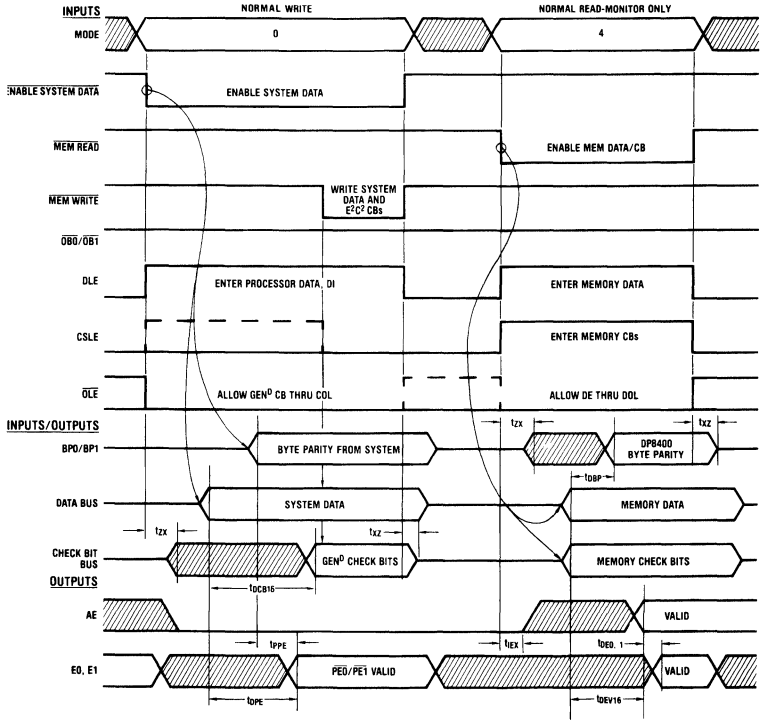
Note 5: All switching parameters measured from 1.5V of input to 1.5V of output. Input pulse amplitude 0V to 3V, $t_r = t_f = 2.5$ ns.

Typical Applications



TL/F/6899–17

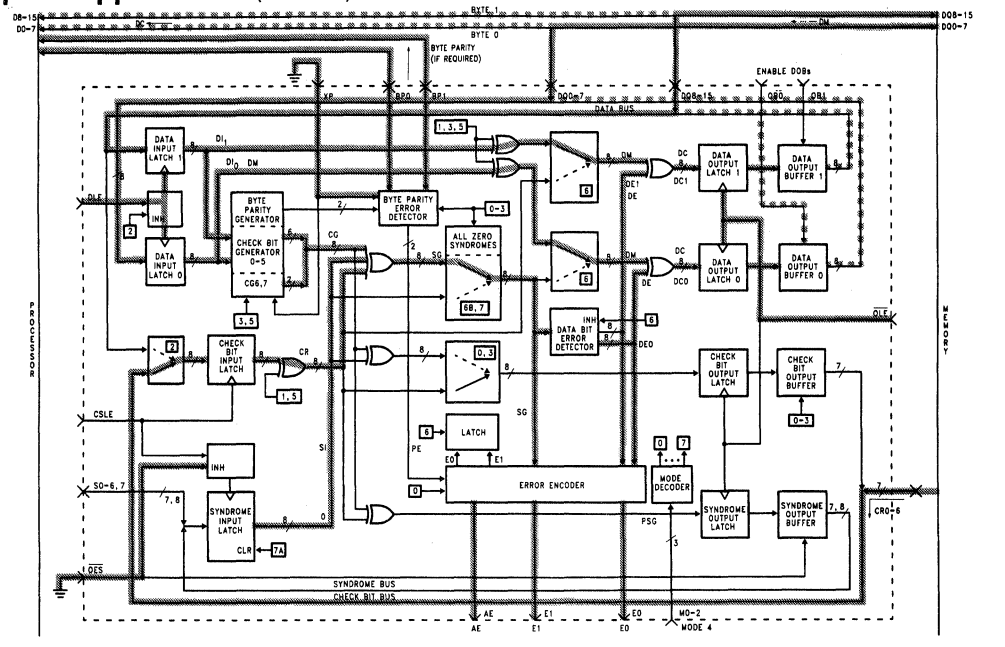
FIGURE 9a. DP8400-2 16-Bit Configuration, Normal WRITE with Byte Parity Error Detect If Required



TL/F/6899–18

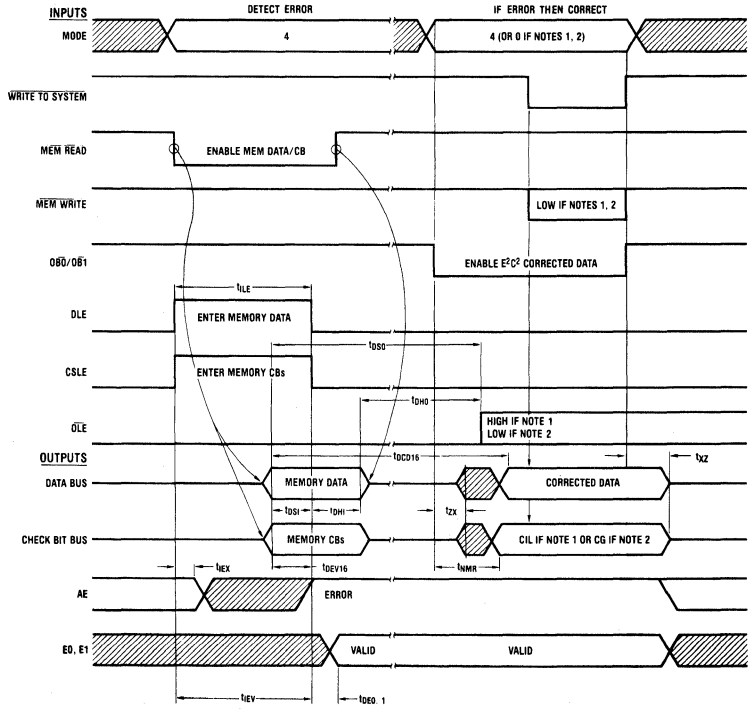
FIGURE 9b. DP8400-2 16-Bit Configuration, Normal WRITE and Normal READ Timing Diagram

Typical Applications (Continued)



TL/F/6899–19

FIGURE 10a. DP8400-2 16-Bit Configuration, Normal READ — Detect Error (And Correct if Required---)



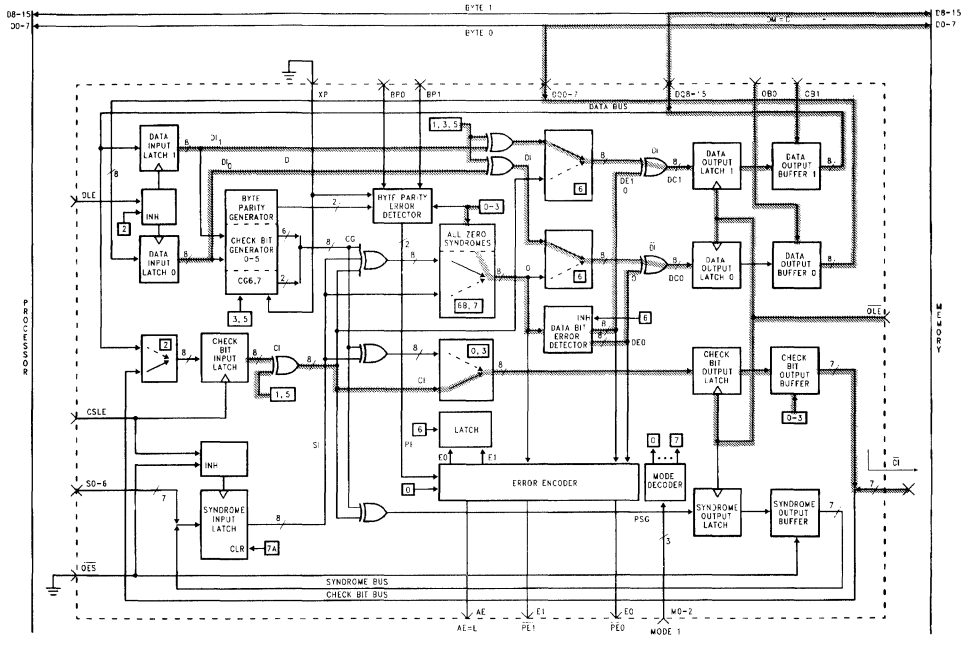
TL/F/6899–20

Note 1: If rewriting correct data and CBs to same location and single data error was detected.

Note 2: If rewriting correct data and CBs to same location and single check bit was detected.

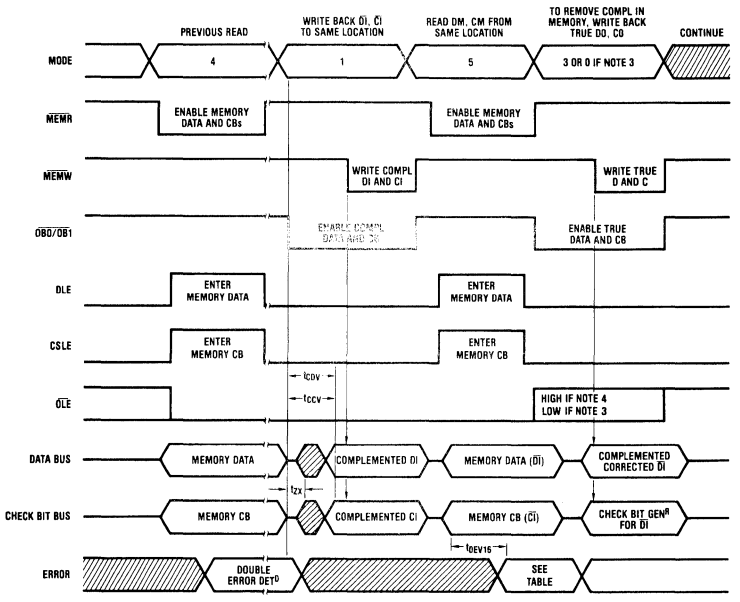
FIGURE 10b. DP8400-2 16-Bit Configuration, DETECT THEN CORRECT Timing Diagram

Typical Applications (Continued)



TL/F/6899–21

FIGURE 11a. DP8400-2 16-Bit Configuration, COMPLEMENT WRITE



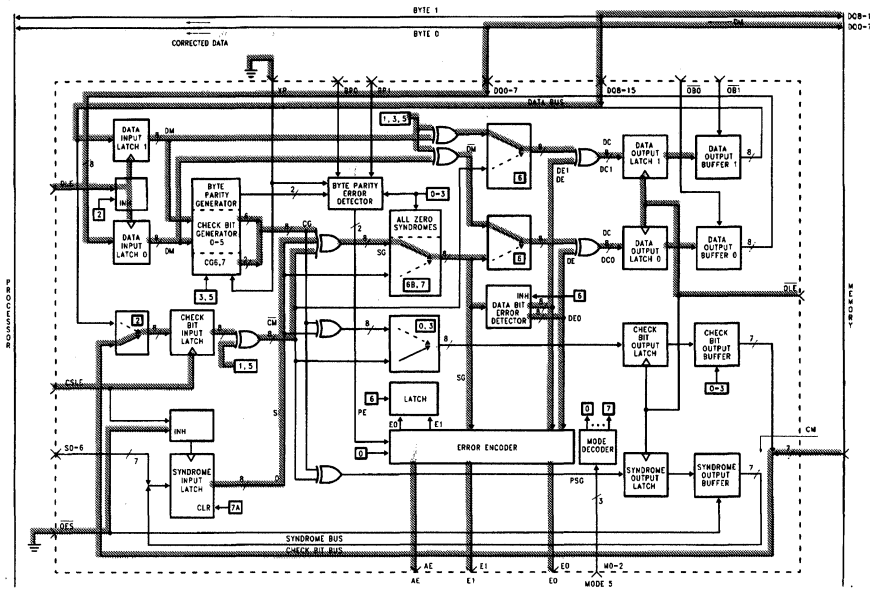
TL/F/6899–22

Note 3: If rewriting corrected data and CBs back to same location and 1 soft data bit error was detected.

Note 4: If rewriting corrected data and CBs back to same location and 2 hard errors or 1 soft check bit was detected.

FIGURE 11b. DP8400-2 16-Bit Configuration, Detect 2 Errors, COMPLEMENT WRITE, COMPLEMENT READ, Output Corrected Data Timing Diagram

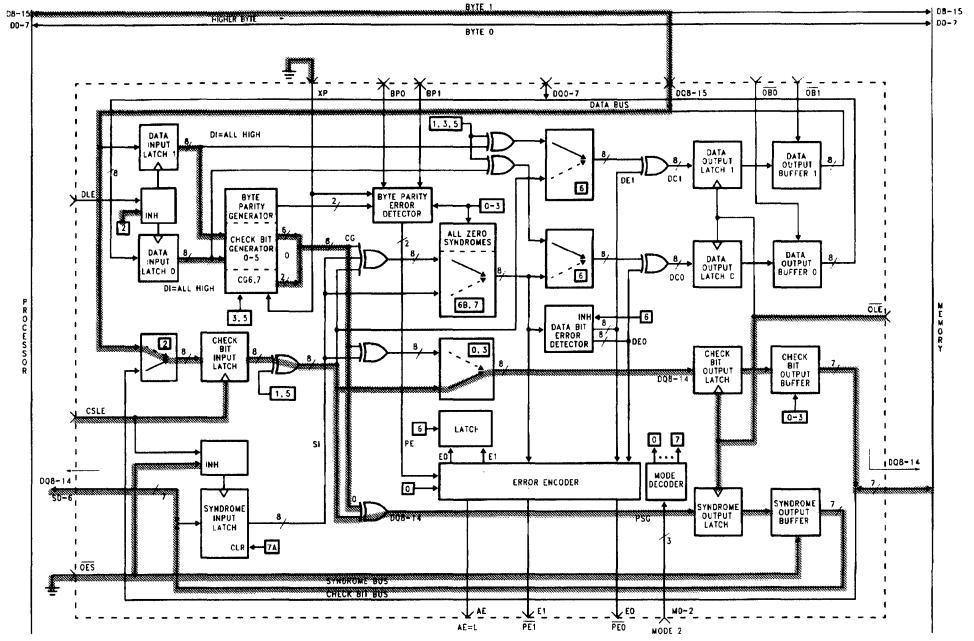
Typical Applications (Continued)



TL/F/6899–23

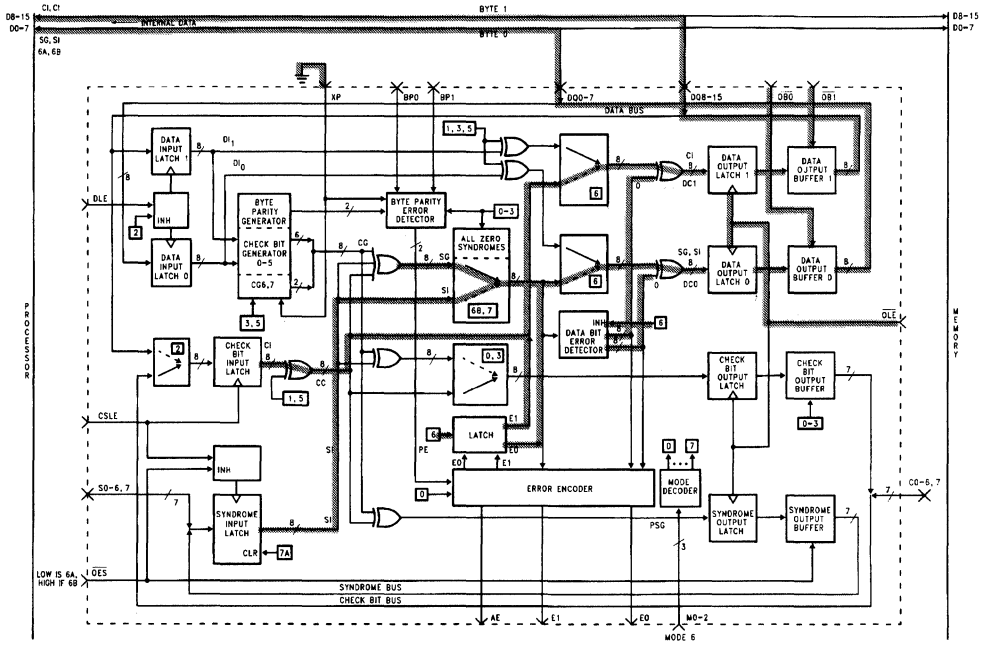
FIGURE 11c. DP8400-2 16-Bit Configuration, COMPLEMENT READ and Output Corrected if One or Two Hard Errors

Typical Applications (Continued)



TL/F/6899-24

FIGURE 12a. DP8400-2 16-Bit Configuration, Diagnostic WRITE, READ. Data Bus to Check Bit Bus or Syndrome Bus (Providing DI = HIGH in Previous Cycle to Set CG = All Zero For Transfer to S)



TL/F/6899-25

FIGURE 12b. DP8400-2 16-Bit Configuration, Monitor on Data Bus — Memory Check Bits

Typical Applications (Continued)

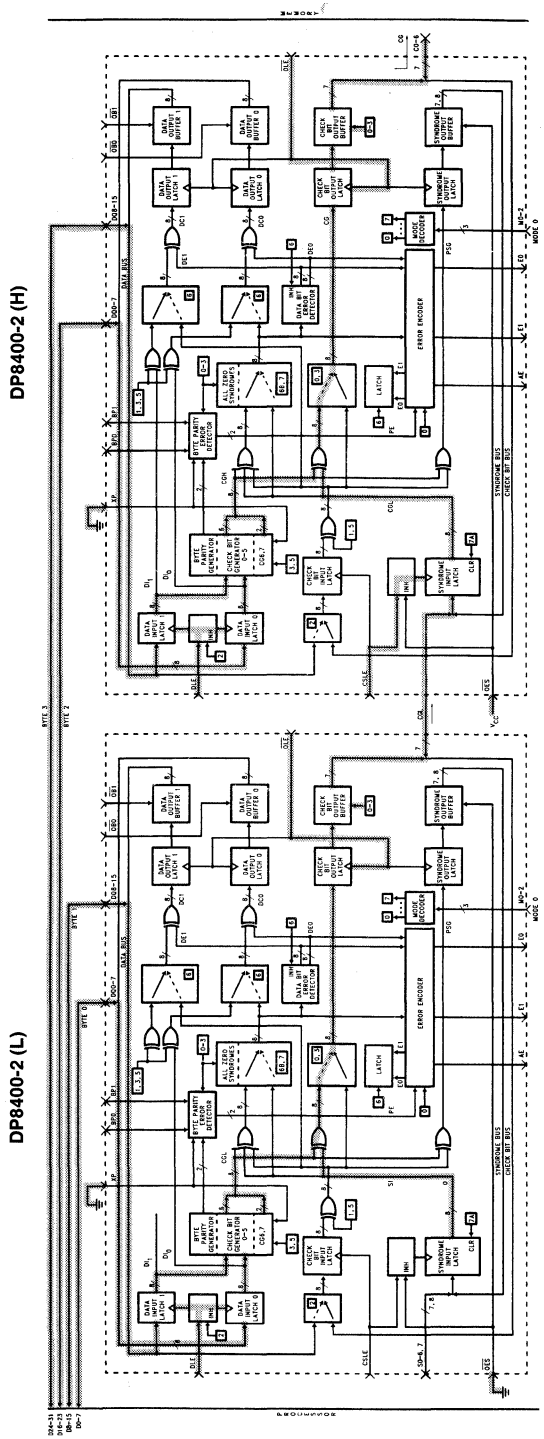


FIGURE 13a. DP8400-2 32-Bit Configuration, WRITE

TL/F/6899–26

Typical Applications (Continued)

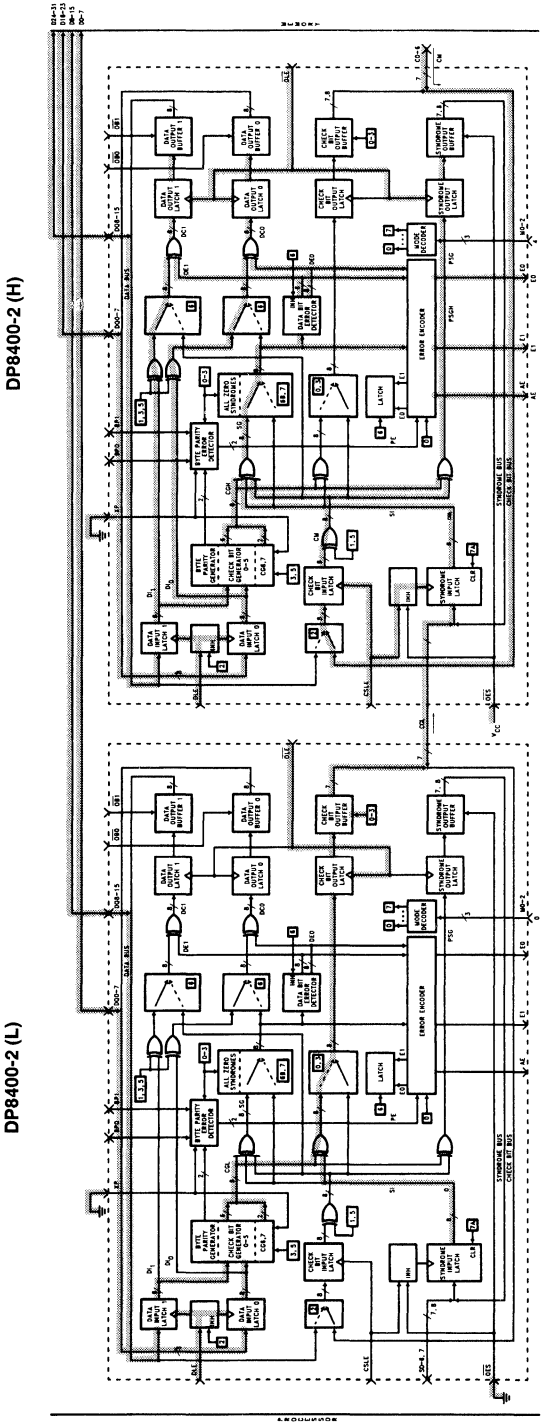


FIGURE 13b. DP8400-2 32-Bit Configuration, READ Detect Error Only

TL/F/6899–27

Typical Applications (Continued)

DP8400-2 (L)

DP8400-2 (H)

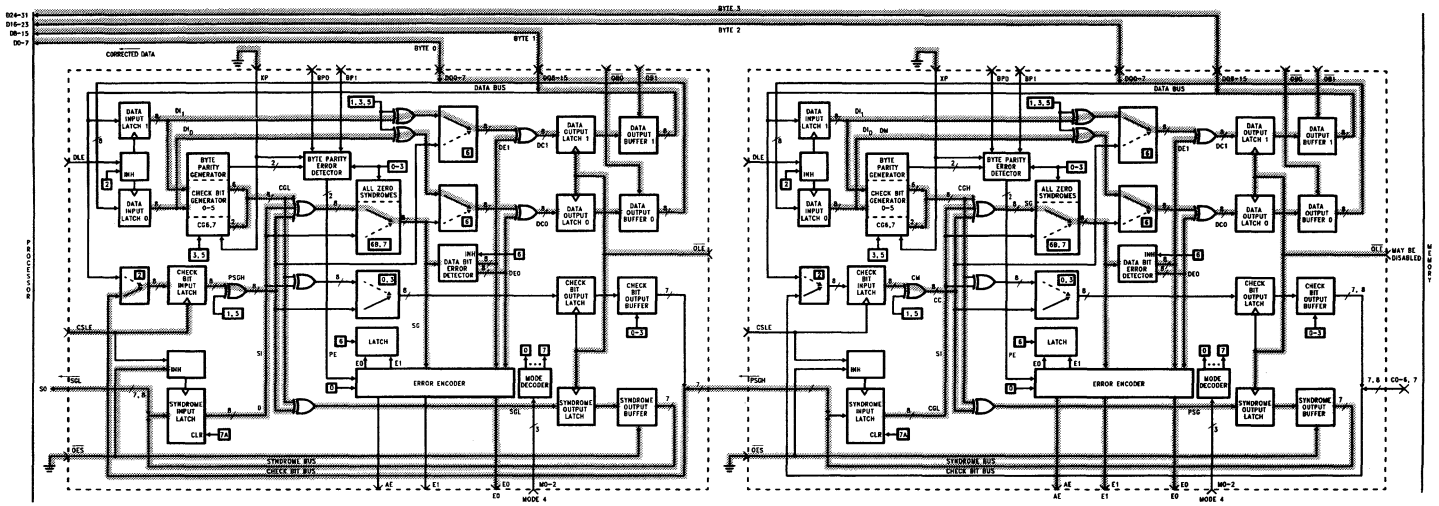
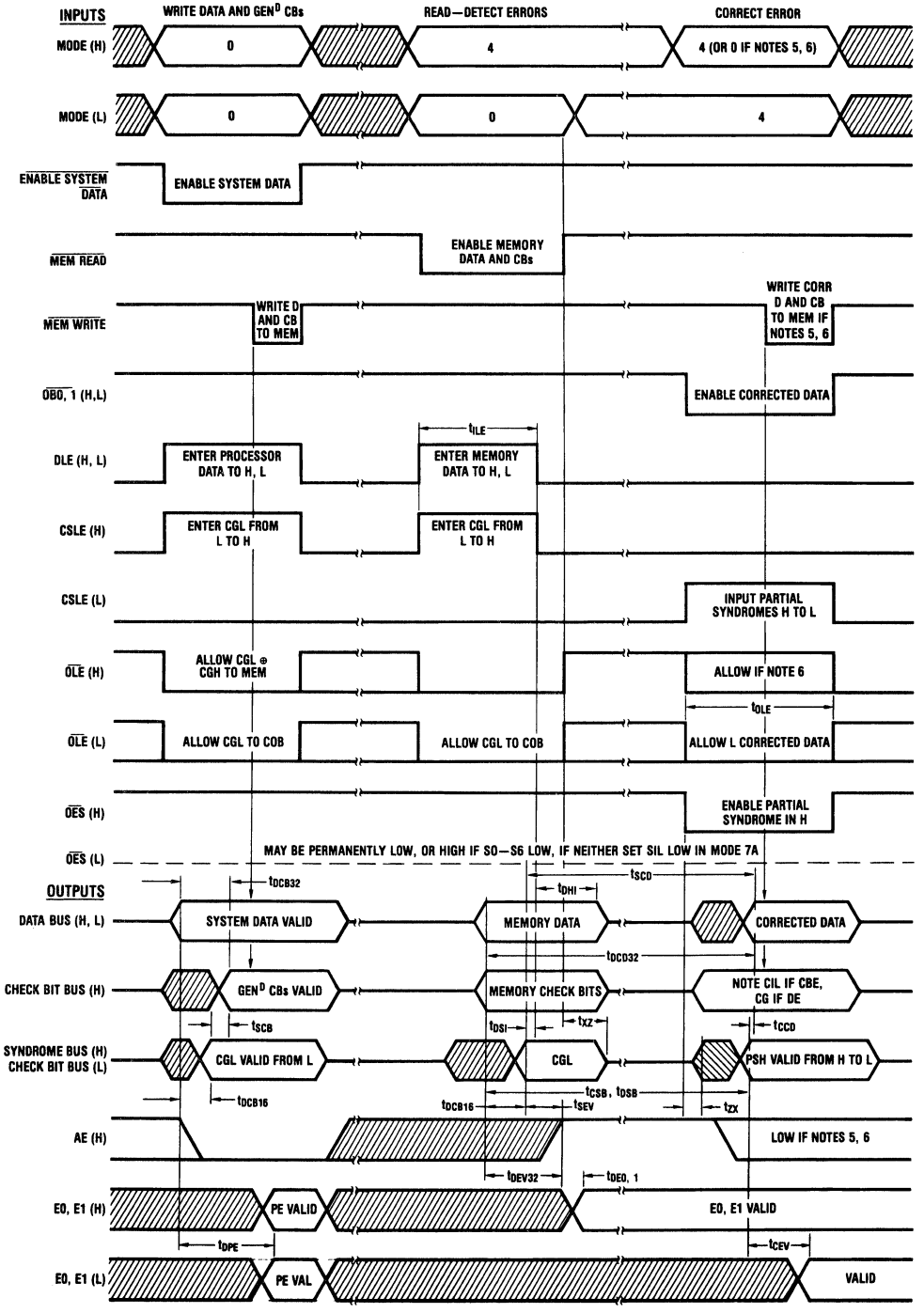


FIGURE 13c. DP8400-2 32-Bit Configuration, READ Correct Data

TL/F/6899–28

Typical Applications (Continued)

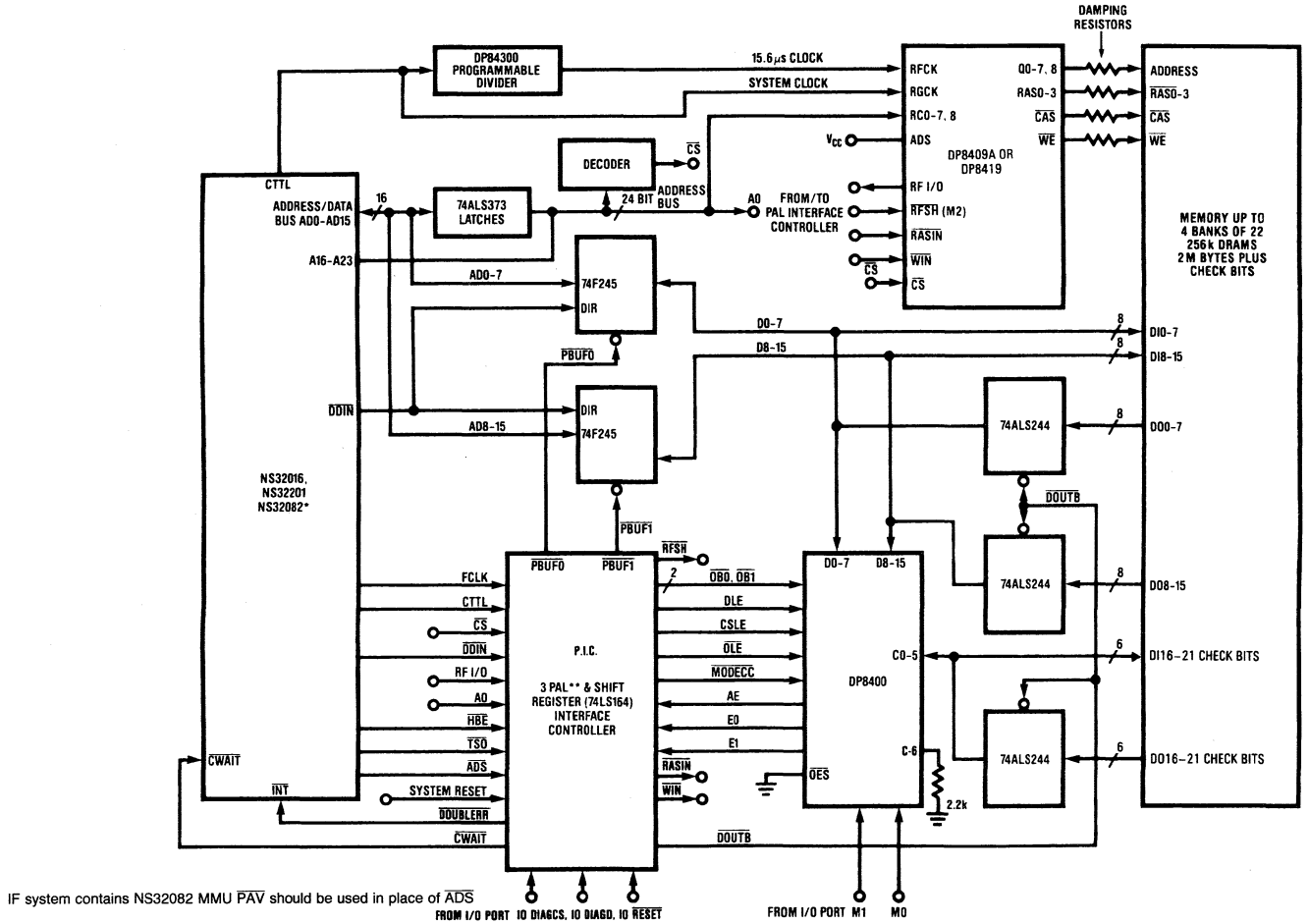


Note 5: If rewriting corrected data and CBs back to same location and single data error was detected. TL/F/6899–29

Note 6: If rewriting corrected data and CBs back to same location and single check bit error was detected.

FIGURE 13d. DP8400-2 32-Bit Configuration, WRITE, DETECT and CORRECT Timing Diagram

NS32016, DP8400-2, DP8409A or DP8419 Error Correcting Memory System

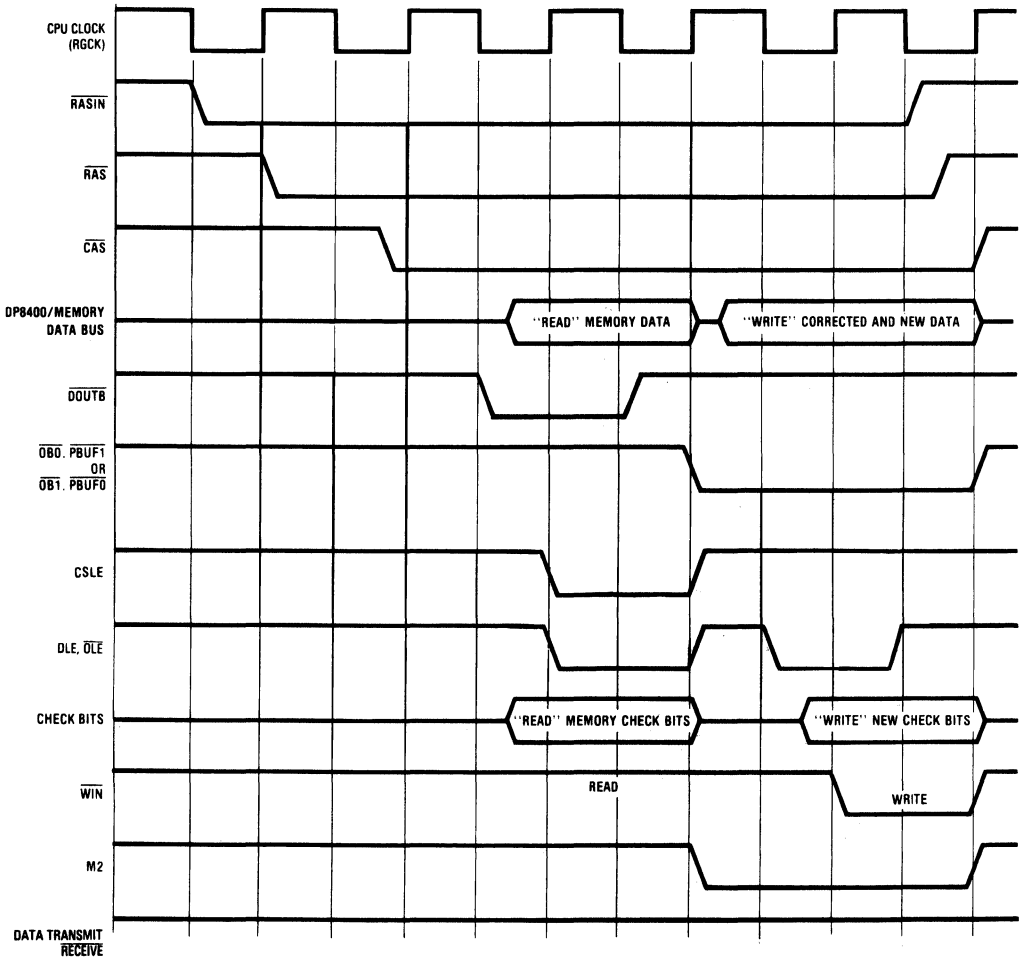


IF system contains NS32082 MMU $\overline{PAV}$ should be used in place of $\overline{ADS}$

TL/F/6899–30

FIGURE 14a. DP8400-2/8409A System Interface Block Diagram (See Figure 14b for Byte Write Control Timing)

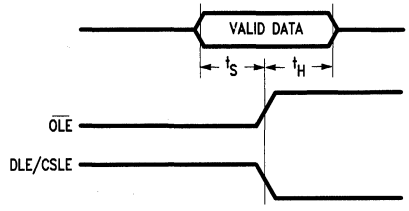
Typical Applications (Continued)



TL/F/6899–31

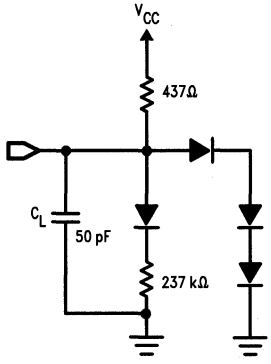
FIGURE 14b. DP8400-2 16-Bit Configuration, Byte Write Timing

Typical Applications (Continued)



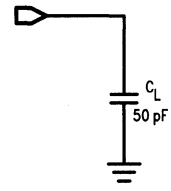
TL/F/6899–32

FIGURE 15. Timing Waveform for Set-Up and Hold Time



TL/F/6899–33

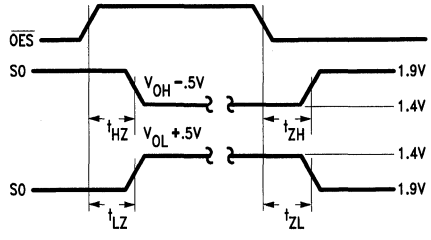
(a) TRI-STATE LOAD



TL/F/6899–34

(b) TEST LOAD

FIGURE 16. Loading Circuit



TL/F/6899–35

FIGURE 17. TRI-STATE Measurement

National Semiconductor Corporation

PRELIMINARY

DP8402A/DP8403/DP8404/DP8405 32-Bit Parallel Error Detection and Correction Circuits (EDAC's)

General Description

The DP8402A, DP8403, DP8404 and DP8405 devices are 32-bit parallel error detection and correction circuits (EDACs) in 52-pin DP8402A and DP8403 or 48-pin DP8404 and DP8405 600-mil packages. The EDACs use a modified Hamming code to generate a 7-bit check word from a 32-bit data word. This check word is stored along with the data word during the memory write cycle. During the memory read cycle, the 39-bit words from memory are processed by the EDACs to determine if errors have occurred in memory.

Single-bit errors in the 32-bit data word are flagged and corrected.

Single-bit errors in the 7-bit check word are flagged, and the CPU sends the EDAC through the correction cycle even though the 32-bit data word is not in error. The correction cycle will simply pass along the original 32-bit data word in this case and produce error syndrome bits to pinpoint the error-generating location.

Double bit errors are flagged but not corrected. These errors may occur in any two bits of the 39-bit word from memory (two errors in the 32-bit data word, two errors in the 7-bit check word, or one error in each word). The gross-error condition of all lows or all highs from memory will be detected. Otherwise, errors in three or more bits of the 39-bit word are beyond the capabilities of these devices to detect.

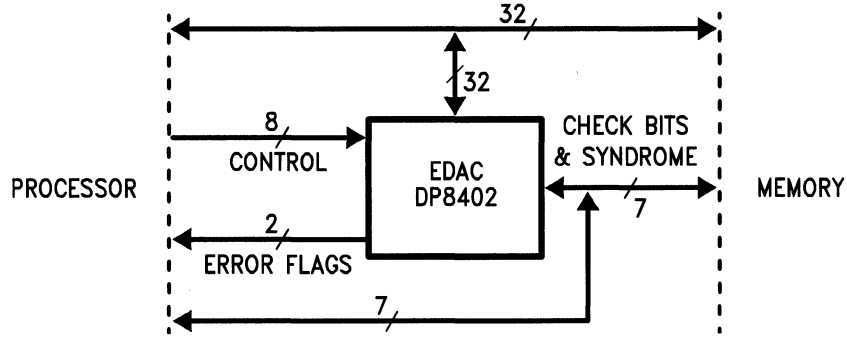
Read-modify-write (byte-control) operations can be performed with the DP8402A and DP8403 EDACs by using output latch enable, $\overline{\text{LEDBO}}$, and the individual $\overline{\text{OEB0}}$ thru $\overline{\text{OEB3}}$ byte control pins.

Diagnostics are performed on the EDACs by controls and internal paths that allow the user to read the contents of the DB and CB input latches. These will determine if the failure occurred in memory or in the EDAC.

Features

- Detects and corrects single-bit errors
- Detects and flags double-bit errors
- Built-in diagnostic capability
- Fast write and read cycle processing times
- Byte-write capability . . . DP8402A and DP8403
- Fully pin and function compatible with TI's SN74ALS632A thru SN74ALS635 series

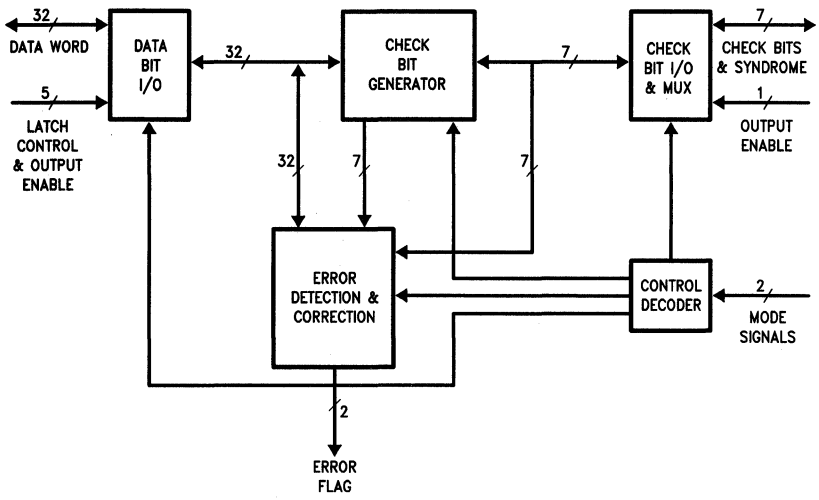
System Environment



TL/F/8535–1

7

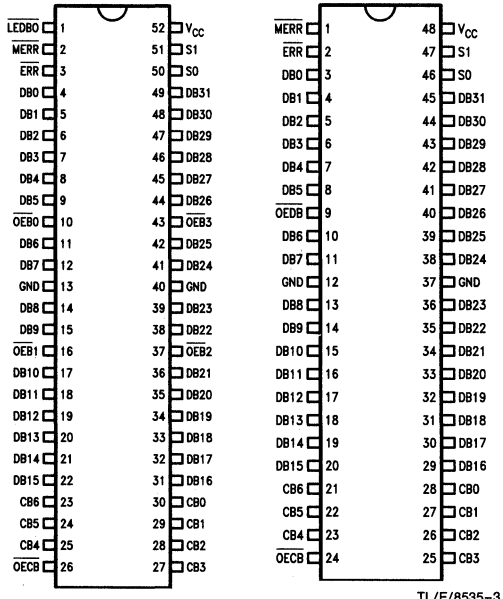
Simplified Functional Block and Connection Diagrams



TL/F/8535–2

Device	Package	Byte-Write	Output
DP8402A	52-pin	yes	TRI-STATE®
DP8403	52-pin	yes	Open-Collector
DP8404	48-pin	no	TRI-STATE
DP8405	48-pin	no	Open-Collector

Dual-In-Line Packages



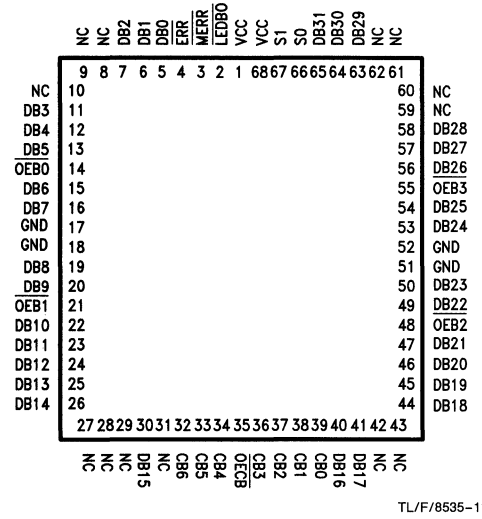
TL/F/8535–10

TL/F/8535–3

Top View

Order Number DP8402AD, DP8403D, DP8404D or DP8405D
See NS Package Number D48A or D52A

Plastic Chip Carrier



TL/F/8535–11

Top View

Order Number DP8402AV
See NS Package Number V68A

Mode Definitions

MODE	PIN NAME S1	S0	DESCRIPTION MODE	OPERATION
0	L	L	WRITE	Input dataword and output checkword
1	L	H	DIAGNOSTICS	Input various data words against latched checkword/output valid error flags.
2	H	L	READ & FLAG	Input dataword and output error flags
3	H	H	CORRECT	Latched input data and checkword/output corrected data and syndrome code

Pin Definitions

S0, S1 — Control of EDAC mode, see preceding Mode Definitions

DB0 thru DB31 — I/O port for 32 bit dataword.

CB0 thru CB6 — I/O port for 7 bit checkword. Also output port for the syndrome error code during error correction mode.

$\overline{OEB0}$ thru $\overline{OEB3}$ (DP8402A, DP8403) — Dataword output buffer enable. When high, output buffers are at TRI-STATE. Each pin controls 8 I/O ports. $\overline{OEB0}$ controls DB0 thru DB7, $\overline{OEB1}$ controls DB8 thru DB15, $\overline{OEB2}$ controls DB16 thru DB23 and $\overline{OEB3}$ controls DB24 thru DB31.

$\overline{LEDBO}$ (DP8402A, DP8403) — Data word output Latch enable. When high it inhibits input to the Latch. Operates on all 32 bits of the dataword.

$\overline{OEDB}$ (DP8404, DP8405) — TRI-STATE control for the data I/O port. When high output buffers are at TRI-STATE.

$\overline{OECB}$ — Checkword output buffer enable. When high the output buffers are in TRI-STATE mode.

$\overline{ERR}$ — Single error output flag, a low indicates at least a single bit error.

$\overline{MERR}$ — Multiple error output flag, a low indicates two or more errors present.

PCC Pin Definitions DP8402A

pin 1	V_{CC}	pin 35	$\overline{OECB}$
2	$\overline{LEDBO}$	36	CB3
3	$\overline{MERR}$	37	CB2
4	$\overline{ERR}$	38	CB1
5	DB0	39	CB0
6	DB1	40	DB16
7	DB2	41	DB17
8	NC	42	NC
9	NC	43	NC
10	NC	44	DB18
11	DB3	45	DB19
12	DB4	46	DB20
13	DB5	47	DB21
14	$\overline{OEBO}$	48	$\overline{OEB2}$
15	DB6	49	DB22
16	DB7	50	DB23
17	GND	51	GND
18	GND	52	GND
19	DB8	53	DB24
20	DB9	54	DB25
21	$\overline{OEB1}$	55	$\overline{OEB3}$
22	DB10	56	DB26
23	DB11	57	DB27
24	DB12	58	DB28
25	DB13	59	NC
26	DB14	60	NC
27	NC	61	NC
28	NC	62	NC
29	NC	63	DB29
30	DB15	64	DB30
31	NC	65	DB31
32	CB6	66	S0
33	CB5	67	S1
34	CB4	68	V_{CC}

TABLE I. Write Control Function

Memory Cycle	EDAC Function	Control S1	S0	Data I/O	DB Control $\overline{OEBn}$ or $\overline{OEDB}$	DB Output Latch DP8402A, DP8403 $\overline{LEDBO}$	Check I/O	CB Control $\overline{OECB}$	Error Flags $\overline{ERR}$	$\overline{MERR}$
Write	Generate check word	L	L	Input	H	X	Output check bits†	L	H	H

†See Table II for details on check bit generation.

Memory Write Cycle Details

During a memory write cycle, the check bits (CB0 thru CB6) are generated internally in the EDAC by seven 16-input parity generators using the 32-bit data word as defined in Table 2. These seven check bits are stored in memory along with the original 32-bit data word. This 32-bit word will later be used in the memory read cycle for error detection and correction.

TABLE II. Parity Algorithm

Check Word	32-Bit Data Word																													
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CB0	X		X	X		X					X		X	X	X			X			X		X	X	X	X		X				X
CB1				X		X		X		X		X		X	X	X				X		X		X		X		X			X	X
CB2	X		X			X	X		X			X	X			X	X		X			X	X		X		X	X	X		X
CB3			X	X	X				X	X	X				X	X			X	X	X				X	X	X			X	X
CB4	X	X							X	X	X	X	X	X			X	X							X	X	X	X	X	X		
CB5	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X								
CB6	X	X	X	X	X	X	X	X																	X	X	X	X	X	X	X	X

The seven check bits are parity bits derived from the matrix of data bits as indicated by "X" for each bit.

Memory Read Cycle (Error Detection & Correction Details)

During a memory read cycle, the 7-bit check word is retrieved along with the actual data. In order to be able to determine whether the data from the memory is acceptable to use as presented on the bus, the error flags must be tested to determine if they are at the high level.

The first case in Table III represents the normal, no-error conditions. The EDAC presents highs on both flags. The next two cases of single-bit errors give a high on $\overline{MERR}$ and a low on $\overline{ERR}$, which is the signal for a correctable error, and the EDAC should be sent through the correction cycle. The last three cases of double-bit errors will cause the EDAC to signal lows on both $\overline{ERR}$ and $\overline{MERR}$, which is the interrupt indication for the CPU.

TABLE III. Error Function

Total Number of Errors		Error Flags		Data Correction
32-Bit Data Word	7-Bit Check Word	$\overline{ERR}$	$\overline{MERR}$	
0	0	H	H	Not applicable
1	0	L	H	Correction
0	1	L	H	Correction
1	1	L	L	Interrupt
2	0	L	L	Interrupt
0	2	L	L	Interrupt

The DP8402 check bit syndrome matrix can be seen in TABLE II. The horizontal rows of this matrix generate the check bits by selecting different combinations of data bits, indicated by "X"s in the matrix, and generating parity from them. For instance, parity check bit "0" is generated by EXCLUSIVE NORing the following data bits together; 31, 29, 28, 26, 21, 19, 18, 17, 14, 11, 9, 8, 7, 6, 4, and 0.

During a WRITE operation (mode 0) the data enters the DP8402 and check bits are generated at the check bit input/output port. Both the data word and the check bits are then written to memory.

During a READ operation (mode 2, error detection) the data and check bits that were stored in memory, now possibly in error, are input through the data and check bit I/O ports. New check bits are internally generated from the data word. These new check bits are then compared, by an EXCLUSIVE NOR operation, with the original check bits that were stored in memory. The EXCLUSIVE NOR of the original check bits, that were stored in memory, with the new check bits is called the syndrome word. If the original check bits are the same as the new check bits, a no error condition, then a syndrome word of all ones is produced and both error flags ($\overline{ERR}$ and $\overline{MERR}$) will be high. The DP8402 matrix encodes errors as follows:

TABLE IV. Read, Flag, and Correct Function

Memory Cycle	EDAC Function	Control S1	Control S0	Data I/O	DB Control $\overline{OEBn}$ or $\overline{OEDB}$	DB Output Latch DP8402A, DP8403 $\overline{LEDBO}$	Check I/O	CB Control $\overline{OECB}$	Error Flags $\overline{ERR}$	Error Flags $\overline{MERR}$
Read	Read & flag	H	L	Input	H	X	Input	H	Enabled†	
Read	Latch input data and check bits	H	H	Input data latched	H	L	Input check word latched	H	Enabled†	
Read	Output corrected data & syndrome bits	H	H	Output corrected data word	L	X	Output syndrome bits‡	L	Enabled†	

†See Table III for error description.

‡See Table V for error location.

Memory Read Cycle (Error Detection & Correction Details) (Continued)

1) Single data bit errors cause 3 or 5 bits in the syndrome word to go low. The columns of the check bit syndrome matrix (TABLE II) are the syndrome words for all single bit data errors in the 32 bit word (also see TABLE V). The data bit in error corresponds to the column in the check bit syndrome matrix that matches the syndrome word. For instance, the syndrome word indicating that data bit 31 is in error would be (CB6-CB0) = "0001010", see the column for data bit 31 in TABLE II, or see TABLE V. During mode 3 (S0 = S1 = 1) the syndrome word is decoded, during single data bit errors, and used to invert the bit in error thus correcting the data word. The corrected word is made available on the data I/O port (DB0 thru DB31), the check word I/O port (CB0 thru CB6) presents the 7-bit syndrome error code. This syndrome error code can be used to locate the bad memory chip.

2) A single check bit error will cause that particular check bit to go low in the syndrome word.

3) A double bit error will cause an even number of bits in the syndrome word to go low. The syndrome word will then be the EXCLUSIVE NOR of the two individual syndrome words corresponding to the 2 bits in error. The two-bit error is not correctable since the parity tree can only identify single bit errors.

If any of the bits in the syndrome word are low the "$\overline{\text{ERR}}$" flag goes low. The "$\overline{\text{MERR}}$" (dual error) flag goes low during any double bit error conditions. (See Table III).

Three or more simultaneous bit errors can cause the EDAC to believe that no error, a correctable error, or an uncorrectable error has occurred and will produce erroneous results in all three cases. It should be noted that the gross-error conditions of all lows and all highs will be detected.

TABLE V. Syndrome Decoding

Syndrome Bits							Error
6	5	4	3	2	1	0	
L	L	L	L	L	L	L	unc
L	L	L	L	L	L	H	2-bit
L	L	L	L	L	H	L	2-bit
L	L	L	L	L	H	H	unc
L	L	L	L	H	L	L	2-bit
L	L	L	L	H	L	H	unc
L	L	L	L	H	H	L	unc
L	L	L	L	H	H	H	2-bit
L	L	L	H	L	L	L	2-bit
L	L	L	H	L	L	H	unc
L	L	L	H	L	H	L	DB31
L	L	L	H	L	H	H	2-bit
L	L	L	H	H	L	L	unc
L	L	L	H	H	L	H	2-bit
L	L	L	H	H	H	L	2-bit
L	L	L	H	H	H	H	DB30
L	L	H	L	L	L	L	2-bit
L	L	H	L	L	L	H	unc
L	L	H	L	L	H	L	DB29
L	L	H	L	L	H	H	2-bit
L	L	H	L	H	L	L	DB28
L	L	H	L	H	L	H	2-bit
L	L	H	L	H	H	L	2-bit
L	L	H	L	H	H	H	DB27
L	L	H	H	L	L	L	DB26
L	L	H	H	L	L	H	2-bit
L	L	H	H	L	H	L	2-bit
L	L	H	H	L	H	H	DB25
L	L	H	H	H	L	L	2-bit
L	L	H	H	H	L	H	DB24
L	L	H	H	H	H	L	unc
L	L	H	H	H	H	H	2-bit

Syndrome Bits							Error
6	5	4	3	2	1	0	
L	H	L	L	L	L	L	2-bit
L	H	L	L	L	L	H	unc
L	H	L	L	L	H	L	DB7
L	H	L	L	L	H	H	2-bit
L	H	L	L	H	L	L	DB6
L	H	L	L	H	L	H	2-bit
L	H	L	L	H	H	L	2-bit
L	H	L	L	H	H	H	DB5
L	H	L	H	L	L	L	DB4
L	H	L	H	L	L	H	2-bit
L	H	L	H	L	H	L	2-bit
L	H	L	H	L	H	H	DB3
L	H	L	H	H	L	L	2-bit
L	H	L	H	H	L	H	DB2
L	H	L	H	H	H	L	unc
L	H	L	H	H	H	H	2-bit
L	H	H	L	L	L	L	DB0
L	H	H	L	L	L	H	2-bit
L	H	H	L	L	H	L	2-bit
L	H	H	L	L	H	H	unc
L	H	H	L	H	L	L	2-bit
L	H	H	L	H	L	H	DB1
L	H	H	L	H	H	L	unc
L	H	H	L	H	H	H	2-bit
L	H	H	H	L	L	L	2-bit
L	H	H	H	L	L	H	unc
L	H	H	H	L	H	L	unc
L	H	H	H	L	H	H	2-bit
L	H	H	H	H	L	L	unc
L	H	H	H	H	L	H	2-bit
L	H	H	H	H	H	L	2-bit
L	H	H	H	H	H	H	CB6

Syndrome Bits							Error
6	5	4	3	2	1	0	
H	L	L	L	L	L	L	2-bit
H	L	L	L	L	L	H	unc
H	L	L	L	L	H	L	unc
H	L	L	L	L	H	H	2-bit
H	L	L	L	H	L	L	unc
H	L	L	L	H	L	H	2-bit
H	L	L	L	H	H	L	2-bit
H	L	L	L	H	H	H	unc
H	L	L	H	L	L	L	unc
H	L	L	H	L	L	H	2-bit
H	L	L	H	L	H	L	2-bit
H	L	L	H	L	H	H	DB15
H	L	L	H	H	L	L	2-bit
H	L	L	H	H	L	H	unc
H	L	L	H	H	H	L	DB14
H	L	L	H	H	H	H	2-bit
H	L	H	L	L	L	L	unc
H	L	H	L	L	L	H	2-bit
H	L	H	L	L	H	L	2-bit
H	L	H	L	L	H	H	DB13
H	L	H	L	H	L	L	2-bit
H	L	H	L	H	L	H	DB12
H	L	H	L	H	H	L	DB11
H	L	H	L	H	H	H	2-bit
H	L	H	H	L	L	L	2-bit
H	L	H	H	L	L	H	DB10
H	L	H	H	L	H	L	DB9
H	L	H	H	L	H	H	2-bit
H	L	H	H	H	L	L	DB8
H	L	H	H	H	L	H	2-bit
H	L	H	H	H	H	L	2-bit
H	L	H	H	H	H	H	CB5

Syndrome Bits							Error
6	5	4	3	2	1	0	
H	H	L	L	L	L	L	unc
H	H	L	L	L	L	H	2-bit
H	H	L	L	L	H	L	2-bit
H	H	L	L	L	H	H	DB23
H	H	L	L	H	L	L	2-bit
H	H	L	L	H	L	H	DB22
H	H	L	L	H	H	L	DB21
H	H	L	L	H	H	H	2-bit
H	H	L	H	L	L	L	2-bit
H	H	L	H	L	L	H	DB20
H	H	L	H	L	H	L	DB19
H	H	L	H	L	H	H	2-bit
H	H	L	H	H	L	L	DB18
H	H	L	H	H	L	H	2-bit
H	H	L	H	H	H	L	2-bit
H	H	L	H	H	H	H	CB4
H	H	H	L	L	L	L	2-bit
H	H	H	L	L	L	H	DB16
H	H	H	L	L	H	L	unc
H	H	H	L	L	H	H	2-bit
H	H	H	L	H	L	L	DB17
H	H	H	L	H	L	H	2-bit
H	H	H	L	H	H	L	2-bit
H	H	H	L	H	H	H	CB3
H	H	H	H	L	L	L	unc
H	H	H	H	L	L	H	2-bit
H	H	H	H	L	H	L	2-bit
H	H	H	H	L	H	H	CB2
H	H	H	H	H	L	L	2-bit
H	H	H	H	H	L	H	CB1
H	H	H	H	H	H	L	CB0
H	H	H	H	H	H	H	none

CB X = error in check bit X
DB Y = error in data bit Y
2-bit = double-bit error
unc = uncorrectable multibit error

7

TABLE VI. Read-Modify-Write Function

MEMORY CYCLE	EDAC FUNCTION	CONTROL S1	CONTROL S0	BYTEn†	$\overline{OEB}$n†	DB OUTPUT LATCH $\overline{LEDBO}$	CHECK I/O	CB CONTROL	ERROR FLAG $\overline{ERR}$ $\overline{MERR}$
Read	Read & Flag	H	L	Input	H	X	Input	H	Enabled
Read	Latch input data & check bits	H	H	Input data latched	H	L	Input check word latched	H	Enabled
Read	Latch corrected data word into output latch	H	H	Output data word latched	H	H	Hi-Z	H	Enabled
							Output Syndrome bits	L	
Modify /write	Modify appropriate byte or bytes & generate new check word	L	L	Input modified BYTE0	H	H	Output check word	L	H H
				Ouput unchanged BYTE0	L				

†$\overline{OEB}$0 controls DB_0–DB_7 (BYTE0), $\overline{OEB}$1 controls DB_8–DB_{15} (BYTE1), $\overline{OEB}$2 controls DB16–DB23 (BYTE2), $\overline{OEB}$3 controls DB24–DB31 (BYTE3).

Read-Modify-Write (Byte Control) Operations

The DP8402A and DP8403 devices are capable of byte-write operations. The 39-bit word from memory must first be latched into the DB and CB input latches. This is easily accomplished by switching from the read and flag mode (S1 = H, SO = L) to the latch input mode (S1 = H, S0 = H). The EDAC will then make any corrections, if necessary, to the data word and place it at the input of the output data latch. This data word must then be latched into the output data latch by taking $\overline{LEDBO}$ from a low to a high.

Byte control can now be employed on the data word through the $\overline{OEB}$0 through $\overline{OEB}$3 controls. $\overline{OEB}$0 controls DB0–DB7 (byte 0), $\overline{OEB}$1 controls DB8–DB15 (byte 1), $\overline{OEB}$2 controls DB16–DB23 (byte 2), and $\overline{OEB}$3 controls DB24–DB31 (byte 3). Placing a high on the byte control will disable the output and the user can modify the byte. If a low is placed on the byte control, then the original byte is allowed to pass onto the data bus unchanged. If the original data word is altered through byte control, a new check word must be generated before it is written back into memory. This is easily accomplished by taking control S1 and S0 low. Table VI lists the read-modify-write functions.

Diagnostic Operations

The DP8402A thru DP8405 are capable of diagnostics that allow the user to determine whether the EDAC or the memory is failing. The diagnostic function tables will help the user to see the possibilities for diagnostic control.

In the diagnostic mode (S1 = L, S0 = H), the checkword is latched into the input latch while the data input remains transparent. This lets the user apply various data words against a fixed known checkword. If the user applies a diagnostic data word with an error in any bit location, the $\overline{ERR}$ flag should be low. If a diagnostic data word with two errors in any bit location is applied, the $\overline{MERR}$ flag should be low. After the checkword is latched into the input latch, it can be verified by taking $\overline{OECB}$ low. This outputs the latched checkword. With the DP8402A and DP8403, the diagnostic data word can be latched into the output data latch and verified. It should be noted that the DP8404 and DP8405 do not have this pass-through capability because they do not contain an output data latch. By changing from the diagnostic mode (S1 = L, S0 = H) to the correction mode (S1 = H, S0 = H), the user can verify that the EDAC will correct the diagnostic data word. Also, the syndrome bits can be produced to verify that the EDAC pinpoints the error location. Table VII DP8402A and DP8403 and Table VIII DP8404 and DP8405 list the diagnostic functions.

TABLE VII. DP8402A, DP8403 Diagnostic Function

EDAC FUNCTION	CONTROL S1	CONTROL S0	DATA I/O	DB BYTE CONTROL $\overline{OEBn}$	DB OUTPUT LATCH $\overline{LEDBO}$	CHECK I/O	CB CONTROL $\overline{OECB}$	ERROR FLAGS $\overline{ERR}$	ERROR FLAGS $\overline{MERR}$
Read & flag	H	L	Input correct data word	H	X	Input correct check bits	H	H	H
Latch input check word while data input latch remains transparent	L	H	Input diagnostic data word†	H	L	Input check bits latched	H	Enabled	
Latch diagnostic data word into output latch	L	H	Input diagnostic data word†	H	H	Output latched check bits	L	Enabled	
						Hi-Z	H		
Latch diagnostic data word into input latch	H	H	Input diagnostic data word latched	H	H	Output syndrome bits	L	Enabled	
						Hi-Z	H		
Output diagnostic data word & syndrome bits	H	H	Output diagnostic data word	L	H	Output syndrome bits	L	Enabled	
						Hi-Z	H		
Output corrected diagnostic data word & output syndrome bits	H	H	Output corrected diagnostic data word	L	L	Output syndrome bits	L	Enabled	
						Hi-Z	H		

†Diagnostic data is a data word with an error in one bit location except when testing the $\overline{MERR}$ error flag. In this case, the diagnostic data word will contain errors in two bit locations.

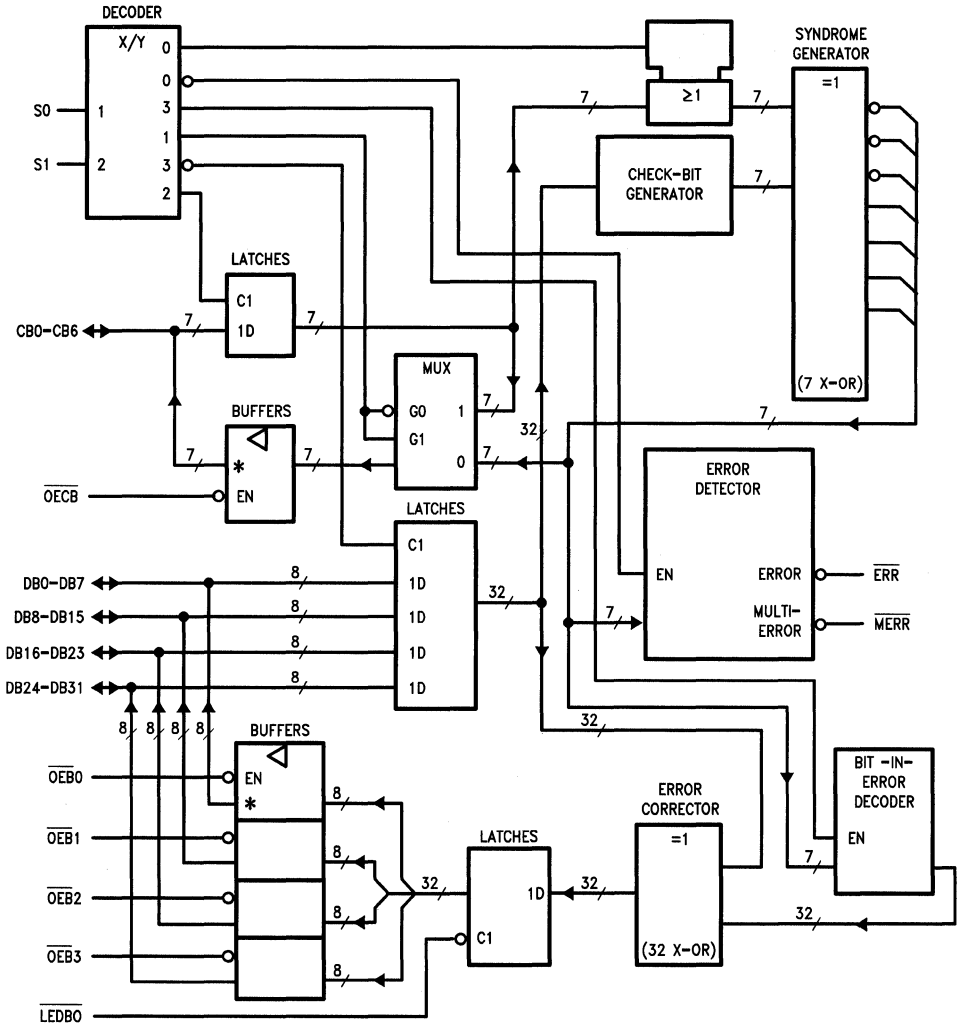
TABLE VIII. DP8404, DP8405 Diagnostic Function

EDAC FUNCTION	CONTROL S1	CONTROL S0	DATA I/O	DB CONTROL $\overline{OEDB}$	CHECK I/O	DB CONTROL $\overline{OECB}$	ERROR FLAGS $\overline{ERR}$	ERROR FLAGS $\overline{MERR}$
Read & flag	H	L	Input correct data word	H	Input correct check bits	H	H	H
Latch input check bits while data input latch remains transparent	L	H	Input diagnostic data word†	H	Input check bits latched	H	Enabled	
Output input check bits	L	H	Input diagnostic data word†	H	Output input check bits	L	Enabled	
Latch diagnostic data into input latch	H	H	Input diagnostic data word latched	H	Output syndrome bits	L	Enabled	
					Hi-Z	H		
Output corrected diagnostic data word	H	H	Output corrected diagnostic data word	L	Output syndrome bits	L	Enabled	
					Hi-Z	H		

†Diagnostic data is a data word with an error in one bit location except when testing the $\overline{MERR}$ error flag. In this case, the diagnostic data word will contain errors in two bit locations.

7

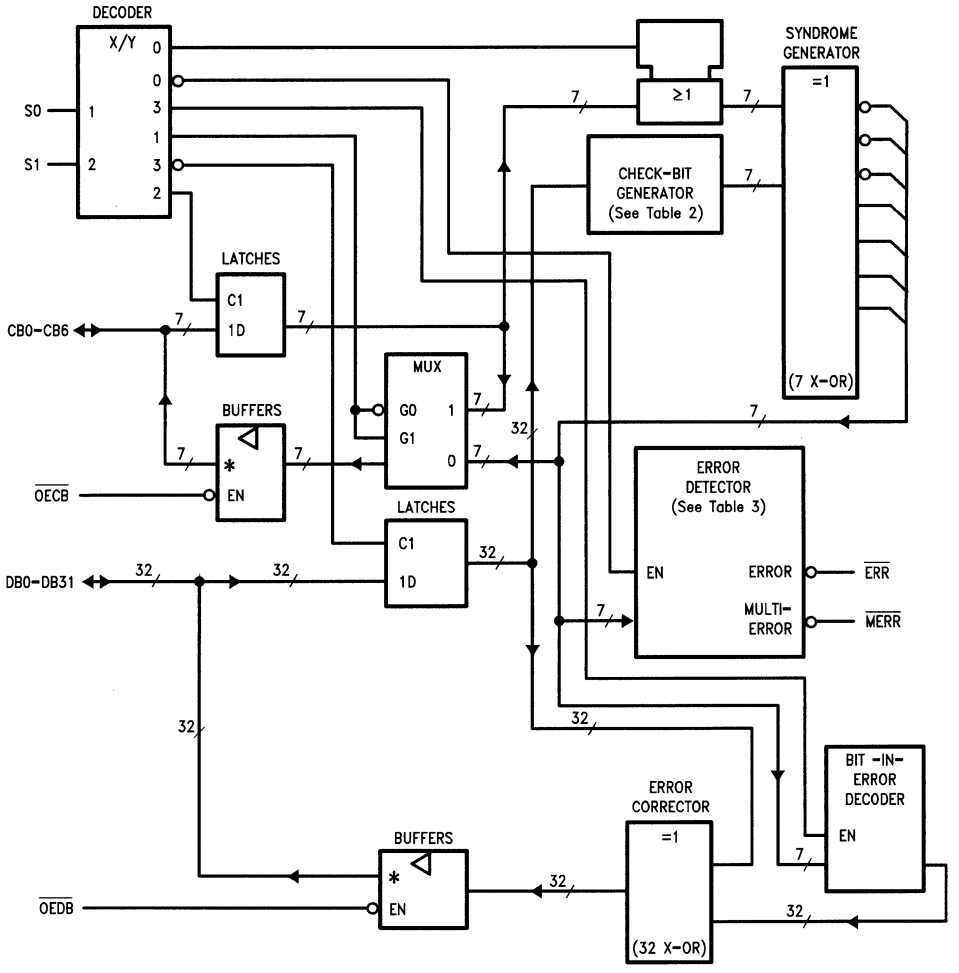
DP8402A, DP8403 Logic Diagram (Positive Logic)



DP8402A HAS TRI-STATE (▽) CHECK-BIT AND DATA OUTPUTS.
DP8403 HAS OPEN-COLLECTOR (◇) CHECK-BIT AND DATA OUTPUTS.

TL/F/8535-4

DP8404, DP8405 Logic Diagram (Positive Logic)



DP8404 HAS TRI-STATE (▽) CHECK-BIT AND DATA OUTPUTS.
DP8405 HAS OPEN-COLLECTOR (◇) CHECK-BIT AND DATA OUTPUTS.

TL/F/8535-5

7

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Over Operating Free-Air Temperature Range (unless otherwise noted)

Supply Voltage, V_{CC} (See Note 1) 7V

Input Voltage: CB and DB 5.5V

All Others 7V

Operating Free-Air Temperature: Military −55°C to +125°C

Commercial 0° to +70°C

Storage Temperature Range −65°C to +150°C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Military			Commercial			Units
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	Supply Voltage		4.5	5	5.5	4.5	5	5.5	V
V_{IH}	High-Level Input Voltage		2			2			V
V_{IL}	Low-Level Input Voltage				0.8			0.8	V
I_{OH}	High-Level Output Current	$\overline{ERR}$ Or $\overline{MERR}$			−0.4			−0.4	mA
		DB Or CB DP8402A, DP8404			−1			−2.6	
I_{OL}	Low-Level Output Current	$\overline{ERR}$ Or $\overline{MERR}$			4			8	mA
		DB or CB			12			24	
t_w	Pulse Duration	$\overline{LEDBO}$ Low	25			25			ns
t_{su}	Setup Time	(1) Data And Check Word Before S0 ↑ (S1 = H)	15			10			ns
		(2) SO High Before $\overline{LEDBO}$ ↑ (S1 = H)†	45			45			
		(3) $\overline{LEDBO}$ High Before The Earlier of S0 ↓ or S1 ↓ †	0			0			
		(4) $\overline{LEDBO}$ High Before S1 ↑ (S0 = H)	0			0			
		(5) Diagnostic Data Word Before S1 ↑ (S0 = H)	15			10			
		(6) Diagnostic Check Word Before The Later Of S1 ↓ or S0 ↑	15			10			
		(7) Diagnostic Data Word Before $\overline{LEDBO}$ ↑ (S1 = L and S0 = H)‡	25			20			
t_h	Hold Time	(8) Read-Mode, S0 Low And S1 High	35			30			ns
		(9) Data And Check Word After S0 ↑ (S1 = H)	20			15			
		(10) Data Word After S1 ↑ (S0 = H)	20			15			
		(11) Check Word After The Later of S1 ↓ or S0 ↑	20			15			
		(12) Diagnostic Data Word After $\overline{LEDBO}$ ↑ (S1 = L And S0 = H)‡	0			0			
t_{corr}	Correction Time (see *Figure 1*)*		65			58			ns
T_A	Operating Free-Air Temperature		−55		125	0		70	°C

*This specification may be interpreted as the maximum delay to guarantee valid corrected data at the output and includes the t_{su} setup delay.

†These times ensure that corrected data is saved in the output data latch.

‡These times ensure that the diagnostic data word is saved in the output data latch.

DP8402A, DP8404 Electrical Characteristics

Over Recommended Operating Free-Air Temperature Range (unless otherwise noted)

Symbol	Parameter	Test Conditions	Military			Commercial			Units
			Min	Typ†	Max	Min	Typ†	Max	
V_{IK}		V_{CC} = 4.5V, I_I = −18 mA			−1.5			−1.5	V
V_{OH}	All outputs	V_{CC} = 4.5V to 5.5V, I_{OH} = − 0.4 mA	V_{CC}−2			V_{CC}−2			V
	DB or CB	V_{CC} = 4.5V, I_{OH} = −1 mA	2.4	3.3					
		V_{CC} = 4.5V, I_{OH} = −2.6 mA				2.4	3.2		
V_{OL}	$\overline{ERR}$ or $\overline{MERR}$	V_{CC} = 4.5V, I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
		V_{CC} = 4.5V, I_{OL} = 8 mA					0.35	0.5	
	DB or CB	V_{CC} = 4.5V, I_{OL} = 12 mA		0.25	0.4		0.25	0.4	
		V_{CC} = 4.5V, I_{OL} = 24 mA					0.35	0.5	
I_I	S0 or S1	V_{CC} = 5.5V, V_I = 7V			0.1			0.1	mA
	All others	V_{CC} = 5.5V, V_I = 5.5V			0.1			0.1	
I_{IH}	S0 or S1	V_{CC} = 5.5V, V_I = 2.7V			20			20	µA
	All others‡				20			20	
I_{IL}	S0 or S1	V_{CC} = 5.5V, V_I = 0.4V			−0.4			−0.4	mA
	All others‡				−0.1			−0.1	
I_O§		V_{CC} = 5.5V, V_O = 2.25V	−30		−112	−30		−112	mA
I_{CC}		V_{CC} = 5.5V, (See Note 1)		150	250		150	250	mA

DP8403, DP8405 Electrical Characteristics

Over Recommended Operating Free-Air Temperature Range (unless otherwise noted)

Symbol	Parameter	Test Conditions	Military			Commercial			Units
			Min	Typ†	Max	Min	Typ†	Max	
V_{IK}		V_{CC} = 4.5V, I_I = −18 mA			−1.5			−1.5	V
V_{OH}	$\overline{ERR}$ or $\overline{MERR}$	V_{CC} = 4.5V to 5.5V, I_{OH} = −0.4 mA	V_{CC}−2			V_{CC}−2			V
I_{OH}	DB or CB	V_{CC} = 4.5V, V_{OH} = 5.5V			0.1			0.1	mA
V_{OL}	$\overline{ERR}$ or $\overline{MERR}$	V_{CC} = 4.5V, I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
		V_{CC} = 4.5V, I_{OL} = 8 mA					0.35	0.5	
	DB or CB	V_{CC} = 4.5V, I_{OL} = 12 mA		0.25	0.4		0.25	0.4	
		V_{CC} = 4.5V, I_{OL} = 24 mA					0.35	0.5	
I_I	S0 or S1	V_{CC} = 5.5V, V_I = 7V							mA
	All others	V_{CC} = 5.5V, V_I = 5.5V							
I_{IH}	S0 or S1	V_{CC} = 5.5V, V_I = 2.7V							µA
	All others‡								
I_{IL}	S0 or S1	V_{CC} = 5.5V, V_I = 0.4V							mA
	All others‡								
I_O§	$\overline{ERR}$ or $\overline{MERR}$	V_{CC} = 5.5V, V_O = 2.25V	−30		−112	−30		−112	mA
I_{CC}		V_{CC} = 5.5V, (See Note 1)		150			150		mA

†All typical values are at V_{CC} = 5V, T_A = +25°C.

‡For I/O ports (Q_A through Q_H), the parameters I_{IH} and I_{IL} include the off-state output current.

§The output conditions have been chosen to produce a current that closely approximates one half of the true short-circuit output current, I_{OS}.

Note 1: I_{CC} is measured with S0 and S1 at 4.5V and all CB and DB pins grounded.

7

DP8402A Switching Characteristics

V_{CC} = 4.5V to 5.5V, C_L = 50 pF, T_A = Min to Max (unless otherwise noted)

Symbol	From (Input)	To (Output)	Test Conditions	Military		Commercial		Units
				Min	Max	Min	Max	
t_{pd}	DB and CB	$\overline{ERR}$	S1 = H, S0 = L, R_L = 500Ω	10	43	10	40	ns
	DB	$\overline{ERR}$	S1 = L, S0 = H, R_L = 500Ω	10	43	10	40	
t_{pd}	DB and CB	$\overline{MERR}$	S1 = H, S0 = L, R_L = 500Ω	15	67	15	55	ns
	DB	$\overline{MERR}$	S1 = L, S0 = H, R_L = 500Ω	15	67	15	55	
t_{pd}	S0 ↓ and S1 ↓	CB	R1 = R2 = 500Ω	10	60	10	48	ns
t_{pd}	DB	CB	S1 = L, S0 = L, R1 = R2 = 500Ω	10	60	10	48	ns
t_{pd}	$\overline{LEDB0}$ ↓	DB	S1 = X, S0 = H, R1 = R2 = 500Ω	7	35	7	30	ns
t_{pd}	S1 ↑	CB	S0 = H, R1 = R2 = 500Ω	10	60	10	50	ns
t_{en}	$\overline{OECB}$ ↓	CB	S0 = H, S1 = X, R1 = R2 = 500Ω	2	30	2	25	ns
t_{dis}	$\overline{OECB}$ ↑	CB	S0 = H, S1 = X, R1 = R2 = 500Ω	2	30	2	25	ns
t_{en}	$\overline{OEB0}$ thru $\overline{OEB3}$ ↓	DB	S0 = H, S1 = X, R1 = R2 = 500Ω	2	30	2	25	ns
t_{dis}	$\overline{OEB0}$ thru $\overline{OEB3}$ ↑	DB	S0 = H, S1 = X, R1 = R2 = 500Ω	2	30	2	25	ns

DP8403 Switching Characteristics

V_{CC} = 4.5V to 5.5V, C_L = 50 pF, T_A = Min to Max (unless otherwise noted)

Symbol	From (Input)	To (Output)	Test Conditions	Military			Commercial			Units
				Min	Typ†	Max	Min	Typ†	Max	
t_{pd}	DB and CB	$\overline{ERR}$	S1 = H, S0 = L, R_L = 500Ω		26			26		ns
	DB	$\overline{ERR}$	S1 = L, S0 = H, R_L = 500Ω		26			26		
t_{pd}	DB and CB	$\overline{MERR}$	S1 = H, S0 = L, R_L = 500Ω		40			40		ns
			S1 = L, S0 = H, R_L = 500Ω		40			40		
t_{pd}	S0 ↓ and S1 ↓	CB	R_L = 680Ω		40			40		ns
t_{pd}	DB	CB	S1 = L, S0 = L, R_L = 680Ω		40			40		ns
t_{pd}	$\overline{LEDB0}$ ↓	DB	S1 = X, S0 = H, R_L = 680Ω		26			26		ns
t_{pd}	S1 ↑	CB	S0 = H, R_L = 680Ω		40			40		ns
t_{PLH}	$\overline{OECB}$ ↑	CB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns
t_{PHL}	$\overline{OECB}$ ↓	CB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns
t_{PLH}	$\overline{OEB0}$ thru $\overline{OEB3}$ ↑	DB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns
t_{PHL}	$\overline{OEB0}$ thru $\overline{OEB3}$ ↓	DB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns

†All typical values are at V_{CC} = 5V, T_A = +25°C.

DP8404 Switching Characteristics, V_{CC} = 4.5V to 5.5V, C_L = 50 pF, T_A = Min to Max

Symbol	From (Input)	To (Output)	Test Conditions	Military			Commercial			Units
				Min	Typ†	Max	Min	Typ†	Max	
t_{pd}	DB and CB	$\overline{ERR}$	S1 = H, S0 = L, R_L = 500Ω		26			26		ns
			S1 = L, S0 = H, R_L = 500Ω		26			26		
t_{pd}	DB and CB	$\overline{MERR}$	S1 = H, S0 = L, R_L = 500Ω		40			40		ns
			S1 = L, S0 = H, R_L = 500Ω		40			40		
t_{pd}	S0 ↓ and S1 ↓	CB	R1 = R2 = 500Ω		35			35		ns
t_{pd}	DB	CB	S1 = L, S0 = L, R1 = R2 = 500Ω		35			35		ns
t_{pd}	S1 ↑	CB	S0 = H, R1 = R2 = 500Ω		35			35		ns
t_{en}	$\overline{OECB}$ ↓	CB	S1 = X, S0 = H, R1 = R2 = 500Ω		18			18		ns
t_{dis}	$\overline{OECB}$ ↑	CB	S1 = X, S0 = H, R1 = R2 = 500Ω		18			18		ns
t_{en}	$\overline{OECB}$ ↓	DB	S1 = X, S0 = H, R1 = R2 = 500Ω		18			18		ns
t_{dis}	$\overline{OECB}$ ↑	DB	S1 = X, S0 = H, R1 = R2 = 500Ω		18			18		ns

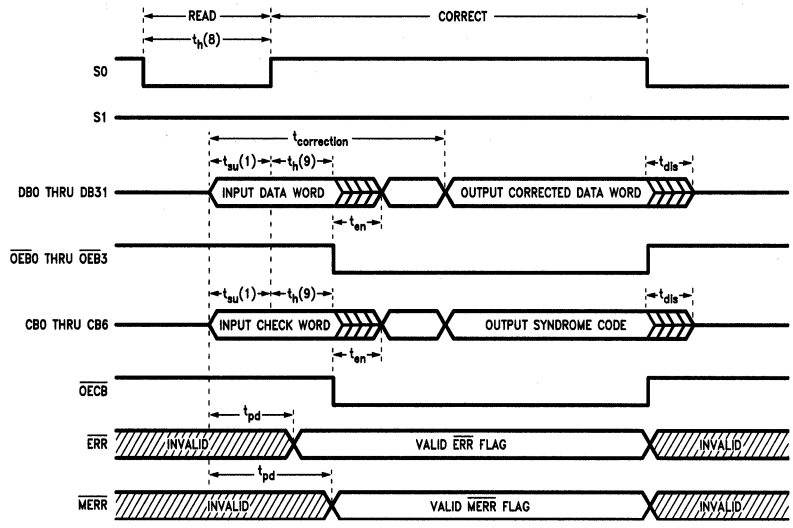
DP8405 Switching Characteristics, V_{CC} = 4.5V to 5.5V, C_L = 50 pF, T_A = Min to Max

Symbol	From (Input)	To (Output)	Test Conditions	Military			Commercial			Units
				Min	Typ†	Max	Min	Typ†	Max	
t_{pd}	DB and CB	$\overline{ERR}$	S1 = H, S0 = L, R_L = 500Ω		26			26		ns
	DB	$\overline{ERR}$	S1 = L, S0 = H, R_L = 500Ω		26			26		
t_{pd}	DB and CB	$\overline{MERR}$	S1 = H, S0 = L, R_L = 500Ω		40			40		ns
			S1 = L, S0 = H, R_L = 500Ω		40			40		
t_{pd}	S0 ↓ and S1 ↓	CB	R_L = 680Ω		40			40		ns
t_{pd}	DB	CB	S1 = L, S0 = L, R_L = 680Ω		40			40		ns
t_{pd}	S1 ↑	DB	S0 = H, R_L = 680Ω		40			40		ns
t_{PLH}	$\overline{OECB}$ ↑	CB	S1 = X, S0 = H, R_L = 500Ω		24			24		ns
t_{PHL}	$\overline{OECB}$ ↓	CB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns
t_{PLH}	$\overline{OEDB}$ ↑	DB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns
t_{PHL}	$\overline{OEDB}$ ↓	DB	S1 = X, S0 = H, R_L = 680Ω		24			24		ns

†All typical values are at V_{CC} = 5V, T_A = +25°C.

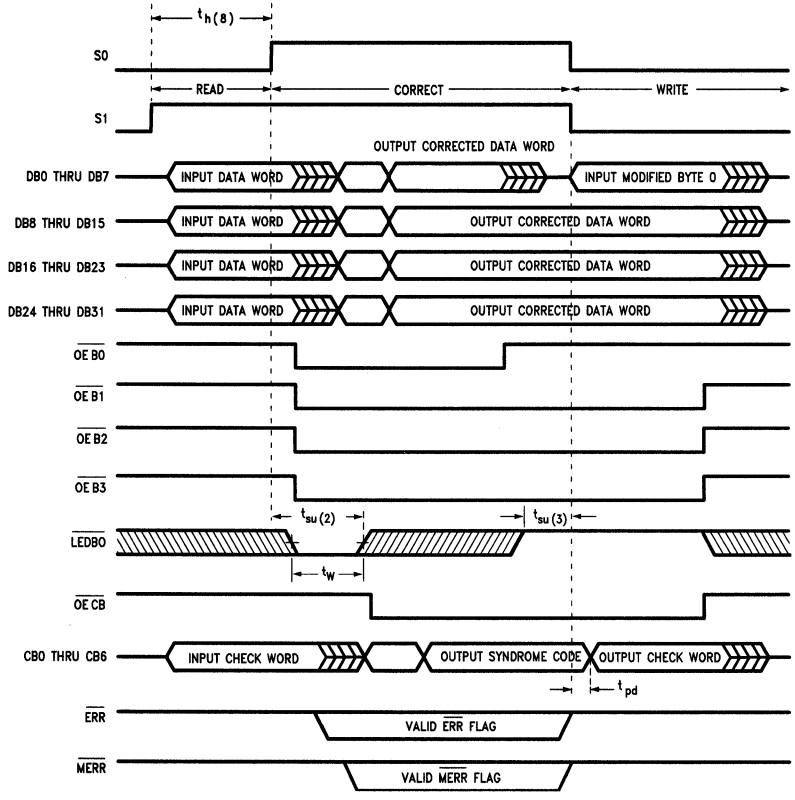
7

Switching Waveforms



TL/F/8535–6

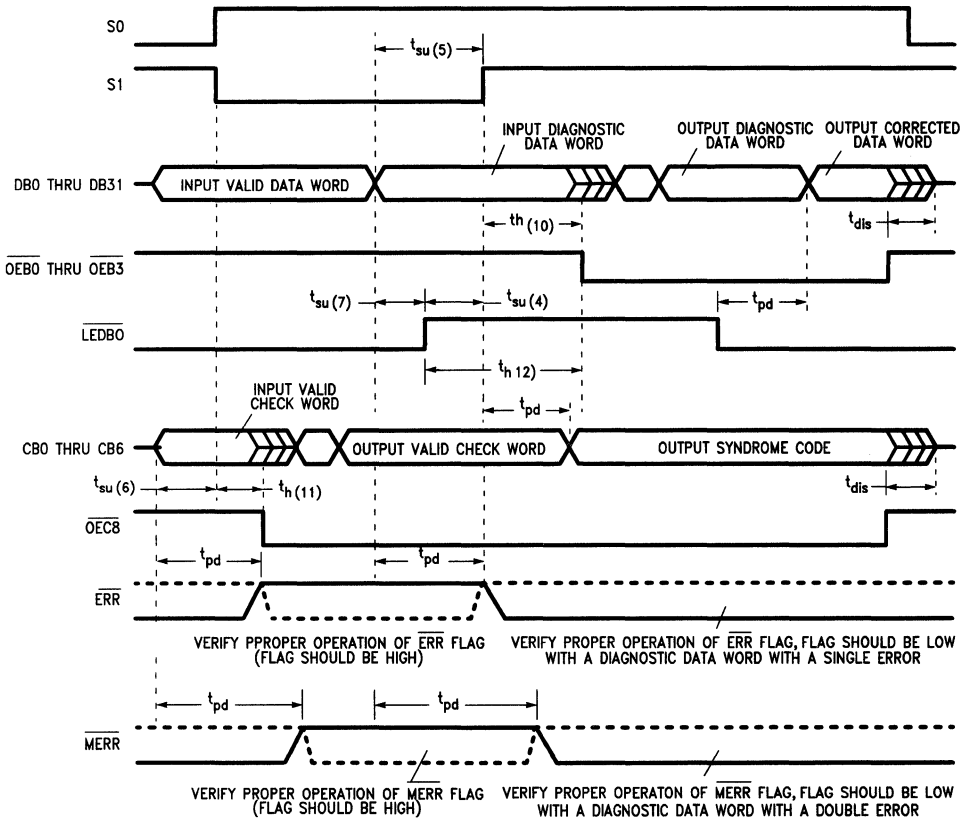
FIGURE 1. Read, Flag, and Correct Mode



TL/F/8535–7

FIGURE 2. Read, Correct Modify Mode

Switching Waveforms (Continued)

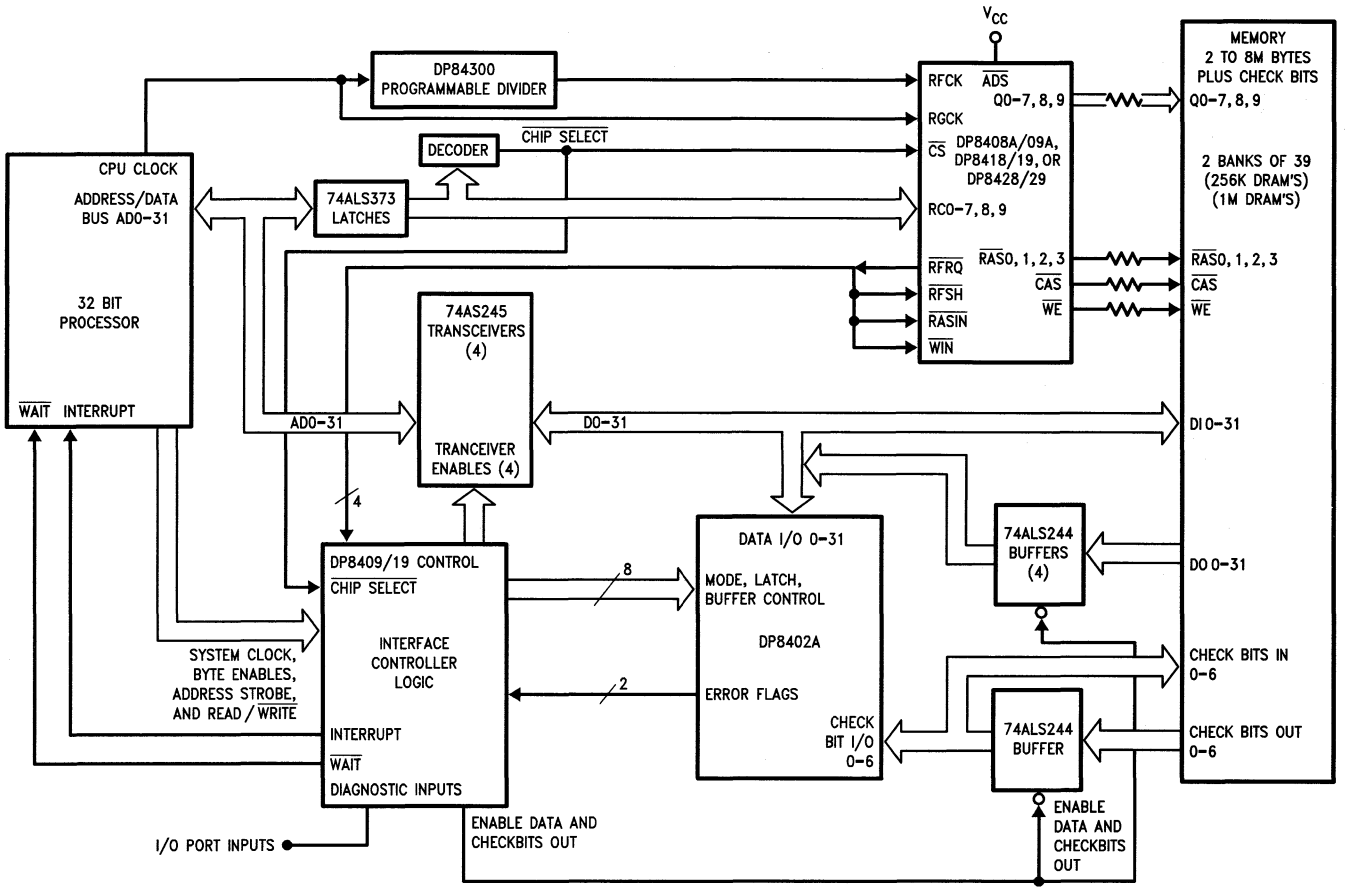


TL/F/8535–8

FIGURE 3. Diagnostic Mode

7

DP8402A System Diagram



TL/F/8535-9

National Semiconductor Corporation

DP8409A Multi-Mode Dynamic RAM Controller/Driver

General Description

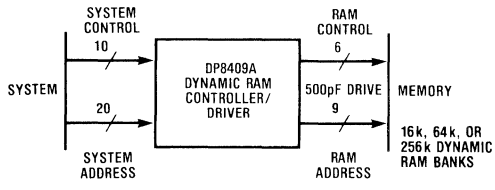
Dynamic memory system designs, which formerly required several support chips to drive the memory array, can now be implemented with a single IC . . . the DP8409A Multi-Mode Dynamic RAM Controller/Driver. The DP8409A is capable of driving all 16k and 64k Dynamic RAMs (DRAMs) as well as 256k DRAMs. Since the DP8409A is a one-chip solution (including capacitive-load drivers), it minimizes propagation delay skews, the major performance disadvantage of multiple-chip memory drive and control.

The DP8409A's 8 modes of operation offer a wide selection of DRAM control capabilities. Memory access may be controlled externally or on-chip automatically; an on-chip refresh counter makes refreshing (either externally or automatically controlled) less complicated; and automatic memory initialization is both simple and fast.

The DP8409A is a 48-pin DRAM Controller/Driver with 9 multiplexed address outputs and 6 control signals. It consists of two 9-bit address latches, a 9-bit refresh counter, and control logic. All output drivers are capable of driving 500 pF loads with propagation delays of 25 ns. The DP8409A timing parameters are specified driving the typical load capacitance of 88 DRAMs, including trace capacitance.

The DP8409A has 3 mode-control pins: M2, M1, and M0, where M2 is in general $\overline{\text{REFRESH}}$. These 3 pins select 8 modes of operation. Inputs B1 and B0 in the memory access modes (M2 = 1), are select inputs which select one of four $\overline{\text{RAS}}$ outputs. During normal access, the 9 address outputs can be selected from the Row Address Latch or the Column Address Latch. During refresh, the 9-bit on-chip refresh counter is enabled onto the address bus and in this mode all $\overline{\text{RAS}}$ outputs are selected, while $\overline{\text{CAS}}$ is inhibited.

The DP8409A can drive up to 4 banks of DRAMs, with each bank comprised of 16k's, 64k's, or 256k's. Control signal outputs $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{WE}}$ are provided with the same drive capability. Each $\overline{\text{RAS}}$ output drives one bank of DRAMs so that the four $\overline{\text{RAS}}$ outputs are used to select the banks, while $\overline{\text{CAS}}$, $\overline{\text{WE}}$, and the multiplexed addresses can be connected to all of the banks of DRAMs. This leaves the non-selected banks in the standby mode (less than one tenth of the operating power) with the data outputs in TRI-STATE®. Only the bank with its associated $\overline{\text{RAS}}$ low will be written to or read from.



TL/F/8409–1

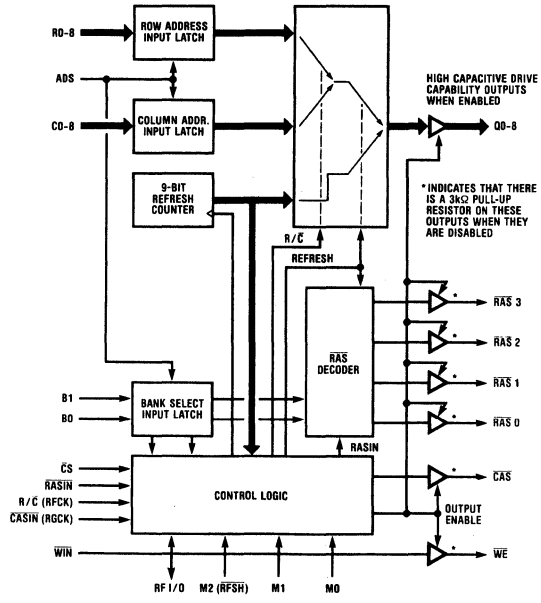
Operational Features

- All DRAM drive functions on one chip—minimizes skew on outputs, maximizes AC peformance
- On-chip capacitive-load drives (specified to drive up to 88 DRAMs)
- Drives directly all 16k, 64k, and 256k DRAMs
- Capable of addressing 64k, 256k, or 1M words
- Propagation delays of 25 ns typical at 500 pF load
- $\overline{\text{CAS}}$ goes low automatically after column addresses are valid if desired
- Auto Access mode provides $\overline{\text{RAS}}$, row to column select, then $\overline{\text{CAS}}$ automatically and fast
- $\overline{\text{WE}}$ follows $\overline{\text{WIN}}$ unconditionally—offering READ, WRITE or READ-MODIFY-WRITE cycles
- On-chip 9-bit refresh counter with selectable End-of-Count (127, 255 or 511)
- End-of-Count indicated by RF I/O pin going low at 127, 255 or 511
- Low input on RF I/O resets 9-bit refresh counter
- $\overline{\text{CAS}}$ inhibited during refresh cycle
- Fall-through latches on address inputs controlled by ADS
- TRI-STATE outputs allow multi-controller addressing of memory
- Control output signals go high-impedance logic "1" when disabled for memory sharing
- Power-up: counter reset, control signals high, address outputs TRI-STATE, and End-of-Count set to 127

Mode Features

- 8 modes of operation: 3 access, 3 refresh, and 2 set-up
- 2 externally controlled modes: 1 access and 1 refresh (Modes 0, 4)
- 2 auto-access modes $\overline{\text{RAS}} \rightarrow \text{R}/\overline{\text{C}} \rightarrow \overline{\text{CAS}}$ automatic, with t_{RAH} = 20 or 30 ns minimum (Modes 5, 6)
- Auto-access mode allows Hidden Refreshing (Mode 5)
- Forced Refresh requested on RF I/O if no Hidden Refresh (Mode 5)
- Forced Refresh performed after system acknowledge of request (Mode 1)
- Automatic Burst Refresh mode stops at End-of-Count of 127, 255, or 511 (Mode 2)
- 2 All-$\overline{\text{RAS}}$ Acces modes externally or automatically controlled for memory initialization (Modes 3a, 3b)
- Automatic All-$\overline{\text{RAS}}$ mode with external 8-bit counter frees system for other set-up routines (Mode 3a)
- End-of-Count value of Refresh Counter set by B1 and B0 (Mode 7)

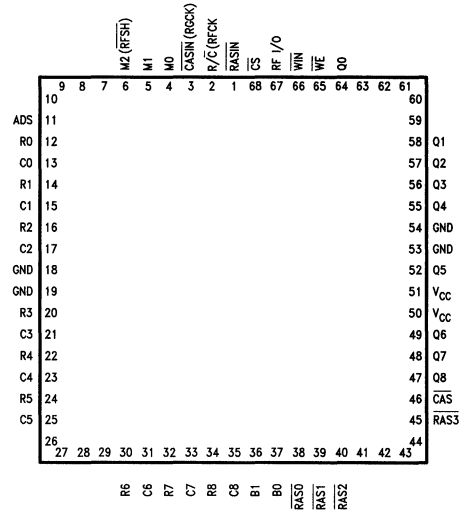
Block and Connection Diagrams



TL/F/8409-2

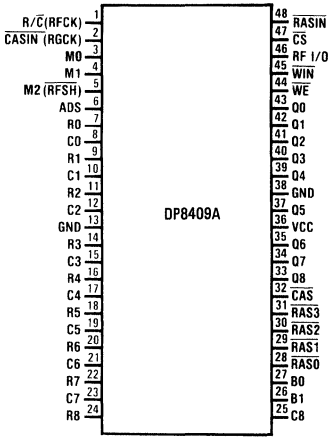
Order Number DP8409AD, DP8409AN, DP8409AN-3 or DP8409AV-2
See NS Package Number D48A, N48A or V68A

68 Pin PCC



TL/F/8409-3

Dual-In-Line Package



Top View

TL/F/8409-5

Pin Definitions

V_{CC}, GND, GND—V_{CC} = 5V ±5%. The three supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. There are also two ground pins to reduce the low level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 9 address bits change in the same direction simultaneously. A recommended solution would be a 1 μF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected as close as possible to pins 36 and 38 to reduce lead inductance. See figure below.

TL/F/8409-4

*Capacitor values should be chosen depending on the particular application.

R0–R8: Row Address Inputs.

C0–C8: Column Address Inputs.

Q0–Q8: Multiplexed Address Outputs—Selected from the Row Address Input Latch, the Column Address Input Latch, or the Refresh Counter.*

$\overline{RASIN}$: Row Address Strobe Input—Enables selected $\overline{RAS}_n$ output when M2 ($\overline{RFSH}$) is high, or all $\overline{RAS}_n$ outputs when $\overline{RFSH}$ is low.

R/$\overline{C}$ (RFCK)—In Auto-Refresh Mode this pin is the external Refresh Clock Input: one refresh cycle has to be performed each clock period. In all other modes it is Row/Column Select Input: selects either the row or column address input latch onto the output bus.

Pin Definitions (Continued)

TABLE I. DP8409A Mode Select Options

Mode	($\overline{RFSH}$) M2	M1	M0	Mode of Operation	Conditions
0	0	0	0	Externally Controlled Refresh	RF I/O = $\overline{EOC}$
1	0	0	1	Auto Refresh—Forced	RF I/O = Refresh Request ($\overline{RFRQ}$)
2	0	1	0	Internal Auto Burst Refresh	RF I/O = $\overline{EOC}$
3a	0	1	1	All $\overline{RAS}$ Auto Write	RF I/O = $\overline{EOC}$; All $\overline{RAS}$ Active
3b	0	1	1	Externally Controlled All $\overline{RAS}$ Access	All $\overline{RAS}$ Active
4	1	0	0	Externally Controlled Access	Active $\overline{RAS}$ Defined by Table II
5	1	0	1	Auto Access, Slow t_{RAH}, Hidden Refresh	Active $\overline{RAS}$ Defined by Table II
6	1	1	0	Auto Access, Fast t_{RAH}	Active $\overline{RAS}$ Defined by Table II
7	1	1	1	Set End of Count	See Table III for Mode 7

$\overline{CASIN}$ (RGCK)—In Auto-Refresh Mode, Auto Burst Mode, and All-$\overline{RAS}$ Auto-Write Mode, this pin is the $\overline{RAS}$ Generator Clock input. In all other modes it is $\overline{CASIN}$ (Column Address Strobe Input), which inhibits $\overline{CAS}$ output when high in Modes 4 and 3b. In Mode 6 it can be used to prolong $\overline{CAS}$ output.

ADS: Address (Latch) Strobe Input—Row Address, Column Address, and Bank Select Latches are fall-through with ADS high; Latches on high-to-low transition.

$\overline{CS}$: Chip Select Input—The TRI-STATE mode will Address Outputs and puts the control signal into a high-impedance logic "1" state when high (unless refreshing in one of the Refresh Modes). Enables all outputs when low.

M0, M1, M2: Mode Control Inputs—These 3 control pins determine the 8 major modes of operation of the DP8409A as depicted in Table I.

RF I/O—The I/O pin functions as a Reset Counter Input when set low from an external open-collector gate, or as a flag output. The flag goes active-low in Modes 0 and 2 when the End-of-Count output is at 127, 255, or 511 (see Table III). In Auto-Refresh Mode it is the Refresh Request output.

$\overline{WIN}$: Write Enable Input.

$\overline{WE}$: Write Enable Output—Buffered output from $\overline{WIN}$.*

$\overline{CAS}$: Column Address Strobe Output—In Modes 3a, 5, and 6, $\overline{CAS}$ transitions low following valid column address. In Modes 3b and 4, it goes low after R/$\overline{C}$ goes low, or follows $\overline{CASIN}$ going low if R/$\overline{C}$ is already low. $\overline{CAS}$ is high duing refresh.*

$\overline{RAS}$ 0-3: Row Address Strobe Outputs—Selects a memory bank decoded from B1 and B0 (see Table II), if $\overline{RFSH}$ is high. If $\overline{RFSH}$ is low, all banks are selected.*

B0, B1: Bank Select Inputs—Strobed by ADS. Decoded to enable one of the $\overline{RAS}$ outputs when $\overline{RASIN}$ goes low. Also used to define End-of-Count in Mode 7 (Table III).

Conditions for All Modes

INPUT ADDRESSING

The address block consists of a row-address latch, a column-address latch, and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses until after the valid address time, ADS can be permanently high. Otherwise ADS must go low while the addresses are still valid.

In normal memory access operation, $\overline{RASIN}$ and R/$\overline{C}$ are initially high. When the address inputs are enabled into the address latches, the row addresses appear on the Q outputs. The address strobe also inputs the bank-select address, (B0 and B1). If $\overline{CS}$ is low, all outputs are enabled. When $\overline{CS}$ is transitioned high, the address outputs go TRI-STATE and the control outputs first go high through a low impedance, and then are held by an on-chip high impedance. This allows output paralleling with other DP8409As for multi-addressing. All outputs go active about 50 ns after the chip is selected again. If $\overline{CS}$ is high, and a refresh cycle begins, all the outputs become active until the end of the refresh cycle.

DRIVE CAPABILITY

The DP8409A has timing parameters that are specified with up to 600 pF loads. In a typical memory system this is equivalent to about 88, 5V-only DRAMs, with trace lengths kept to a minimum. Therefore, the chip can drive four banks each of 16 or 22 bits, or two banks of 32 or 39 bits, or one bank of 64 or 72 bits.

Less loading will slightly reduce the timing parameters, and more loading will increase the timing parameters, according to the graph of *Figure 10*. The AC performance parameters are specified with the typical load capacitance of 88 DRAMs. This graph can be used to extrapolate the variations expected with other loading.

Because of distributed trace capacitance and inductance and DRAM input capacitance, current spikes can be created, causing overshoots and undershoots at the DRAM inputs that can change the contents of the DRAMs or even destroy them. To remove these spikes, a damping resistor (low inductance, carbon) can be inserted between the DP8409A driver outputs and the DRAMs, as close as possible to the DP8409A. The values of the damping resistors may differ between the different control outputs; $\overline{RAS}$s, $\overline{CAS}$, Q's, and $\overline{WE}$. The damping resistors should be determined by the first prototypes (not wire-wrapped due to the larger distributed capacitance and inductance). The best values for the damping resistors are the critical values giving a critically damped transition on the control outputs. Typical values for the damping resistors will be between 15Ω and 100Ω, the lower the loading the higher the value. (For more information, see AN-305 "Precautions to Take When Driving Memories.")

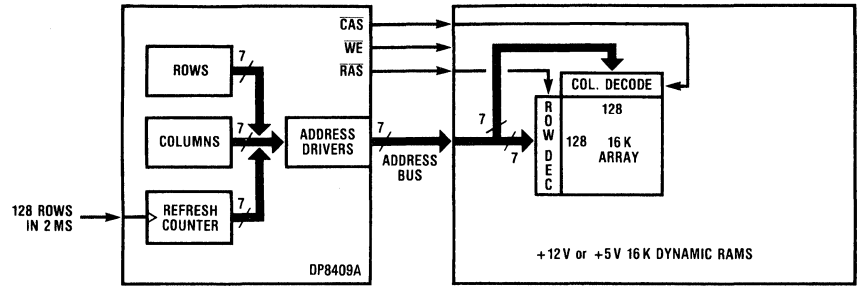
Conditions for All Modes (Continued)

DP8409A DRIVING ANY 16k OR 64k DRAMs

The DP8409A can drive any 16k or 64k DRAMs. All 16k DRAMs are basically the same configuration, including the newer 5V-only version. Hence, in most applications, different manufacturers' DRAMs are interchangeable (for the same supply-rail chips), and the DP8409A can drive all 16k DRAMs (see *Figure 1a*).

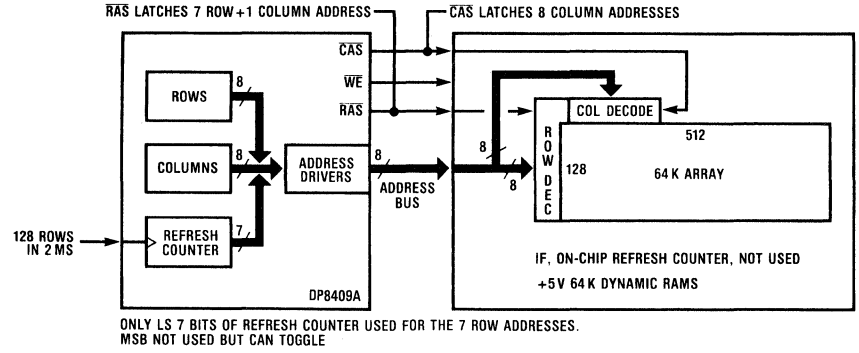
There are three basic configurations for the 5V-only 64k DRAMs: a 128-row by 512-column array with an on-RAM refresh counter, a 128-row by 512-column array with no on-RAM refresh counter, and a 256-row by 256-column array with no on-RAM refresh counter. The DP8409A can drive all three configurations, and at the same time allows them all to be interchangeable (as shown in *Figures 1b* and *1c*), providing maximum flexibility in the choice of DRAMs. Since the 9-bit on-chip refresh counter can be used as a 7-bit refresh counter for the 128-row configuration, or as an 8-bit refresh counter for the 256-row configuration, the on-RAM refresh counter (if present) is never used. As long as 128 rows are refreshed every 2 ms (i.e. 256 rows in 4 ms) all DRAM types are correctly refreshed.

DP8409A Interface between System and DRAM Banks



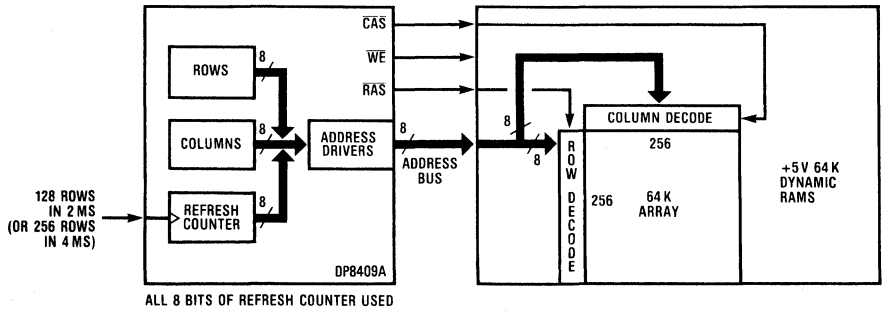
TL/F/8409–6

FIGURE 1a. DP8409A with any 16k DRAMs



TL/F/8409–7

FIGURE 1b. DP8409A with 128 Row x 512 Column 64k DRAM



TL/F/8409–8

FIGURE 1c. DP8409A with 256 x 256 Column 64k DRAM

Conditions for All Modes (Continued)

When the DP8409A is in a refresh mode, the RF I/O pin indicates that the on-chip refresh counter has reached its end-of-count. This end-of-count is selectable as 127, 255 or 512 to accommodate 16k, 64k or 256k DRAMs. Although the end-of-count may be chosen to be any of these, the counter always counts to 511 before rolling over to zero.

READ, WRITE, AND READ-MODIFY-WRITE CYCLES

The output signal, $\overline{WE}$, determines what type of memory access cycle the memory will perform. If $\overline{WE}$ is kept high while $\overline{CAS}$ goes low, a read cycle occurs. If $\overline{WE}$ goes low before $\overline{CAS}$ goes low, a write cycle occurs and data at DI (DRAM input data) is written into the DRAM as $\overline{CAS}$ goes low. If $\overline{WE}$ goes low later than t_{CWD} after $\overline{CAS}$ goes low, first a read occurs and DO (DRAM output data) becomes valid; then data DI is written into the same address in the DRAM when $\overline{WE}$ goes low. In this read-modify-write case, DI and DO cannot be linked together. The type of cycle is therefore controlled by $\overline{WE}$, which follows $\overline{WIN}$.

POWER-UP INITIALIZE

When V_{CC} is first applied to the DP8409A, an initialize pulse clears the refresh counter, the internal control flip-flops, and set the End-of-Count of the refresh counter to 127 (which may be changed via Mode 7). As V_{CC} increases to about 2.3V, it holds the output control signals at a level of one Schottky diode-drop below V_{CC}, and the output address to TRI-STATE. As V_{CC} increases above 2.3V, control of these outputs is granted to the system.

DP8409A Functional Mode Descriptions

Note: All delay parameters stated in text refer to the DP8409A. Substitute the respective delay numbers for the DP8409-2 or DP8409-3 when using these devices.

MODE 0—EXTERNALLY CONTROLLED REFRESH

Figure 2 is the Externally Controlled Refresh Timing. In this mode, the input address latches are disabled from the address outputs and the refresh counter is enabled. When $\overline{RAS}$ occurs, the enabled row in the DRAM is refreshed. In the Externally Controlled Refresh mode, all $\overline{RAS}$ outputs are enabled following $\overline{RASIN}$, and $\overline{CAS}$ is inhibited. This refreshes the same row in all four banks. The refresh counter increments when either $\overline{RASIN}$ or $\overline{RFSH}$ goes low-to-high after a refresh. RF I/O goes low when the count is 127, 255, or 511, as set by End-of-Count (see Table III), with $\overline{RASIN}$ and $\overline{RFSH}$ low. To reset the counter to all zeros, RF I/O is set low through an external open-collector driver.

During refresh, $\overline{RASIN}$ and RFSH must be skewed transitioning low such that the refresh address is valid on the address outputs of the controller before the $\overline{RAS}$ outputs go low. The amount of time that RFSH should go low before $\overline{RASIN}$ does depends on the capacitive loading of the address and $\overline{RAS}$ lines. For the load specified in the switching characteristics of this data sheet, 10 ns is sufficient. Refer to *Figure 2*.

To perform externally controlled burst refresh, $\overline{RASIN}$ is toggled while RFSH is held low. The refresh counter increments with $\overline{RASIN}$ going low to high, so that the DRAM rows are refreshed in succession by $\overline{RASIN}$ going high to low.

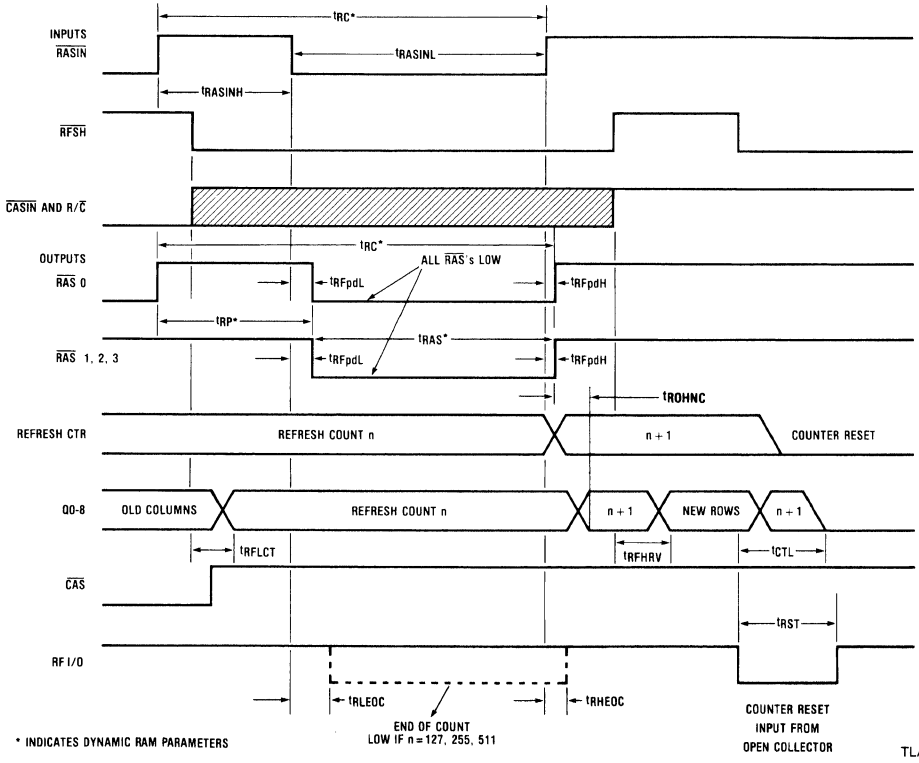


FIGURE 2. External Control Refresh Cycle (Mode 0)

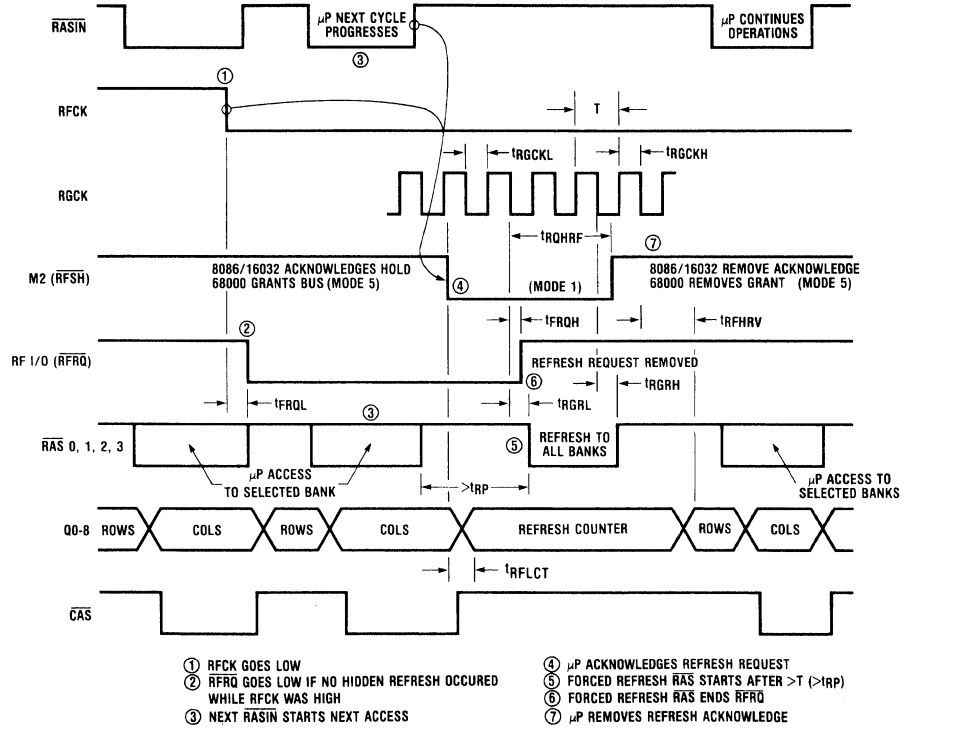
DP8409A Functional Mode Descriptions (Continued)

MODE 1—AUTOMATIC FORCED REFRESH

In Mode 1, the R/$\overline{C}$ (RFCK) pin becomes RFCK (refresh cycle clock), instead of R/$\overline{C}$, and $\overline{CAS}$ remains high. If RFCK is kept permanently high, then whenever M2 ($\overline{RFSH}$) goes low, an externally controlled refresh will occur and all $\overline{RAS}$ outputs will follow $\overline{RASIN}$, strobing the refresh counter contents to the DRAMs. The RF I/O pin will always output high, but when set low externally through an open-collector driver, the refresh counter resets as normal. This externally controlled method may be preferred when operating in the Automatic Access mode (Mode 5), where hidden or forced refreshing is undesirable, but refreshing is still necessary.

If RFCK is an input clock signal, one (and only one) refresh cycle must take place every RFCK cycle. Refer to *Figure 9*. If a hidden refresh does not occur while RFCK is high, in Mode 5, then RF I/O ($\overline{Refresh\ Request}$) goes low immediately after RFCK goes low, indicating to the system that a forced refresh is requested. The system must allow a forced refresh to take place while RFCK is low (refer to *Figure 3*). The $\overline{Refresh\ Request}$ signal on RF I/O may be connected to a $\overline{Hold}$ or $\overline{Bus\ Request}$ input to the system. The system acknowledges the $\overline{Hold}$ or $\overline{Bus\ Request}$ when ready, and outputs Hold Acknowledge or $\overline{Bus\ Request\ Acknowledge}$. If this is connected to the M2 ($\overline{RFSH}$) pin, a forced-refresh cycle will be initiated by the DP8409A, and $\overline{RAS}$ will be internally generated on all four $\overline{RAS}$ outputs, to strobe the refresh counter contents on the address outputs into all the DRAMs. An external $\overline{RAS}$ Generator Clock (RGCK) is required for this function. It is fed to the $\overline{CASIN}$ (RGCK) pin, and may be up to 10 MHz. Whenever M2 goes low (inducing a forced refresh), $\overline{RAS}$ remains high for one to two periods of RGCK, depending on when M2 goes low relative to the high-to-low triggering edge of RGCK; $\overline{RAS}$ then goes low for two periods, performing a refresh on all banks. In order to obtain the minimum delay from M2 going low to $\overline{RAS}$ going low, M2 should go low t_{RFSRG} before the next falling edge of RGCK. The $\overline{Refresh\ Request}$ on RF I/O is terminated as $\overline{RAS}$ begins, so that by the time the system has acknowledged the removal of the request and disabled its Acknowledge, (i.e., M2 goes high), Refresh $\overline{RAS}$ will have ended, and normal operations can begin again in the Automatic Access mode (Mode 5). If it is desired that Refresh $\overline{RAS}$ end in less than 2 periods of RGCK from the time $\overline{RAS}$ went low, then M2 may be high earlier than t_{RQHRF} after RGCK goes low and $\overline{RAS}$ will go high t_{RFRH} after M2, if $\overline{CS}$ is low. If $\overline{CS}$ is high, the $\overline{RAS}$ will go high after 25 ns after M2 goes high.

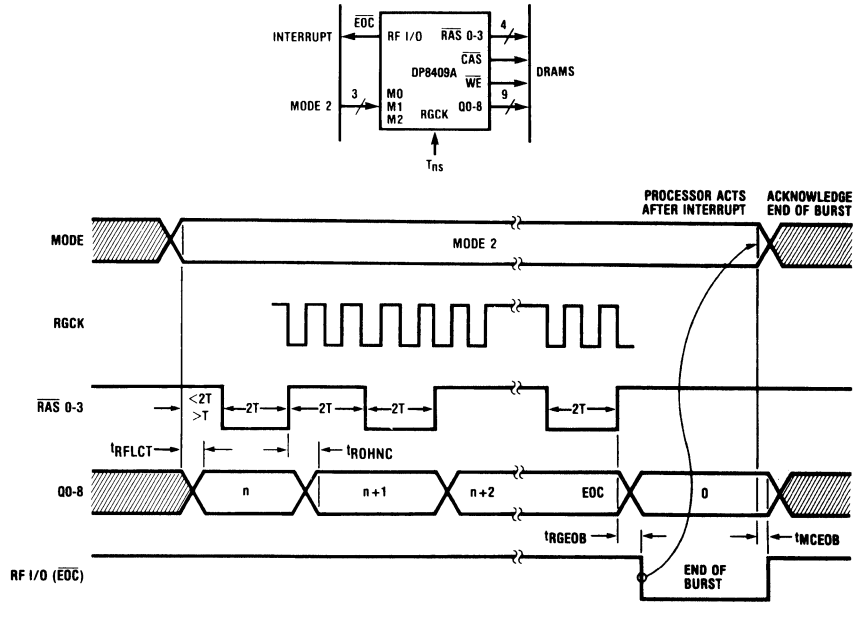
To allow the forced refresh, the system will have been inactive for about 4 periods of RGCK, which can be as fast as 400 ns every RFCK cycle. To guarantee a refresh of 128 rows every 2 ms, a period of up to 16 μs is required for RFCK. In other words, the system may be down for as little as 400 ns every 16 μs, or 2.5% of the time. Although this is not excessive, it may be preferable to perform a Hidden Refresh each RFCK cycle, which is allowed while still in the Auto-Access mode, (Mode 5).



TL/F/8409–10

FIGURE 3. DP8409A Performing a Forced Refresh (Mode 5 → 1 → 5) with Various Microprocessors

DP8409A Functional Mode Descriptions (Continued)



TL/F/8409–11

FIGURE 4. Auto-Burst Mode, Mode 2

MODE 2—AUTOMATIC BURST REFRESH

This mode is normally used before and/or after a DMA operation to ensure that all rows remain refreshed, provided the DMA transfer takes less than 2 ms (see *Figure 4*). When the DP8409A enters this mode, $\overline{\text{CASIN}}$ (RGCK) becomes the $\overline{\text{RAS}}$ Generator Clock (RGCK), and $\overline{\text{RASIN}}$ is disabled. $\overline{\text{CAS}}$ remains high, and RF I/O goes low when the refresh counter has reached the selected End-of-Count and the last $\overline{\text{RAS}}$ has ended. RF I/O then remains low until the Auto-Burst Refresh mode is terminated. RF I/O can therefore be used as an interrupt to indicate the End-of-Burst conditions.

The signal on all four $\overline{\text{RAS}}$ outputs is just a divide-by-four of RGCK; in other words, if RGCK has a 100 ns period, $\overline{\text{RAS}}$ is high and low for 200 ns each cycle. The refresh counter increments at the end of each $\overline{\text{RAS}}$, starting from the count it contained when the mode was entered. If this was zero, then for a RGCK with a 100 ns period with End-of-Count set to 127, RF I/O will go low after 128 × 0.4 μs, or 51.2 μs. During this time, the system may be performing operations that do not involve DRAM. If all rows need to be burst refreshed, the refresh counter may be cleared by setting RF I/O low externally before the burst begins.

Burst-mode refreshing is also useful when powering down systems for long periods of time, but with data retention still required while the DRAMs are in standby. To maintain valid refreshing, power can be applied to the DP8409A (set to Mode 2), causing it to perform a complete burst refresh. When end-of-burst occurs (after 26 μs), power can then be removed from the DP8409A for 2 ms, consuming an average power of 1.3% of normal operating power. No control signal glitches occur when switching power to the DP8409A.

MODE 3a—ALL-$\overline{\text{RAS}}$ AUTOMATIC WRITE

Mode 3a is useful at system initialization, when the memory is being cleared (i.e., with all-zeros in the data field and the corresponding check bits for error detection and correction). This requires writing the same data to each location of memory (every row of each column of each bank). All $\overline{\text{RAS}}$ outputs are activated, as in refresh, and so are $\overline{\text{CAS}}$ and $\overline{\text{WE}}$. To write to all four banks simultaneously, every row is strobed in each column, in sequence, until data has been written to all locations.

To select this mode, B1 and B0 must have previously been set to 00, 01, or 10 in Mode 7, depending on the DRAM size. For example, for 16k DRAMs, B1 and B0 are 00. For 64k DRAMs, B1 and B0 are 01, so that for the configuration of *Figure 1b*, the 8 refresh counter bits are strobed by $\overline{\text{RAS}}$ into the 7 row addresses and the ninth column address. After this Automatic-Write process, B1 and B0 must be set again in Mode 7 to 00 to set End-of-Count to 127. For the configuration of *Figure 1c*, B1 and B0 set to 01 will work for Automatic-Write and End-of-Count equals 255.

In this mode, R/$\overline{\text{C}}$ is disabled, $\overline{\text{WE}}$ is permanently enabled low, and $\overline{\text{CASIN}}$ (RGCK) becomes RGCK. RF I/O goes low whenever the refresh counter is 127, 255, or 511 (as set by End-of-Count in Mode 7), and the $\overline{\text{RAS}}$ outputs are active.

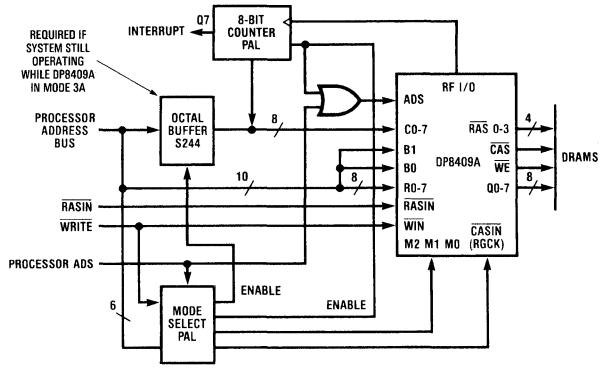
Referring to *Figure 5a*, an external 8-bit counter (for 64k DRAMs) with TRI-STATE outputs is required and must be connected to the column address inputs. It is enabled only during this mode, and is clocked from RF I/O. The DP8409A refresh counter is used to address the rows, and the column address is supplied by the external counter. Every row for each column address is written to in all four banks. At the End-of-Count RF I/O goes low, which clocks the external counter.

Therefore, for each column address, the refresh counter first outputs row-0 to the address bus and all four $\overline{\text{RAS}}$ outputs strobe this row address into the DRAMs (see *Figure 5b*). A minimum of 30 ns after $\overline{\text{RAS}}$ goes low (t_{RAH} = 30 ns), the refresh counter is disabled and the column ad-

DP8409A Functional Mode Descriptions (Continued)

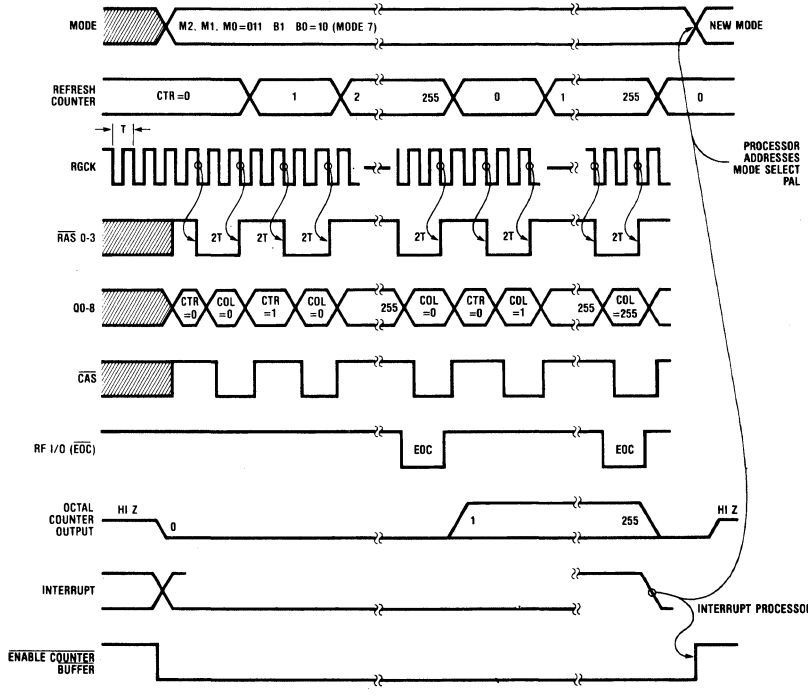
dress input latch is enabled onto the address bus. About 14 ns after the column address is valid, $\overline{CAS}$ goes low, (t_{ASC} = +14 ns), strobing the column address into the DRAMs. When $\overline{RAS}$ and $\overline{CAS}$ go high the refresh counter increments to the next row and the cycle repeats. Since $\overline{WE}$ is kept low in this mode, the data at DI (input data) of the DRAMs is written into each row of the latched column. During each cycle $\overline{RAS}$ is high for two periods of RGCK and low for two periods, giving a total write-cycle time of 400 ns minimum, which is adequate for most 16k and 64k DRAMs. On the last row of a column, RF I/O increments the external counter to the next column address.

At the end of the last column address, an interrupt is generated from the external counter to let the system know that initialization has been completed. During the entire initialization time, the system can be performing other initialization functions. This approach to memory initialization is both automatic and fast. For instance, if four banks of 64k DRAMs are used, and RGCK is 100 ns, a write cycle to the same location in all four banks takes 400 ns, so the total time taken in initializing the 64k DRAMs is 65k × 400 ns or 26 ms. When the system receives the interrupt, the external counter must be permanently disabled. ADS and $\overline{CS}$ are interfaced by the system, and the DP8409A mode is changed. The interrupt must then be disabled.



TL/F/8409–12

FIGURE 5a. DP8409A Extra Circuitry Required for All-$\overline{RAS}$ Auto Write Mode, Mode 3a



TL/F/8409–13

FIGURE 5b. DP8409A All-$\overline{RAS}$ Auto Write Mode, Mode 3a, Timing Waveform

DP8409A Functional Mode Descriptions (Continued)

MODE 3b—EXTERNALLY CONTROLLED ALL-$\overline{RAS}$ WRITE

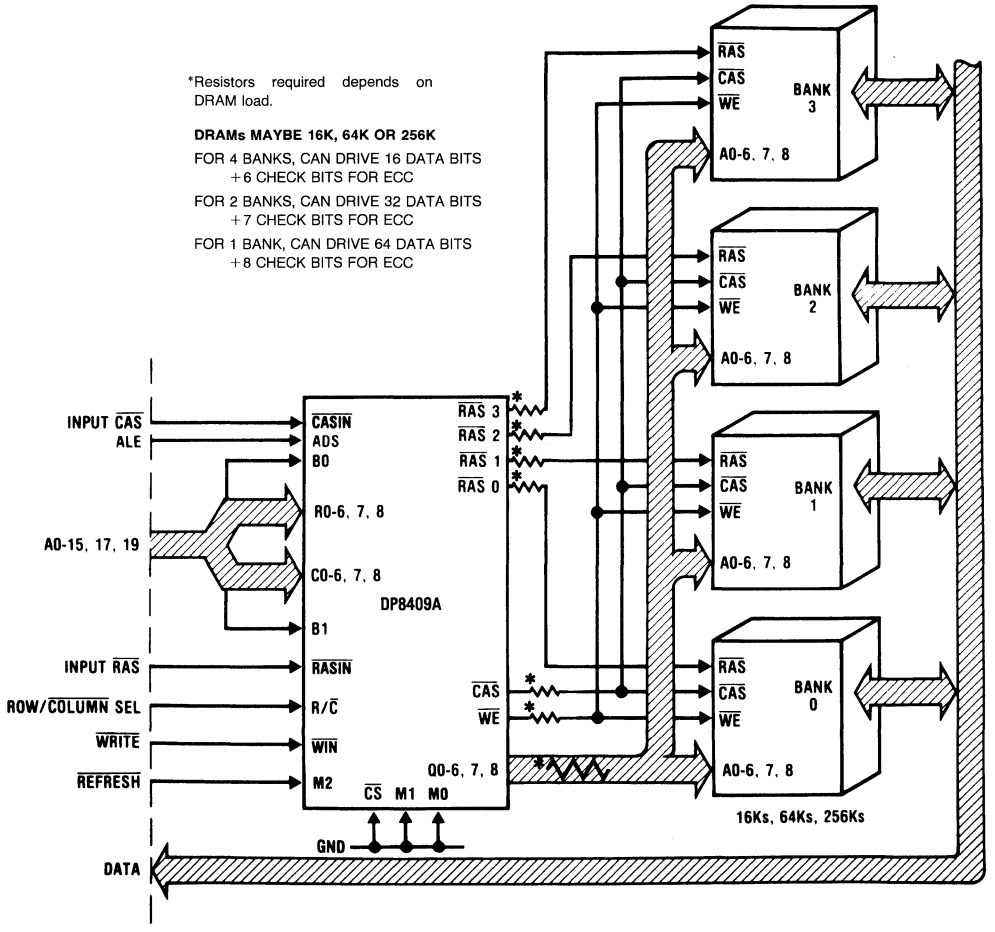
To select this mode, B1 and B0 must first have been set to 11 in Mode 7. This mode is useful at system initialization, but under processor control. The memory address is provided by the processor, which also performs the incrementing. All four $\overline{RAS}$ outputs follow $\overline{RASIN}$ (supplied by the processor), strobing the row address into the DRAMs. R/$\overline{C}$ can now go low, while $\overline{CASIN}$ may be used to control $\overline{CAS}$ (as in the Externally Controlled Access mode), so that $\overline{CAS}$ strobes the column address contents into the DRAMs. At this time $\overline{WE}$ should be low, causing the data to be written into all four banks of DRAMs. At the end of the write cycle, the input address is incremented and latched by the DP8409A for the next write cycle. This method is slower than Mode 3a since the processor must perform the incrementing and accessing. Thus the processor is occupied during RAM initialization, and is not free for other initialization operations. However, initialization sequence timing is under system control, which may provide some system advantage.

MODE 4—EXTERNALLY CONTROLLED ACCESS

This mode facilitates externally controlling all access-timing parameters associated with the DRAMs. The application of modes 0 and 4 are shown in *Figure 6*.

Output Address Selection

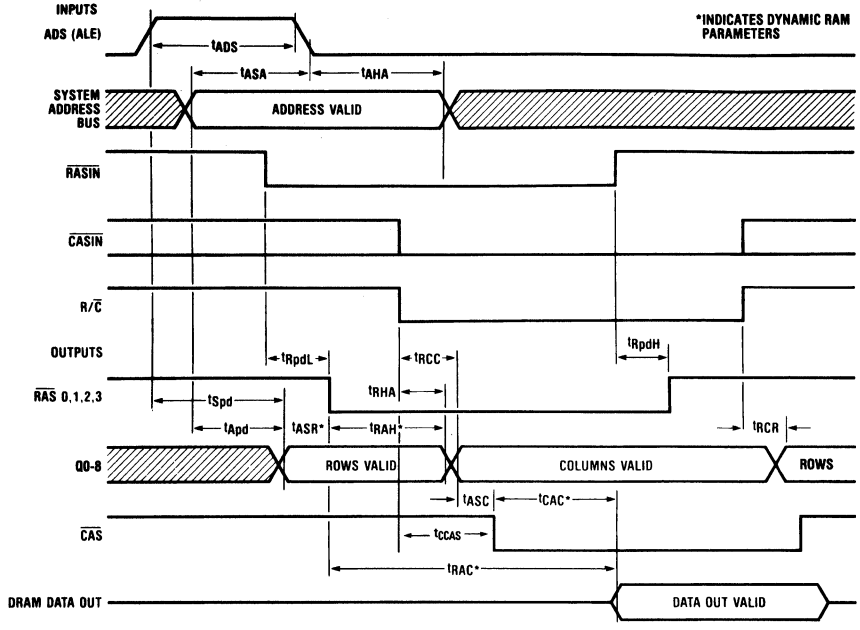
Refer to *Figure 7a*. With M2 ($\overline{RFSH}$) and R/$\overline{C}$ high, the row address latch contents are transferred to the multiplexed address bus output Q0–Q8, provided $\overline{CS}$ is set low. The column address latch contents are output after R/$\overline{C}$ goes low. $\overline{RASIN}$ can go low after the row addresses have been set up on Q0–Q8. This selects one of the $\overline{RAS}$ outputs, strobing the row address on the Q outputs into the desired bank of memory. After the row-address hold-time of the DRAMs, R/$\overline{C}$ can go low so that about 40 ns later column addresses appear on the Q outputs.



TL/F/8409–14

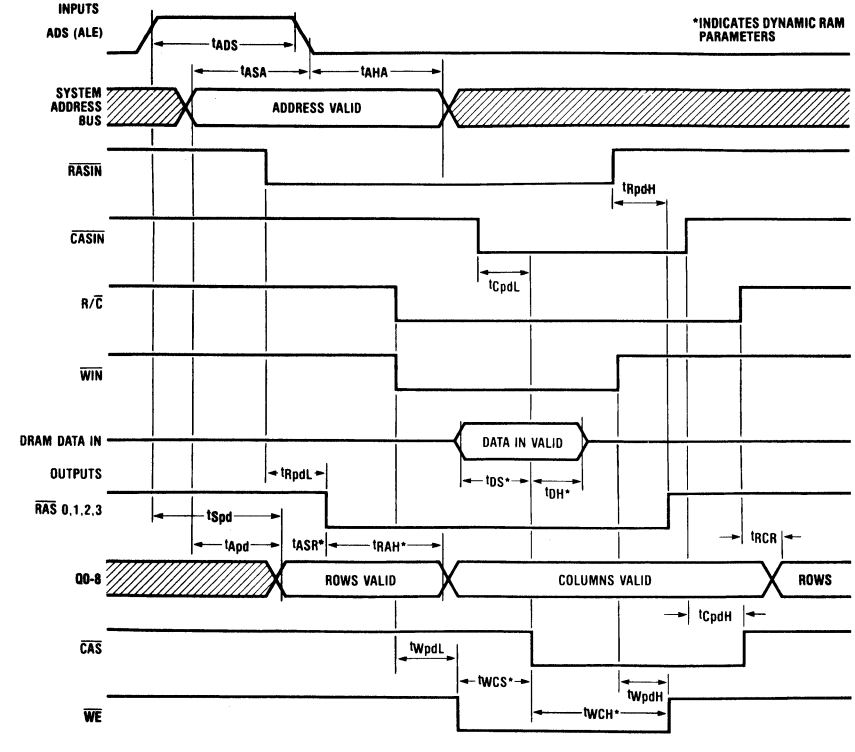
FIGURE 6. Typical Application of DP8409A Using External Control Access and Refresh in Modes 0 and 4

DP8409A Functional Mode Descriptions (Continued)



TL/F/8409–15

FIGURE 7a. Read Cycle Timing (Mode 4)



TL/F/8409–16

FIGURE 7b. Write Cycle Timing (Mode 4)

DP8409A Functional Mode Descriptions (Continued)

Automatic CAS Generation

In a normal memory access cycle $\overline{CAS}$ can be derived from inputs $\overline{CASIN}$ or R/$\overline{C}$. If $\overline{CASIN}$ is high, then R/C going low switches the address output drivers from rows to columns. $\overline{CASIN}$ then going low causes $\overline{CAS}$ to go low approximately 40 ns later, allowing $\overline{CAS}$ to occur at a predictable time (see *Figure 7b*). If $\overline{CASIN}$ is low when R/$\overline{C}$ goes low, $\overline{CAS}$ will be automatically generated, following the row to column transition by about 20 ns (see *Figure 7a*). Most DRAMs have a column address set-up time before $\overline{CAS}$ (t_{ASC}) of 0 ns or −10 ns. In other words, a t_{ASC} greater than 0 ns is safe.

Fast Memory Access

AC parameters t_{DIF1}, t_{DIF2} may be used to determine the minimum delays required between $\overline{RASIN}$, R/$\overline{C}$, and $\overline{CASIN}$ (see Application Brief 9; "Fastest DRAM Access Mode").

MODE 5—AUTOMATIC ACCESS WITH HIDDEN REFRESH

The Auto Access with Hidden Refresh mode has two advantages over the Externally Controlled Access mode, due to the fact that all outputs except $\overline{WE}$ are initiated from $\overline{RASIN}$. First, inputs R/$\overline{C}$ and $\overline{CASIN}$ are unnecessary and can be used for other functions (see Refreshing, below). Secondly, because the output control signals are derived internally from one input signal ($\overline{RASIN}$), timing-skew problems are reduced, thereby reducing memory access time substantially or allowing use of slower DRAMs. The automatic access features of Mode 5 (and Mode 6) of the DP8409A make DRAM accessing appear essentially "static".

Automatic Access Control

The major disadvantage of DRAMs compared to static RAMs is the complex timing involved. First, a $\overline{RAS}$ must occur with the row address previously set up on the multiplexed address bus. After the row address has been held for t_{RAH}, (the Row-Address hold-time of the DRAM), the column address is set up and then $\overline{CAS}$ occurs. This is all performed automatically by the DP8409A in this mode.

Provided the input address is valid as ADS goes low, $\overline{RASIN}$ can go low any time after ADS. This is because the selected $\overline{RAS}$ occurs typically 27 ns later, by which time the row address is already valid on the address output of the DP8409A. The Address Set-Up time (t_{ASR}), is 0 ns on most DRAMs. The DP8409A in this mode (with ADS and $\overline{RASIN}$ edges simultaneously applied) produces a minimum t_{ASR} of 0 ns. This is true provided the input address was valid t_{ASA} before ADS went low (see *Figure 8a*).

Next, the row address is disabled after t_{RAH} (30 ns minimum); in most DRAMs, t_{RAH} minimum is less than 30 ns. The column address is then set up and t_{ASC} later, $\overline{CAS}$ occurs. The only other control input required is $\overline{WIN}$. When a write cycle is required, $\overline{WIN}$ must go low at least 30 ns before $\overline{CAS}$ is output low.

This gives a total typical delay from: input address valid to $\overline{RASIN}$ (15 ns); to $\overline{RAS}$ (27 ns); to rows held (50 ns); to columns valid (25 ns); to $\overline{CAS}$ (23 ns) = 140 ns (that is, 125 ns from $\overline{RASIN}$). All of these typcial figures are for heavy capacitive loading, of approximately 88 DRAMs.

This mode is therefore extremely fast. The external timing is greatly simplified for the memory system designer: the only system signal required is $\overline{RASIN}$.

Refreshing

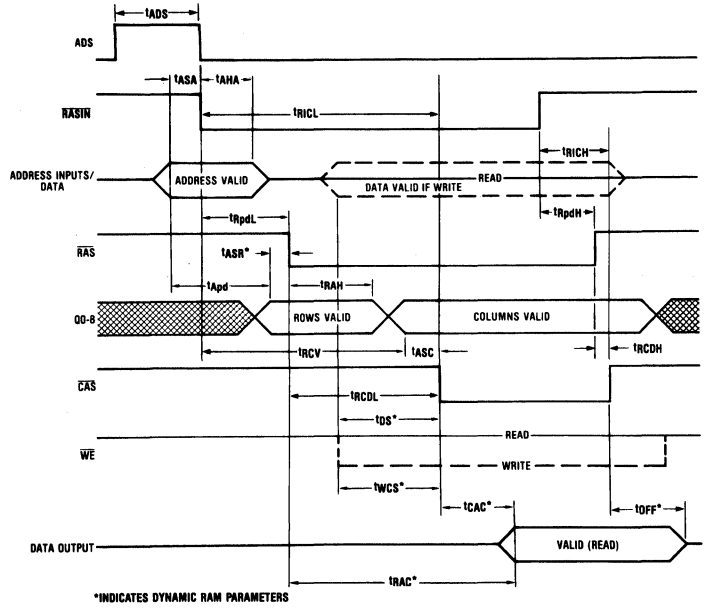
Because R/$\overline{C}$ and $\overline{CASIN}$ are not used in this mode, R/$\overline{C}$ becomes RFCK (refresh clock) and $\overline{CASIN}$ becomes RGCK ($\overline{RAS}$ generator clock). With these two signals it is possible to peform refreshing without extra ICs, and without holding up the processor.

One refresh cycle must occur during each refresh clock period and then the refresh address must be incremented to the next refresh cycle. As long as 128 rows are refreshed every 2 ms (one row every 16 μs), all 16k and 64k DRAMs will be correctly refreshed. The cycle time of RFCK must, therefore, be less than 16 μs. RFCK going high sets an internal refresh-request flip-flop. First the DP8409A will attempt to perform a hidden refresh so that the system throughput will not be affected. If, during the time RFCK is high, $\overline{CS}$ on the DP8409A goes high and $\overline{RASIN}$ occurs, a hidden refresh will occur. In this case, $\overline{RASIN}$ should be considered a common read/write strobe. In other words, if the processor is accessing elsewhere (other than the DRAMs) while RFCK is high, the DP8409A will perform a refresh. The refresh counter is enabled to the address outputs whenever $\overline{CS}$ goes high with RFCK high, and all $\overline{RAS}$ outputs follow $\overline{RASIN}$. If a hidden refresh is taking place as RFCK goes low, the refresh continues. At the start of the hidden refresh, the refresh-request flip-flop is reset so no further refresh can occur until the next RFCK period starts with the positive-going edge of RFCK. Refer to *Figure 9*.

To determine the probability of a Hidden Refresh occurring, assume each system cycle takes 400 ns and RFCK is high for 8 μs, then the system has 20 chances to not select the DP8409A. If during this time a hidden refresh did not occur, then the DP8409A forces a refresh while RFCK is low, but the system chooses when the refresh takes place. After RFCK goes low, (and the internal-request flip-flop has not been reset), RF I/O goes low indicating that a refresh is requested to the system. Only when the system acknowledges this request by setting M2 ($\overline{RFSH}$) low does the DP8409A initiate a forced refresh (which is performed automatically). Refer to Mode 1, and *Figure 3*. The internal refresh request flip-flop is then reset.

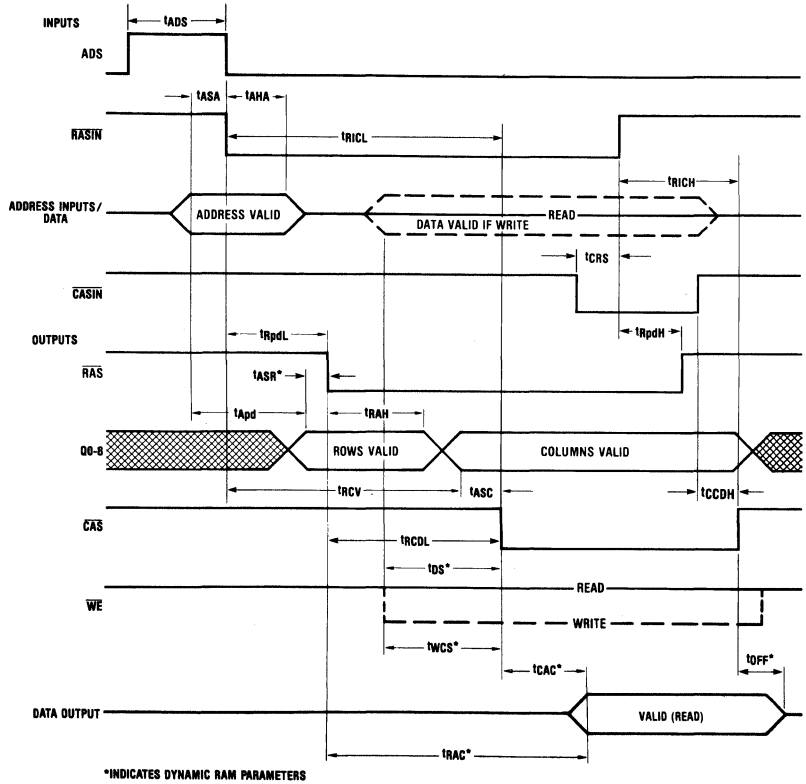
Figure 9 illustrates the refresh alternatives in Mode 5. If a hidden refresh has occurred and $\overline{CS}$ again goes high before RFCK goes low, the chip is deselected. All the control signals go high-impedance high (logic "1") and the address outputs go TRI-STATE until $\overline{CS}$ again goes low. This mode (combined with Mode 1) allows very fast access, and automatic refreshing (possibly not even slowing down the system), with no extra ICs. Careful system design can, and should, provide a higher probability of hidden refresh occurring. The duty cycle of RFCK need not be 50-percent; in fact, the low-time should be designed to be a minimum. This is determined by the worst-case time (required by the system) to respond to the DP8409A's forced-refresh request.

DP8409A Functional Mode Descriptions (Continued)



TL/F/8409–17

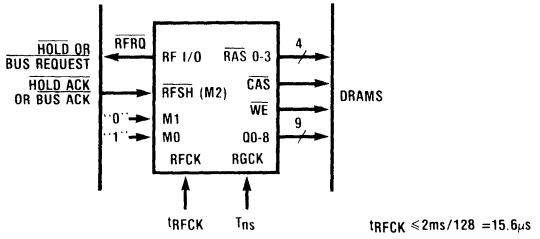
FIGURE 8a. Modes 5, 6 Timing ($\overline{\text{CASIN}}$ High in Mode 6)

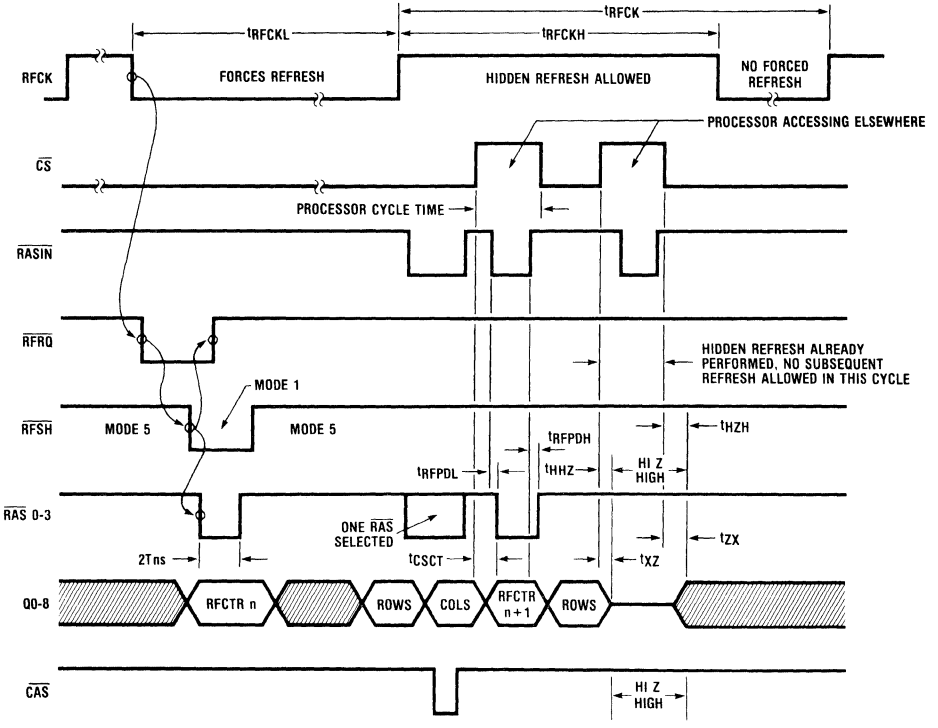


TL/F/8409–18

FIGURE 8b. Mode 6 Timing, Extended $\overline{\text{CAS}}$

DP8409A Functional Mode Descriptions (Continued)





TL/F/8409–19

FIGURE 9. Hidden Refreshing (Mode 5) and Forced Refreshing (Mode 1) Timing

7

DP8409A Functional Mode Descriptions (Continued)

TABLE II. Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{RAS}_n$
B1	B0	
0	0	$\overline{RAS}_0$
0	1	$\overline{RAS}_1$
1	0	$\overline{RAS}_2$
1	1	$\overline{RAS}_3$

Note that $\overline{RASIN}$ going low earlier than t_{CSRL} after $\overline{CS}$ goes low may result in the DP8409A interpreting the $\overline{RASIN}$ as a hidden refresh $\overline{RASIN}$ if no hidden refresh has occurred in the current RFCK cycle. In this case, all $\overline{RAS}$ outputs would go low for a short time. Thus, it is suggested that when using Mode 5, $\overline{RASIN}$ should be held high until t_{CSRL} after $\overline{CS}$ goes low if a refresh is not intended. Similarly, $\overline{CS}$ should be held low for a minimum of t_{CSRL} after $\overline{RASIN}$ returns high when ending the access in Mode 5.

MODE 6—FAST AUTOMATIC ACCESS

The Fast Access mode is similar to Mode 5, but has a faster t_{RAH} of 20 ns, minimum. It therefore can only be used with fast 16k or 64k DRAMs (which have a t_{RAH} of 10 ns to 15 ns) in applications requiring fast access times; $\overline{RASIN}$ to $\overline{CAS}$ is typically 105 ns.

In this mode, the R/$\overline{C}$ (RFCK) pin is not used, but $\overline{CASIN}$ (RGCK) is used as $\overline{CASIN}$ to allow an extended $\overline{CAS}$ after $\overline{RAS}$ has already terminated. Refer to *Figure 8b*. This is desirable with fast cycle-times where $\overline{RAS}$ has to be terminated as soon as possible before the next $\overline{RAS}$ begins (to meet the precharge time, or t_{RP}, requirements of the DRAM). $\overline{CAS}$ may then be held low by $\overline{CASIN}$ to extend the data output valid time from the DRAM to allow the system to read the data. $\overline{CASIN}$ subsequently going high ends $\overline{CAS}$. If this extended $\overline{CAS}$ is not required, $\overline{CASIN}$ should be set high in Mode 6.

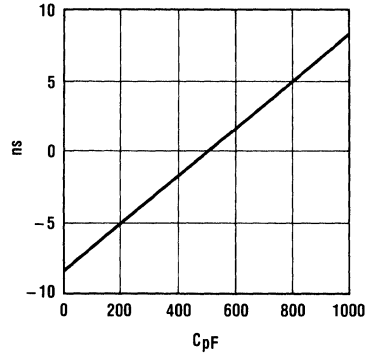
There is no internal refresh-request flip-flop in this mode, so any refreshing required must be done by entering Mode 0 or Mode 2.

MODE 7—SET END-OF-COUNT

The End-of-Count can be externally selected in Mode 7, using ADS to strobe in the respective value of B1 and B0 (see Table III). With B1 and B0 the same $\overline{EOC}$ is 127; with B1 = 0 and B0 = 1, $\overline{EOC}$ is 255; and with B1 = 1 and B0 = 0, $\overline{EOC}$ is 511. This selected value of $\overline{EOC}$ will be used until the next Mode 7 selection. At power-up the $\overline{EOC}$ is automatically set to 127 (B1 and B0 set to 11).

TABLE III. Mode 7

Bank Select (Strobed by ADS)		End of Count Selected
B1	B0	
0	0	127
0	1	255
1	0	511
1	1	127



TL/F/8409–20

FIGURE 10. Change in Propagation Delay vs. Loading Capacitance Relative to a 500 pF Load

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage, V_{CC}	7.0V
Storage Temperature Range	−65°C to +150°C
Input Voltage	5.5V
Output Current	150 mA
Lead Temperature (Soldering, 10 seconds)	300°C

Maximum Power Dissipation* at 25°C	
Cavity Package	3542 mW
Molded Package	2833 mW

*Derate cavity package 23.6 mW/°C above 25°C; derate molded package 22.7 mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
V_{CC} Supply Voltage	4.75	5.25	V
T_A Ambient Temperature	0	+70	°C

Electrical Characteristics V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 6)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_C	Input Clamp Voltage	V_{CC} = Min, I_C = −12 mA		−0.8	−1.2	V
I_{IH1}	Input High Current for ADS, R/$\overline{C}$ Only	V_{IN} = 2.5V		2.0	100	μA
I_{IH2}	Input High Current for All Other Inputs*	V_{IN} = 2.5V		1.0	50	μA
I_I RSI	Output Load Current for RF I/O	V_{IN} = 0.5V, Output High		−1.5	−2.5	mA
I_I CTL	Output Load Current for $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$	V_{IN} = 0.5V, Chip Deselect		−1.5	−2.5	mA
I_{IL1}	Input Low Current for ADS, R/$\overline{C}$ Only	V_{IN} = 0.5V		−0.1	−1.0	mA
I_{IL2}	Input Low Current for All Other Inputs*	V_{IN} = 0.5V		−0.05	−0.5	mA
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_{OL1}	Output Low Voltage*	I_{OL} = 20 mA		0.3	0.5	V
V_{OL2}	Output Low Voltage for RF I/O	I_{OL} = 10 mA		0.3	0.5	V
V_{OH1}	Output High Voltage*	I_{OH} = −1 mA	2.4	3.5		V
V_{OH2}	Output High Voltage for RF I/O	I_{OH} = − 100 μA	2.4	3.5		V
I_{1D}	Output High Drive Current*	V_{OUT} = 0.8V (Note 3)		−200		mA
I_{0D}	Output Low Drive Current*	V_{OUT} = 2.7V (Note 3)		200		mA
I_{OZ}	TRI-STATE Output Current (Address Outputs)	0.4V ≤ V_{OUT} ≤ 2.7V, $\overline{CS}$ = 2.0V, Mode 4	−50	1.0	50	μA
I_{CC}	Supply Current	V_{CC} = Max		250	325	mA

*Except RF I/O Output.

Switching Characteristics: DP8409A/DP8409A-3

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	8409			8409A-3			Units
			Min	Typ	Max	Min	Typ	Max	
ACCESS									
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*	95	125	160	95	125	185	ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figures 8a, 8b*	80	105	140	80	105	160	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*	40	48	60	40	48	70	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figures 8a, 8b*	50	63	80	50	63	95	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*		98	125		98	145	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figures 8a, 8b*		78	105		78	120	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*		27	40		27	40	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figure 8a*		40	65		40	65	ns

Switching Characteristics: DP8409A/DP8409A-3 (Continued)

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	DP8409A			DP8409A-3			Units
			Min	Typ	Max	Min	Typ	Max	
ACCESS (Continued)									
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figure 8b*	40	54	70	40	54	80	ns
t_{RAH}	Row Address Hold Time (Mode 5)	*Figure 8a*	30			30			ns
t_{RAH}	Row Address Hold Time (Mode 6)	*Figures 8a, 8b*	20			20			ns
t_{ASC}	Column Address Setup Time (Mode 5)	*Figure 8a*	8			8			ns
t_{ASC}	Column Address Setup Time (Mode 6)	*Figures 8a, 8b*	6			6			ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 5)	*Figure 8a*		90	120		90	140	ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 6)	*Figures 8a, 8b*		75	105		75	120	ns
t_{RPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Delay	*Figures 7a, 7b, 8a, 8b*	20	27	35	20	27	40	ns
t_{RPDH}	$\overline{RASIN}$ to $\overline{RAS}$ Delay	*Figures 7a, 7b, 8a, 8b*	15	23	32	15	23	37	ns
t_{APDL}	Address Input to Output Low Delay	*Figures 7a, 7b, 8a, 8b*		25	40		25	46	ns
t_{APDH}	Address Input to Output High Delay	*Figures 7a, 7b, 8a, 8b*		25	40		25	46	ns
t_{SPDL}	Address Strobe to Address Output Low	*Figures 7a, 7b*		40	60		40	70	ns
t_{SPDH}	Address Strobe to Address Output High	*Figures 7a, 7b*		40	60		40	70	ns
t_{ASA}	Address Set-Up Time to ADS	*Figures 7a, 7b, 8a, 8b*	15			15			ns
t_{AHA}	Address Hold Time from ADS	*Figures 7a, 7b, 8a, 8b*	15			15			ns
t_{ADS}	Address Strobe Pulse Width	*Figures 7a, 7b, 8a, 8b*	30			30			ns
t_{WPDL}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	*Figure 7b*	15	25	30	15	25	35	ns
t_{WPDH}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	*Figure 7b*	15	30	60	15	30	70	ns
t_{CRS}	$\overline{CASIN}$ Set-Up Time to $\overline{RASIN}$ High (Mode 6)	*Figure 8b*	35			35			ns
t_{CPDL}	$\overline{CASIN}$ to $\overline{CAS}$ Delay (R/$\overline{C}$ Low in Mode 4)	*Figure 7b*	32	41	68	32	41	77	ns
t_{CPDH}	$\overline{CASIN}$ to $\overline{CAS}$ Delay (R/$\overline{C}$ Low in Mode 4)	*Figure 7b*	25	39	50	25	39	60	ns
t_{RCC}	Column Select to Column Address Valid	*Figure 7a*		40	58		40	67	ns
t_{RCR}	Row Select to Row Address Valid	*Figures 7a, 7b*		40	58		40	67	ns
t_{RHA}	Row Address Held from Column Select	*Figure 7a*	10			10			ns
t_{CCAS}	R/$\overline{C}$ Low to $\overline{CAS}$ Low (Mode 4 Auto $\overline{CAS}$)	*Figure 7a*		65	90				ns
t_{DIF1}	Maximum (t_{RPDL} − t_{RHA})	See Mode 4 Descrip.			13			18	ns
t_{DIF2}	Maximum (t_{RCC} − t_{CPDL})	See Mode 4 Descrip.			13			18	ns
REFRESH									
t_{RC}	Refresh Cycle Period	*Figure 2*	100			100			ns
$t_{RASINL, H}$	Pulse Width of $\overline{RASIN}$ during Refresh	*Figure 2*	50			50			ns
t_{RFPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Delay during Refresh	*Figures 2, 9*	35	50	70	35	50	80	ns
t_{RFPDH}	$\overline{RASIN}$ to $\overline{RAS}$ Delay during Refresh	*Figures 2, 9*	30	40	55	30	40	65	ns
t_{RFLCT}	$\overline{RFSH}$ Low to Counter Address Valid	$\overline{CS}$ = X, *Figures 2, 3, 4*		47	60		47	70	ns

Switching Characteristics: DP8409A/DP8409A-3 (Continued)

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	DP8409A			DP8409A-3			Units
			Min	Typ	Max	Min	Typ	Max	
REFRESH (Continued)									
t_{RFHRV}	$\overline{RFSH}$ High to Row Address Valid	*Figures 2, 3*		45	60		45	70	ns
t_{ROHNC}	$\overline{RAS}$ High to New Count Valid	*Figures 2, 4*		30	55		30	55	ns
t_{RLEOC}	$\overline{RASIN}$ Low to End-of-Count Low	C_L = 50 pF, *Figure 2*			80			80	ns
t_{RHEOC}	$\overline{RASIN}$ High to End-of-Count High	C_L = 50 pF, *Figure 2*			80			80	ns
t_{RGEOB}	RGCK Low to End-of-Burst Low	C_L = 50 pF, *Figure 4*			95			95	ns
t_{MCEOB}	Mode Change to End-of-Burst High	C_L = 50 pF, *Figure 4*			75			75	ns
t_{RST}	Counter Reset Pulse Width	*Figure 2*	70			70			ns
t_{CTL}	RF I/O Low to Counter Outputs All Low	*Figure 2*			100			100	ns
$t_{RFCKL, H}$	Minimum Pulse Width of RFCK	*Figure 9*	100			100			ns
T	Period of $\overline{RAS}$ Generator Clock	*Figure 3*	100			100			ns
t_{RGCKL}	Minimum Pulse Width Low of RGCK	*Figure 3*	35			40			ns
t_{RGCKH}	Minimum Pulse Width High of RGCK	*Figure 3*	35			40			ns
t_{FRQL}	RFCK Low to Forced $\overline{RFRQ}$ Low	C_L = 50 pF, *Figure 3*		20	30		20	30	ns
t_{FRQH}	RGCK Low to Forced $\overline{RFRQ}$ High	C_L = 50 pF, *Figure 3*		50	75		50	75	ns
t_{RGRL}	RGCK Low to $\overline{RAS}$ Low	*Figure 3*	50	65	95	50	65	95	ns
t_{RGRH}	RGCK Low to $\overline{RAS}$ High	*Figure 3*	40	60	85	40	60	85	ns
t_{RQHRF}	$\overline{RFSH}$ Hold Time from $\overline{RFSH\ RQST}$ (RF I/O)	*Figure 3*	2T			2T			ns
t_{RFRH}	$\overline{RFSH}$ High to $\overline{RAS}$ High (ending forced RFSH)	See Mode 1 Descrip.	55	80	110	55	80	125	ns
t_{RFSRG}	$\overline{RFSH}$ Low Set-Up to RGCK Low (Mode 1)	See Mode 1 Descrip.	35			40			ns
t_{CSCT}	$\overline{CS}$ High to RFSH Counter Valid	*Figure 9*		55	70		55	75	ns
t_{CSRL}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low	See Mode 5 Descrip.	30			30			ns
TRI-STATE									
t_{ZH}	$\overline{CS}$ Low to Address Output High from Hi-Z	*Figures 9, 12,* R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{HZ}	$\overline{CS}$ High to Address Output Hi-Z from High	C_L = 15 pF, *Figures 9, 12,* R2 = 1k, S1 Open		20	40		20	40	ns
t_{ZL}	$\overline{CS}$ Low to Address Output Low from Hi-Z	*Figures 9, 12,* R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{LZ}	$\overline{CS}$ High to Address Output Hi-Z from Low	C_L = 15 pF, *Figures 9, 12,* R1 = 1k, S2 Open		25	50		25	50	ns
t_{HZH}	$\overline{CS}$ Low to Control Output High from Hi-Z High	*Figures 9, 12,* R2 = 750Ω, S1 Open		50	80		50	80	ns
t_{HHZ}	$\overline{CS}$ High to Control Output Hi-Z High from High	C_L = 15 pF, *Figures 9, 12,* R2 = 750Ω, S1 Open		40	75		40	75	ns

Switching Characteristics: DP8409A/DP8409A-3 (Continued)

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	DP8409A			DP8409A-3			Units
			Min	Typ	Max	Min	Typ	Max	
TRI-STATE (Continued)									
t_{HZL}	$\overline{CS}$ Low to Control Output Low from Hi-Z High	*Figure 12*, S1, S2 Open		45	75		45	75	ns
t_{LHZ}	$\overline{CS}$ High to Control Output Hi-Z High from Low	C_L = 15 pF, *Figure 12*, R2 = 750Ω, S1 Open		50	80		50	80	ns

Switching Characteristics: DP8409A-2

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	8409A-2			Units
			Min	Typ	Max	
ACCESS						
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*	75	100	130	ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figures 8a, 8b*	65	90	115	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*	40	48	60	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figures 8a, 8b*	50	63	80	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*		75	100	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figures 8a, 8b*		65	85	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	*Figure 8a*		27	40	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figure 8a*		40	65	ns
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	*Figure 8b*	40	54	70	ns
t_{RAH}	Row Address Hold Time (Mode 5) (Note 7)	*Figure 8a*	20			ns
t_{RAH}	Row Address Hold Time (Mode 6) (Note 7)	*Figures 8a, 8b*	12			ns
t_{ASC}	Column Address Set-Up Time (Mode 5)	*Figure 8a*	3			ns
t_{ASC}	Column Address Set-Up Time (Mode 6)	*Figures 8a, 8b*	3			ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 5)	*Figure 8a*		80	105	ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 6)	*Figures 8a, 8b*		70	90	ns
t_{RPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Delay	*Figures 7a, 7b, 8a, 8b*	20	27	35	ns
t_{RPDH}	$\overline{RASIN}$ to $\overline{RAS}$ Delay	*Figures 7a, 7b, 8a, 8b*	15	23	32	ns
t_{APDL}	Address Input to Output Low Delay	*Figures 7a, 7b, 8a, 8b*		25	40	ns
t_{APDH}	Address Input to Output High Delay	*Figures 7a, 7b, 8a, 8b*		25	40	ns
t_{SPDL}	Address Strobe to Address Output Low	*Figures 7a, 7b*		40	60	ns
t_{SPDH}	Address Strobe to Address Output High	*Figures 7a, 7b*		40	60	ns
t_{ASA}	Address Set-Up Time to ADS	*Figures 7a, 7b, 8a, 8b*	15			ns
t_{AHA}	Address Hold Time from ADS	*Figures 7a, 7b, 8a, 8b*	15			ns
t_{ADS}	Address Strobe Pulse Width	*Figures 7a, 7b, 8a, 8b*	30			ns

Switching Characteristics: DP8409A-2 (Continued)

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	8409A-2			Units
			Min	Typ	Max	
ACCESS (Continued)						
t_{WPDL}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	*Figure 7b*	15	25	30	ns
t_{WPDH}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	*Figure 7b*	15	30	60	ns
t_{CRS}	$\overline{CASIN}$ Set-Up Time to $\overline{RASIN}$ High (Mode 6)	*Figure 8b*	35			ns
t_{CPDL}	$\overline{CASIN}$ to $\overline{CAS}$ Delay (R/$\overline{C}$ Low in Mode 4)	*Figure 7b*	32	41	58	ns
t_{CPDH}	$\overline{CASIN}$ to $\overline{CAS}$ Delay (R/$\overline{C}$ Low in Mode 4)	*Figure 7b*	25	39	50	ns
t_{RCC}	Column Select to Column Address Valid	*Figure 7a*		40	58	ns
t_{RCR}	Row Select to Row Address Valid	*Figures 7a, 7b*		40	58	ns
t_{RHA}	Row Address Held from Column Select	*Figure 7a*	10			ns
t_{CCAS}	R/$\overline{C}$ Low to $\overline{CAS}$ Low (Mode 4 Auto $\overline{CAS}$)	*Figure 7a*		55	75	ns
t_{DIF1}	Maximum (t_{RPDL} − t_{RHA})	See Mode 4 Descript.			13	ns
t_{DIF2}	Maximum (t_{RCC} − t_{CPDL})	See Mode 4 Descript.			13	ns
REFRESH						
t_{RC}	Refresh Cycle Period	*Figure 2*	100			ns
$t_{RASINL, H}$	Pulse Width of $\overline{RASIN}$ during Refresh	*Figure 2*	50			ns
t_{RFPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Delay during Refresh	*Figures 2, 9*	35	50	70	ns
t_{RFPDH}	$\overline{RASIN}$ to $\overline{RAS}$ Delay during Refresh	*Figures 2, 9*	30	40	55	ns
t_{RFLCT}	$\overline{RFSH}$ Low to Counter Address Valid	$\overline{CS}$ = X, *Figures 2, 3, 4*		47	60	ns
t_{RFHRV}	$\overline{RFSH}$ High to Row Address Valid	*Figures 2, 3*		45	60	ns
t_{ROHNC}	$\overline{RAS}$ High to New Count Valid	*Figures 2, 4*		30	55	ns
t_{RLEOC}	$\overline{RASIN}$ Low to End-of-Count Low	C_L = 50 pF, *Figure 2*			80	ns
t_{RHEOC}	$\overline{RASIN}$ High to End-of-Count High	C_L = 50 pF, *Figure 2*			80	ns
t_{RGEOB}	RGCK Low to End-of-Burst Low	C_L = 50 pF, *Figure 4*			95	ns
t_{MCEOB}	Mode Change to End-of-Burst High	C_L = 50 pF, *Figure 4*			75	ns
t_{RST}	Counter Reset Pulse Width	*Figure 2*	70			ns
t_{CTL}	RF I/O Low to Counter Outputs All Low	*Figure 2*			100	ns
$t_{RFCKL, H}$	Minimum Pulse Width of RFCK	*Figure 9*	100			ns
T	Period of $\overline{RAS}$ Generator Clock	*Figure 3*	100			ns
t_{RGCKL}	Minimum Pulse Width Low of RGCK	*Figure 3*	35			ns
t_{RGCKH}	Minimum Pulse Width High of RGCK	*Figure 3*	35			ns
t_{FRQL}	RFCK Low to Forced $\overline{RFRQ}$ Low	C_L = 50 pF, *Figure 3*		20	30	ns

Switching Characteristics: DP8409A-2 (Continued)

V_{CC} = 5.0V ±5%, 0°C ≤ T_A ≤ 70°C (unless otherwise noted) (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, (unless otherwise noted). See *Figure 11* for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7 kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Parameter	Conditions	8409A-2			Units
			Min	Typ	Max	
REFRESH (Continued)						
t_{FRQH}	RGCK Low to Forced $\overline{RFRQ}$ High	C_L = 50 pF, *Figure 3*		50	75	ns
t_{RGRL}	RGCK Low to $\overline{RAS}$ Low	*Figure 3*	50	65	95	ns
t_{RGRH}	RGCK Low to $\overline{RAS}$ High	*Figure 3*	40	60	85	ns
t_{RQHRF}	$\overline{RFSH}$ Hold Time from $\overline{RFSH}$ $\overline{RQST}$ (RF I/O)	*Figure 3*	2T			ns
t_{RFRH}	$\overline{RFSH}$ High to $\overline{RAS}$ High (Ending Forced RFSH)	See Mode 1 Descrip.	55	80	110	ns
t_{RFSRG}	$\overline{RFSH}$ Low Set-Up to RGCK Low (Mode 1)	See Mode 1 Descrip.	35			ns
t_{CSCT}	$\overline{CS}$ High to RFSH Counter Valid	*Figure 9*		55	70	ns
t_{CSRL}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low	See Mode 5 Descrip.	30			ns
t_{ZH}	$\overline{CS}$ Low to Address Output High from Hi-Z	*Figures 9, 12,* R1 = 3.5k, R2 = 1.5k		35	60	ns
t_{HZ}	$\overline{CS}$ High to Address Output Hi-Z from High	C_L = 15 pF, *Figures 9, 12,* R2 = 1k, S1 Open		20	40	ns
t_{ZL}	$\overline{CS}$ Low to Address Output Low from Hi-Z	*Figures 9, 12,* R1 = 3.5k, R2 = 1.5k		35	60	ns
t_{LZ}	$\overline{CS}$ High to Address Output Hi-Z from Low	C_L = 15 pF, *Figures 9, 12,* R1 = 1k, S2 Open		25	50	ns
t_{HZH}	$\overline{CS}$ Low to Control Output High from Hi-Z High	*Figures 9, 12,* R2 = 750Ω, S1 Open		50	80	ns
t_{HHZ}	$\overline{CS}$ High to Control Output Hi-Z High from High	C_L = 15 pF, *Figures 9, 12,* R2 = 750Ω, S1 Open		40	75	ns
t_{HZL}	$\overline{CS}$ Low to Control Output Low from Hi-Z High	*Figure 12,* S1, S2 Open		45	75	ns
t_{LHZ}	$\overline{CS}$ High to Control Output Hi-Z High from Low	C_L = 15 pF, *Figure 12,* R2 = 750Ω, S1 Open		50	80	ns

Input Capacitance T_A = 25°C (Notes 2, 6)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{IN}	Input Capacitance ADS, R/$\overline{C}$			8		pF
C_{IN}	Input Capacitance All Other Inputs			5		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: This test is provided as a monitor of Driver output source and sink current capability. Caution should be exercised in testing these parameters. In testing these parameters, a 15Ω resistor should be placed in series with each output under test. One output should be tested at a time and test time should not exceed 1 second.

Note 4: Input pulse 0V to 3.0V, t_R = t_F = 2.5 ns, f = 2.5 MHz, t_{PW} = 200 ns. Input reference point on AC measurements is 1.5V. Output reference points are 2.7V for High and 0.8V for Low.

Note 5: The load capacitance on RF I/O should not exceed 50 pF.

Note 6: Applies to all DP8409A versions unless otherwise specified.

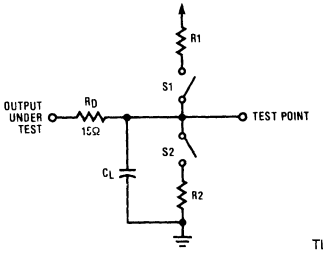
Note 7: The DP8409A-2 device can only be used with memory devices that meet the t_{RAH} specification indicated.

Applications

If external control is preferred, the DP8409A may be used in Mode 0 or 4, as in *Figure 6*.

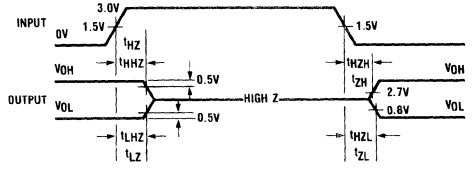
If basic auto access and refresh are required, then in cases where the user requires the minimum of external complexity, Modes 1 and 5 are ideal, as shown in *Figure 13a*. The DP843X2 is used to provide proper arbitration between memory access and refresh. This chip supplies all the necessary control signals to the processor as well as the DP8409A. Furthermore, two separate $\overline{CAS}$ outputs are also included for systems using byte-writing. The refresh clock RFCK may be divided down from either RGCK using an IC counter such as the DM74LS393 or better still, the DP84300 Programmable Refresh Timer. The DP84300 can provide RFCK periods ranging from 15.4 μs to 15.6 μs based on the input clock of 2 to 10 MHz. *Figure 13b* shows the general timing diagram for interfacing the DP8409A to different microprocessors using the interface controller DP843X2.

If the system is complex, requiring automatic access and refresh, burst refresh, and all-banks auto-write, then more circuitry is required to select the mode. This may be accomplished by utilizing a PAL®. The PAL has two functions. One as an address comparator, so that when the desired port address occurs (programmed in the PAL), the comparator gates the data into a latch, where it is connected to the mode pins of the DP8409A. Hence the mode of the DP8409A can be changed as desired with one PAL chip merely by addressing the PAL location, and then outputting data to the mode-control pins. In this manner, all the automatic modes may be selected, assigning R/C as RFCK always, and $\overline{CASIN}$ as RGCK always. The output from RF I/O may be used as End-of-Count to an interrupt, or $\overline{Refresh\ Request}$ to $\overline{HOLD}$ or $\overline{BUS\ REQUEST}$. A complex system may use Modes 5 and 1 for automatic access and refresh, Modes 3a and 7 for system initialization, and Mode 2 (auto-burst refresh) before and after DMA.



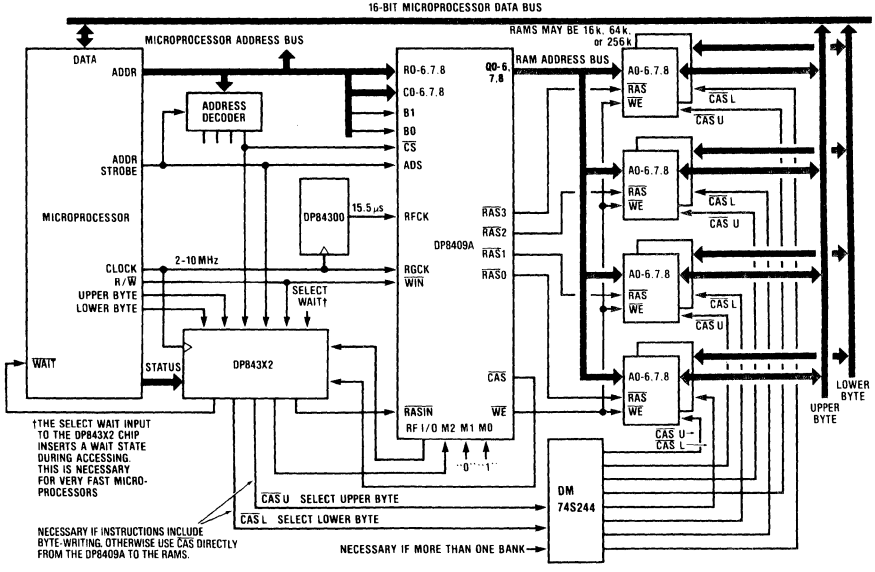
TL/F/8409–21

FIGURE 11. Output Load Circuit



TL/F/8409–22

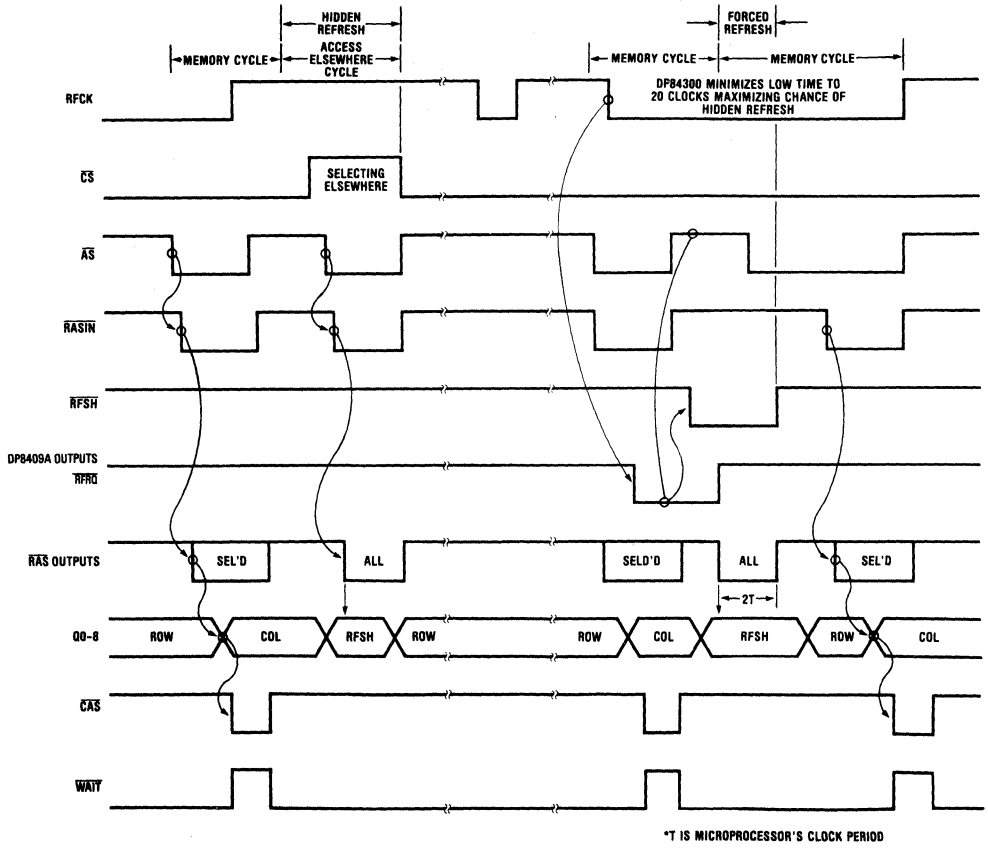
FIGURE 12. Waveform



TL/F/8409–23

FIGURE 13a. Connecting the DP8409A Between the 16-Bit Microprocessor and Memory

Applications (Continued)



TL/F/8409–24

FIGURE 13b. DP8409A Auto Refresh

National Semiconductor Corporation

DP84412 Dynamic RAM Controller Interface Series Circuit for the Series 32000® CPU

General Description

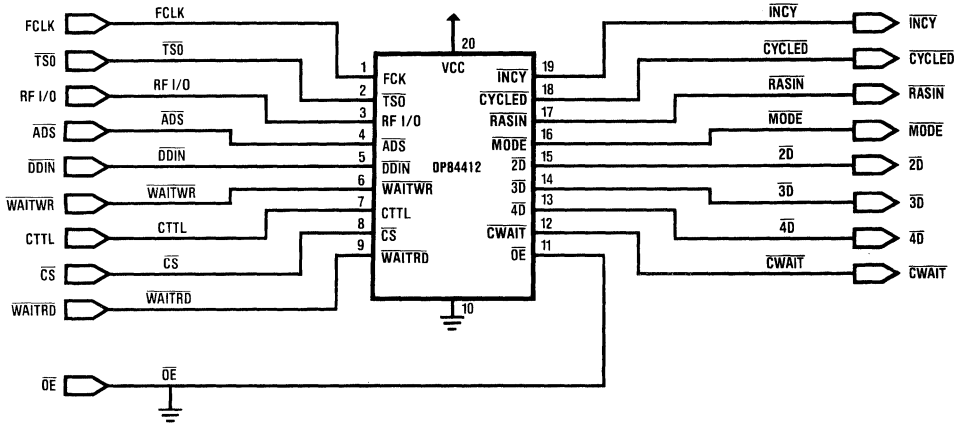
The DP84412 is a new Programmable Array Logic (PAL®) device, that replaces the DP84312, designed to allow an easy interface between the National Semiconductor Series 32000 family of processors and the National Semiconductor DP8409A, DP8429, or DP8419 DRAM controller.

The new DP84412 supplies all the control signals needed to perform memory read, write and refresh and work with the National Semiconductor Series 32000 family of processors up to 10 MHz. Logic is also included to insert WAIT states, if wanted, into the microprocessor READ or WRITE cycles when using fast CPUs.

Features

- Provides a 3-chip solution for the Series 32000 family, dynamic RAM interface (DP8409A or DP8419, DP84412, and clock divider).
- Works with all Series 32000 family speed versions up to 10 MHz.
- Operation of Series 32000 processor at 10 MHz with no WAIT states.
- Controls DP8409A or DP8419 Mode 5 accesses, hidden refreshes and Mode 1 Forced Refreshes automatically.
- Inserts WAIT states in READ or WRITE cycles automatically depending on whether $\overline{\text{WAITRD}}$ or $\overline{\text{WAITWR}}$ are low, or if $\overline{\text{CS}}$ becomes active during a forced Refresh cycle.
- Uses a standard National Semiconductor PAL part (DMPAL16R6A).
- The PAL logic equations can be modified by the user for his specific application and programmed into any of the PALs in the National Semiconductor family, including the new very high speed PALs ("B" PAL parts).

Connection Diagram



TL/F/8397–1

Order Number DP84412J or DP84412N
See NS Package Number N20A or J20A

Absolute Maximum Ratings

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

	Operating	Programming
Supply Voltage, V_{CC}	7V	12V
Input Voltage	5.5V	12V

	Operating	Programming
Off-State Output Voltage	5.5V	12V
Storage Temperature Range	−65°C to +150°C	

Recommended Operating Conditions

Symbol	Parameter		Commercial			Units
			Min	Typ	Max	
V_{CC}	Supply Voltage		4.75	5	5.25	V
t_w	Width of Clock	Low	15	10		ns
		High	15	10		
t_{su}	Setup Time from Input or Feedback to Clock		25	16		ns
t_h	Hold Time		0	−10		ns
T_A	Operating Free-Air Temperature		0	25	75	°C
T_C	Operating Case Temperature					°C

Electrical Characteristics Over Recommended Operating Temperature Range

Symbol	Parameter	Test Conditions		Min	Typ	Max	Units
V_{IH}	High Level Input Voltage			2			V
V_{IL}	Low Level Input Voltage					0.8	V
V_{IC}	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−0.8	−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, V_{IL} = 0.8V, V_{IH} = 2V	I_{OH} = −3.2 mA COM	2.4	2.8		V
V_{OL}	Low Level Output Voltage	V_{CC} = Min, V_{IL} = 0.8V, V_{IH} = 2V	I_{OL} = 24 mA COM		0.3	0.5	V
I_{OZH}	Off-State Output Current	V_{CC} = Max, V_{IL} = 0.8V, V_{IH} = 2V	V_O = 2.4V			100	μA
I_{OZL}			V_O = 0.4V			−100	μA
I_I	Maximum Input Current	V_{CC} = Max, V_I = 5.5V				1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.4V				25	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.02	−0.25	mA
I_{OS}	Output Short-Circuit Current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply Current	V_{CC} = Max			120	180	mA

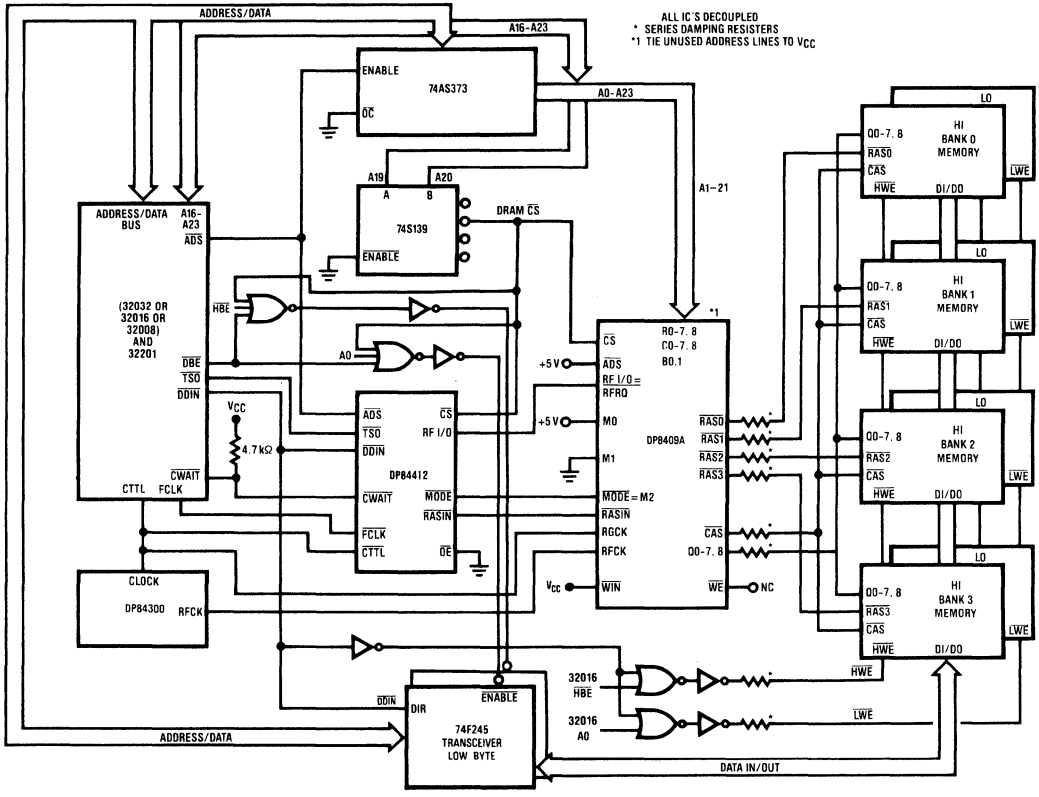
Switching Characteristics Over Recommended Ranges of Temperature and V_{CC}

V_{CC} = 5V ±10%. Commercial: T_A = 0°C to 75°C, V_{CC} = 5V ±5%

Symbol	Parameter	Test Conditions R1, R2	Commercial			Units
			Min	Typ	Max	
t_{PD}	Input or Feedback to Output	CL = 50 pF		15	25	ns
t_{CLK}	Clock to Output or Feedback			10	15	ns
t_{PZX}	Pin 11 to Output Enable			10	20	ns
t_{PXZ}	Pin 11 to Output Disable	C_L = 5 pF		11	20	ns
t_{PZX}	Input to Output Enable	C_L = 50 pF		10	25	ns
t_{PXZ}	Input to Output Disable	C_L = 5 pF		13	25	ns
f_{MAX}	Maximum Frequency		25	30		ns

V_{CC} = Max at minimum temperature.

PAL For Series 32000 Family Systems



TL/F/8397-2

Mnemonic Description

INPUTS SIGNALS

1) "FCLK" Fast clock from the NS32201 TCU clock chip, this signal runs at twice the speed of the system clock.
2) "$\overline{\text{TSO}}$" From the NS32201 TCU clock chip, this signal indicates the start of the "T2" state and goes high at the beginning of the "T4" state.
3) "RFI/O" $\overline{\text{RFRQ}}$ (refresh request) in mode 5. From 8409A, an active low signal.
4) "$\overline{\text{ADS}}$" From the Series 32000 CPU, address strobe. If the system includes the MMU (NS32082) then $\overline{\text{PAV}}$ should be connected to this input.
5) "$\overline{\text{DDIN}}$" Used to differentiate between READ and WRITE cycles, and to allow $\overline{\text{CS}}$ READ cycles to start early.
6) "$\overline{\text{WAITWRITE}}$" This signal is used to add a WAIT state into a $\overline{\text{CS}}$ WRITE access cycle, and delay $\overline{\text{RASIN}}$ until the end of the "T2" clock period.
7) "CTTL" From the NS32201 TCU clock chip, this signal runs at the system clock frequency.
8) "$\overline{\text{CS}}$" From decoder chip (chip select) (active low).
9) "$\overline{\text{WAITREAD}}$" Used to insert 1 wait state into the Series 32000 READ bus cycle. The wait state allows the use of memory with longer access times (t_{CAC}). An active low signal.
10) "$\overline{\text{OE}}$" This input enables the outputs of the "D-Flip Flop" outputs of the PAL.

OUTPUTS SIGNALS

1) "$\overline{\text{MODE}}$" This pin goes to M2 on the DP8409A to change from mode 5 to mode 1 (only used for forced refresh).
2) "$\overline{\text{2DLY}}$" Delay used internal to the PAL.
3) "$\overline{\text{3DLY}}$" Delay used internal to the PAL.
4) "$\overline{\text{4DLY}}$" Delay used internal to the PAL.
5) "$\overline{\text{RASIN}}$" To the 8409A (creates RASs). Goes low earlier for READ cycles than WRITE cycles.
6) "$\overline{\text{CYCLED}}$" Goes active low once a hidden refresh (non $\overline{\text{CS}}$ cycle) or DRAM access has been performed. $\overline{\text{CYCLED}}$ always goes low at the beginning of the "T3" processor state. This signal goes high (reset) by the end of the processor bus cycle as indicated by $\overline{\text{TSO}}$ being high.
7) "$\overline{\text{CWAIT}}$" This output inserts "WAIT" or "HOLD" states into the NS32016 machine cycles (only WAIT states are used in this application). This output is in "not enabled" condition when $\overline{\text{CS}}$ is high (not chip selected).
8) "$\overline{\text{INCYCLE}}$" This signal goes active from the CPU $\overline{\text{ADS}}$ signal. This signal indicates that the processor is doing an access somewhere in the system. This signal stays low for several T states of the access cycle.

Functional Description

The following description applies to both the DP8409A and the DP8419 dynamic RAM controllers.

A memory cycle starts when chip select ($\overline{\text{CS}}$) and address strobe ($\overline{\text{ADS}}$) are true. $\overline{\text{RASIN}}$ is supplied from the DP84412 to the DP8409A dynamic RAM controller, which then supplies a $\overline{\text{RAS}}$ signal to the selected dynamic RAM bank. After the necessary row address hold time, the DP8409A switches the address outputs to the column address. The DP8409A then supplies the required $\overline{\text{CAS}}$ signal to the DRAM. In order to do byte operations it is suggested that the user provide external logic, as shown in the system block diagram, to produce a HIGH WRITE ENABLE and/or a LOW WRITE ENABLE. To differentiate between a READ and a WRITE, the $\overline{\text{DDIN}}$ signal from the CPU is used. $\overline{\text{DDIN}}$ is also supplied to the external WRITE ENABLE logic.

A refresh cycle is started by one of two conditions. The refresh cycle caused by the first condition is called a hidden refresh. This occurs when refresh clock (RFCK) is high, $\overline{\text{CS}}$ is not true, and $\overline{\text{RASIN}}$ goes true. Here the CPU is accessing something else in the system and the DRAM can be refreshed at that time, thereby being transparent to the CPU. The second type of refresh is called forced refresh. This occurs if no hidden refresh was performed while RFCK was high. When RFCK transitions low a refresh request ($\overline{\text{RFRQ}}$) is generated. If there is not a DRAM access in progress the DP84412 will force a refresh by putting the DP8409A into mode 1 (automatic forced refresh mode). If the CPU tries to access the DRAM during a forced refresh cycle WAIT states will be inserted into its cycles until the forced refresh is over and the DRAM $\overline{\text{RAS}}$ precharge time has been met. Then the pending DRAM access will be allowed to take place.

The DP84412 also allows forced refreshes to take place during long accesses of other devices. For instance, if EEPROM takes several microseconds to write to, the DRAM will still be refreshed while that access is in progress.

In a standard memory cycle, the access can be slowed down by one clock cycle to accommodate slower memories or allow time to generate parity. This is accomplished by inserting a WAIT state into the processor access cycle. The DP84412 can insert WAIT states into either READ or WRITE cycles, or both. The extra WAIT state will not appear during the hidden refresh cycle, so faster devices on the CPU bus will not be affected.

System Interface Description

All members of the Series 32000 family of processors are able to use the DP84412.

The DP84412 differentiates between READ and WRITE cycles, allowing the $\overline{\text{RASIN}}$ signal to start earlier during a READ cycle compared to a WRITE cycle.

$\overline{\text{RASIN}}$ during a READ cycle will always start at the beginning of the "T2" processor cycle. The user must also guarantee that $\overline{\text{CS}}$ is valid a minimum of 30 ns before $\overline{\text{RASIN}}$ becomes valid. The worst case would be at 10 MHz where FCLK preceeds PHI1 by a maximum of 10 ns. $\overline{\text{RASIN}}$ can occur a minimum of approximately 8 ns after FCLK. Therefore $\overline{\text{CS}}$ must occur a minimum of 32 ns (30 ns + 2 ns) before the rising edge of PHI1 at 10 MHz.

The user may want to tie $\overline{\text{CS}}$ low on the DP8409A/19 (disable HIDDEN REFRESH) and use the system transceivers to select the DRAM. In this case one only needs to concern himself with the 10 ns address setup time to $\overline{\text{RASIN}}$.

System Interface Description (Continued)

The DP84412 can be used in a system with the MMU (NS32082) but the signal $\overline{PAV}$ would be connected to the $\overline{ADS}$ input instead of $\overline{ADS}$.

Several other critical parameters in this application that involve the input signals $\overline{DDIN}$, $\overline{CWAIT}$, $\overline{TSO}$, and FCLK. These parameters become most critical at 10 MHz where it is suggested that they are directly connected to the corresponding pins of the Series 32000 family ICs.

This section of the data sheet goes through the calculation of the "tRAC" ($\overline{RAS}$ access time) and "tCAC" ($\overline{CAS}$ access time) required by the DRAM for the Series 32000 family CPUs to operate at a particular clock frequency without introducing wait states into the processor access cycles. Both "tRAC" and "tCAC" must be considered in determining what speed DRAM can be used in a particular system design. The DRAM chosen must meet both the "tRAC" and "tCAC" parameters calculated. In order to determine the "tRAC" and "tCAC" needed the DP8419 and fast PALs ("B" type PALs) timing parameters were used. If the user is using the DP8408A/09A or a slower PAL device he should substitute their respective delays into the equations below.

Most all of the calculations contained in this note use "RAHS" = 1 (15 ns guaranteed minimum row address hold time). Calculations only used "RAHS" = 0 (25 ns guaranteed minimum row address hold time) when the calculated access time from RAS exceeded 200 ns. This is because DRAMs can be found with row access times up to 150 ns that require only 15 ns row address hold times.

EXAMPLE DRAM TIMING CALCULATIONS

A) 8 MHz Series 32000 CPU, No Wait states

#1) $\overline{RASIN}$ = T1 − 2 ns (FCLK to PHI1 skew) + 12 ns ("B" PAL clocked output) = 125 − 2 + 12 = 135 ns maximum

#2) $\overline{RASIN}$ to $\overline{RAS}$ low = 20 ns maximum (DP8419)

#3) $\overline{RASIN}$ to $\overline{CAS}$ low = 80 ns (DP8419 $\overline{RASIN}$ − $\overline{CAS}$ low) − 3 ns (load of 72 DRAMs instead of 88 DRAMs speced in data sheet) = 77 ns

#4) 74F245 transceiver delay = 7 ns maximum

#5) CPU data setup time to "T4" = data setup to PHI2 T.E. + maximum PHI2 F.E. to PHI1 R.E. = 15 + 5 = 20 ns minimum

"tRAC" = T1 + T2 + T3 − #1 − #2 − #4 − #5
= 125 + 125 + 125 − 135 − 20 − 7 − 20 = 193 ns

"tCAC" = T1 + T2 + T3 − #1 − #3 − #4 − #5
= 125 + 125 + 125 − 135 − 77 − 7 − 20 = 136 ns

Therefore the DRAM chosen should have a "tRAC" less then or equal to 193 ns and a "tCAC" less than or equal to 136 ns. Standard 150 ns DRAMs meet this criteria.

The minimum $\overline{RAS}$ PRECHARGE TIME will be approximately one and one half clock periods = 125 + 62 = 187 ns.

The minimum $\overline{CAS}$ PRECHARGE TIME will be approximately one and one half clock periods plus 35 ns (minimum $t_{RICL} - t_{RICH}$ for the DP8409-2) = 125 + 62 + 35 = 222 ns.

The minimum $\overline{RAS}$ PULSE WIDTH will be approximately two clock periods − 5 ns (maximum $t_{RPDL} - t_{RPDH}$ for the DP8409-2) = 250 − 5 = 245 ns.

The minimum $\overline{CAS}$ PULSE WIDTH will be approximately two clock periods − 70 ns (maximum $t_{RICL} - t_{RICH}$ for the DP8409-2) = 250 − 70 = 180 ns.

The smallest pulse widths are generated during WRITE cycles since $\overline{RASIN}$ during WRITE cycles starts later than $\overline{RASIN}$ during READ cycles.

If one inserted a WAIT state in READ cycles the DRAM column access times and the $\overline{RAS}$ pulse width would be increased by one clock period (125 ns in this case). A WAIT state in WRITE cycles would just increase the $\overline{RAS}$ pulse width by one clock period.

B) 10 MHz Series 32000, No Wait States

#1) $\overline{RASIN}$ low = T1 − 2 ns (FCLK − PHI1 skew) + 12 ns ("B" PAL clocked output) = 100 − 2 + 12 = 110 ns maximum

#2) $\overline{RASIN}$ to $\overline{RAS}$ low = 20 ns maximum

#3) $\overline{RASIN}$ to $\overline{CAS}$ low = 80 ns maximum (DP8419 $\overline{RASIN}$ − $\overline{CAS}$ low) − 3 ns (load of 72 DRAMs instead of 88 DRAMs speced in data sheet) = 77 ns

#4) 74F245 transceiver delay = 7 ns maximum

#5) CPU data setup time to "T4" = data setup to PHI2 T.E. + maximum PHI2 F.E. to PHI1 R.E. = 15 + 5 = 15 ns minimum

"tRAC" = T1 + T2 + T3 − #1 − #2 − #4 − #5
= 100 + 100 + 100 − 110 − 20 − 7 − 15 = 148 ns

"tCAC" = T1 + T2 + T3 − #1 − #3 − #4 − #5
= 100 + 100 + 100 − 110 − 77 − 7 − 15 = 91 ns

Therefore the DRAM chosen should have a "tRAC" less then or equal to 148 ns and a "tCAC" less than or equal to 91 ns. Standard 120 ns DRAMs meet this criteria.

The minimum $\overline{RAS}$ PRECHARGE TIME will be approximately one and one half clock periods = 100 + 50 = 150 ns.

The minimum $\overline{CAS}$ PRECHARGE TIME will be approximately one and one half clock periods plus 35 ns (minimum $t_{RICL} - t_{RICH}$ for the DP8409-2) = 100 + 50 + 35 = 185 ns.

The minimum $\overline{RAS}$ PULSE WIDTH will be approximately two clock periods − 5 ns (maximum $t_{RPDL} - t_{RPDH}$ for the DP8409-2) = 200 − 5 = 195 ns.

The minimum $\overline{CAS}$ PULSE WIDTH will be approximately two clock periods − 70 ns (maximum $t_{RICL} - t_{RICH}$ for the DP8409-2) = 200 − 70 = 130 ns.

The smallest pulse widths are generated during WRITE cycles since $\overline{RASIN}$ during WRITE cycles starts later than $\overline{RASIN}$ during READ cycles.

If one inserted a WAIT state in READ cycles the DRAM column access times and the $\overline{RAS}$ pulse width would be increased by one clock period (100 ns in this case). A WAIT state in WRITE cycles would just increase the $\overline{RAS}$ pulse width by one clock period.

SUGGESTIONS

It is suggested that the DP8409A could be used up to 8 MHz. Above 8 MHz one should use the DP8409-2 or the DP8419. Also, fast PALs ("A" or "B" parts) should be used at 8 MHz and above.

INTERPRETING THE DP84412 PAL EQUATIONS

The boolean equations for the DP84412 were written using the standard PALASM™ format. In other words the equation: "IF (V_{CC}) RASIN = INCY*$\overline{MODE}$*4D*DDIN" will mean;

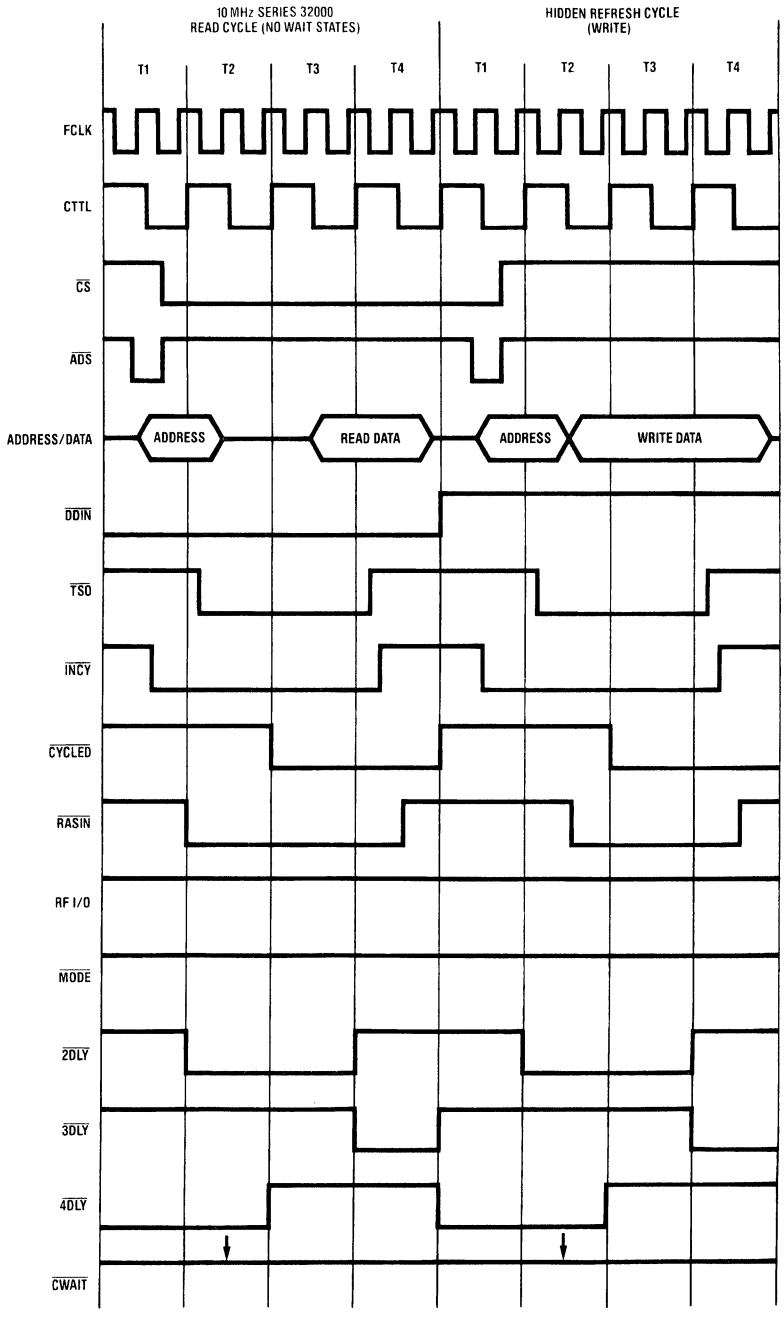
The output "$\overline{RASIN}$" (see pin list for DP84412) will be active low (inverted RASIN) when the output "$\overline{INCY}$" is low (making INCY high) AND the output "$\overline{MODE}$" is high AND the output "$\overline{4D}$" is low (making 4D high) AND the input $\overline{DDIN}$ is low (making DDIN high).

PAL Boolean Equations

```
PAL16R6A     ;FAST PAL
NEW PAL FOR THE NATIONAL SEMICONDUCTOR NS32016, 32008, 32032
NATIONAL SEMICONDUCTOR (WORKS UP 10 MHz)

FCLK /TSO RFIO /ADS /DDIN /WAITWR CTTL /CS /WAITRD GND
/OE /CWAIT /4DLY /3DLY /2DLY /MODE /RASIN /CYCLED /INCY VCC

RASIN := INCY*/CYCLED*/MODE*/CTTL*DDIN+      ;Start /RASIN fast during
                                             ; "READ" cycle
         INCY*/MODE*2DLY*/WAITWR+            ; ''WRITE'' cycle without WAIT states
         /CS•INCY•/MODE•2DLY+                ; Hidden Refresh /RASIN
         CS•INCY•/MODE•2DLY•WAITWR•/CTTL+    ; ''WRITE'' cycle with WAIT states
         RASIN•INCY•/MODE•2DLY               ; continue /RASIN

CYCLED := /MODE*2DLY*/WAITWR*/DDIN*/CTTL+        ;No WAITS inserted
          /MODE*2DLY*/WAITRD*DDIN*/CTTL+         ;No WAITS inserted
          /MODE*2DLY*/4DLY*WAITRD*DDIN*/CTTL+    ;WAIT in READ cycle
          /MODE*2DLY*/4DLY*WAITWR*/DDIN*/CTTL+   ;WAIT in WRITE cycle
          CYCLED*TSO*/MODE+
          CYCLED*/MODE*CTTL

MODE := /RFIO*/INCY*2DLY*/CTTL+              ;forced refresh during idle
                                             ;states, in long cycles,
        MODE*/3DLY+                          ;or at the end of a cycle
        MODE*/4DLY+
        MODE*CTTL

2DLY := MODE*/4DLY*/CTTL+
        2DLY*CTTL+
        INCY*/CYCLED*/MODE*/3DLY*/4DLY*/CTTL+
        CS*DDIN*WAITRD*INCY*/MODE*2DLY*/3DLY*/4DLY+   ;extend 2DLY if
        CS*/DDIN*WAITWR*INCY*/MODE*2DLY*/3DLY*/4DLY   ;  WAIT states
                                                      ;  are wanted

3DLY := 2DLY*/4DLY*/CTTL+
        3DLY*CTTL

4DLY := 3DLY*/CTTL+
        4DLY*CTTL+
        /INCY*/MODE*/CTTL+
        INCY*/MODE*/2DLY*/CTTL

IF (VCC) INCY = ADS*/MODE+
           CS*TSO*/CYCLED*/MODE*/2DLY*/4DLY+   ;Start INCY for CS
           INCY*/CYCLED+                       ;access after forced
           INCY*2DLY                           ;refresh

IF (CS) CWAIT=CS*TSO*/CYCLED*/MODE*/2DLY*/4DLY+   ;for Access during
                                                  ;forced refresh
   CS*TSO*MODE+                                   ;during forced refresh
   CS*INCY*/CYCLED*DDIN*WAITRD*/MODE*2DLY*/3DLY*/4DLY+
                                                  ; /CS READ cycle with
                                                  ; WAIT states
   CS*INCY*/CYCLED*/DDIN*WAITWR*/MODE*2DLY*/3DLY*/4DLY
                                                  ; /CS WRITE cycle with
                                                  ; WAIT states
```

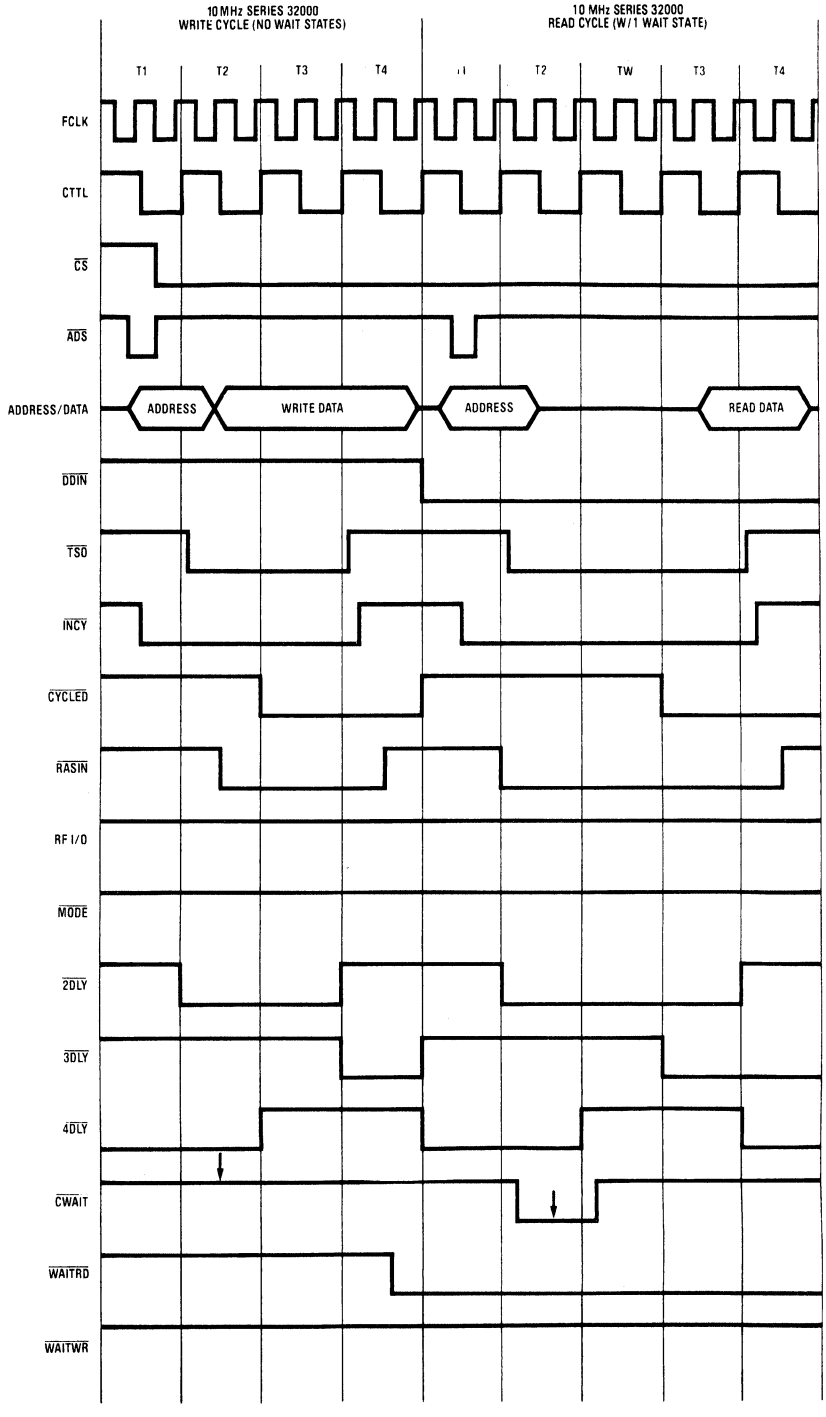
FIGURE 1. Equations for the Series 32000 Family Interface PAL

System Timing Diagrams



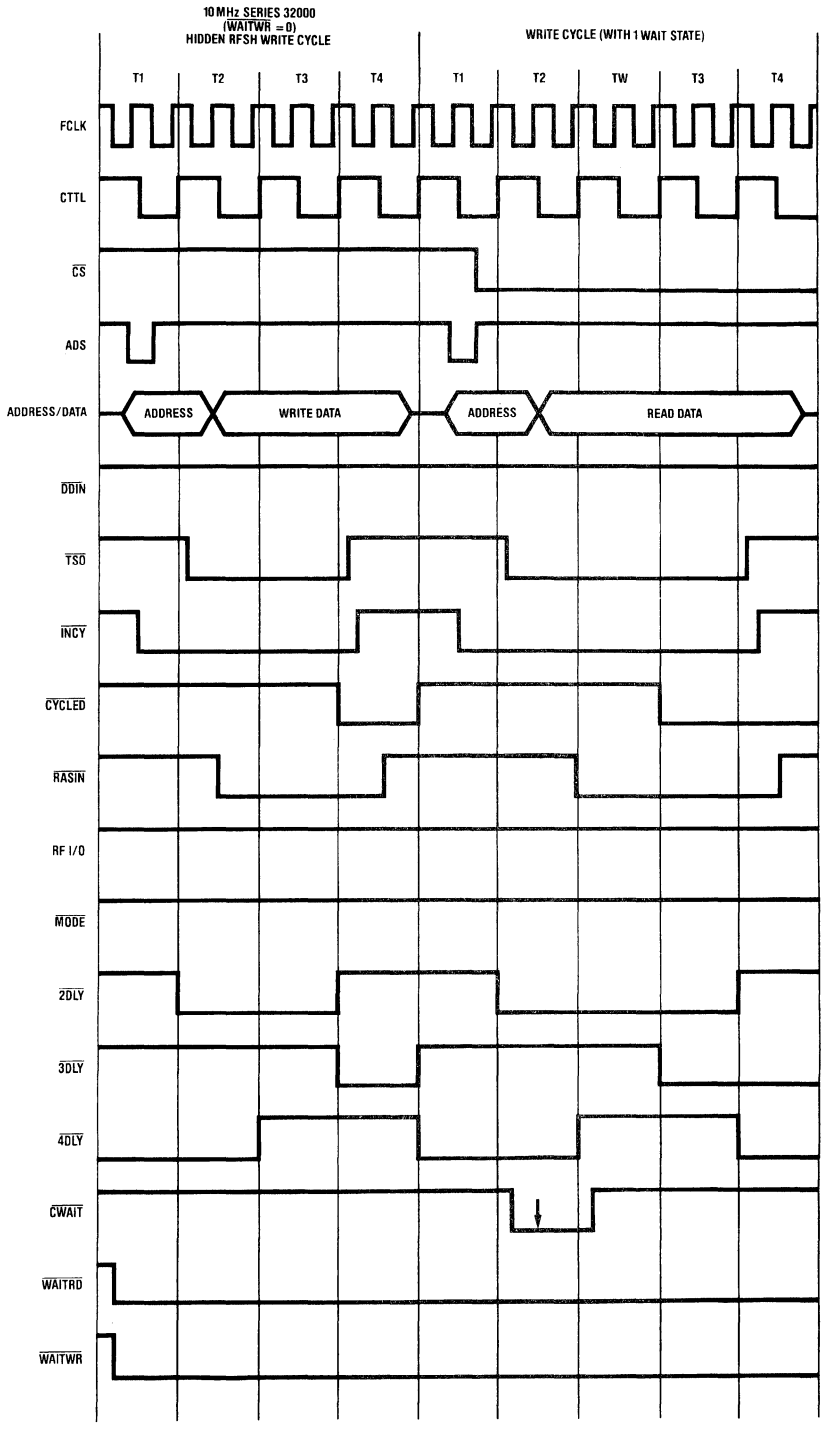
TL/F/8397–3

System Timing Diagrams (Continued)



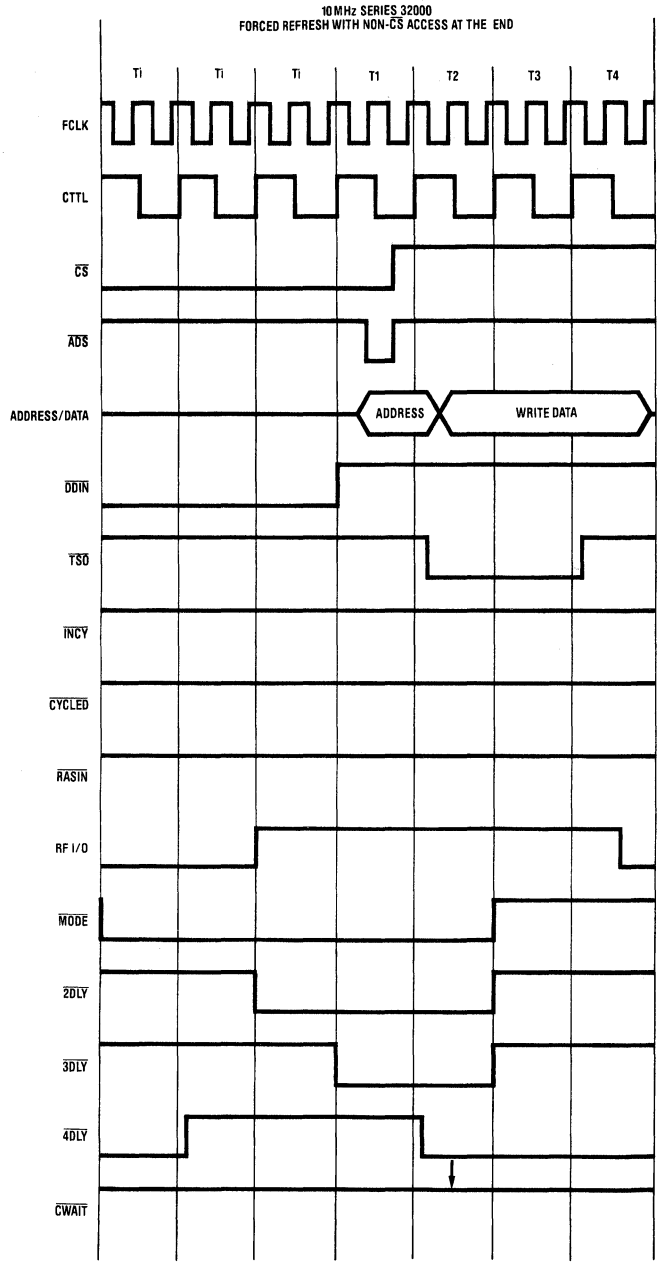
TL/F/8397–4

System Timing Diagrams (Continued)



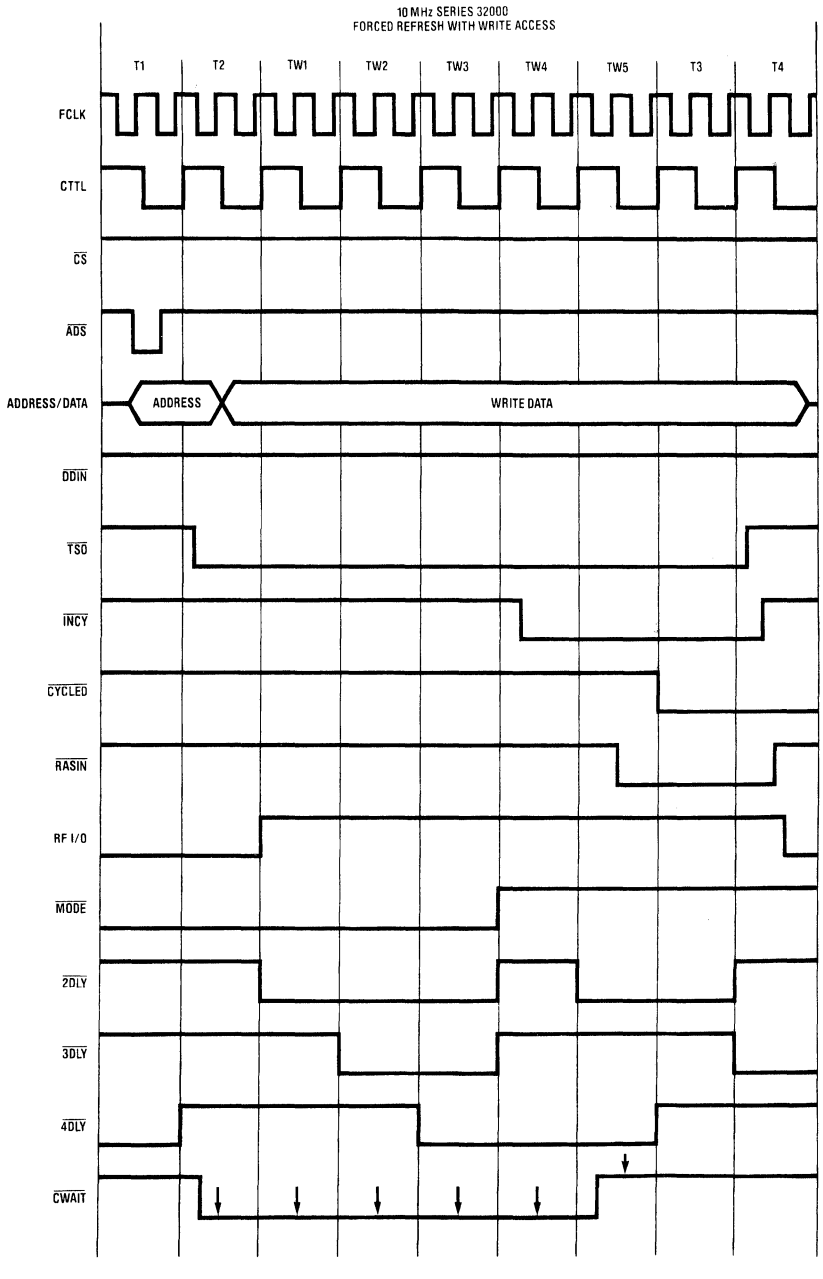
TL/F/8397–5

System Timing Diagrams (Continued)



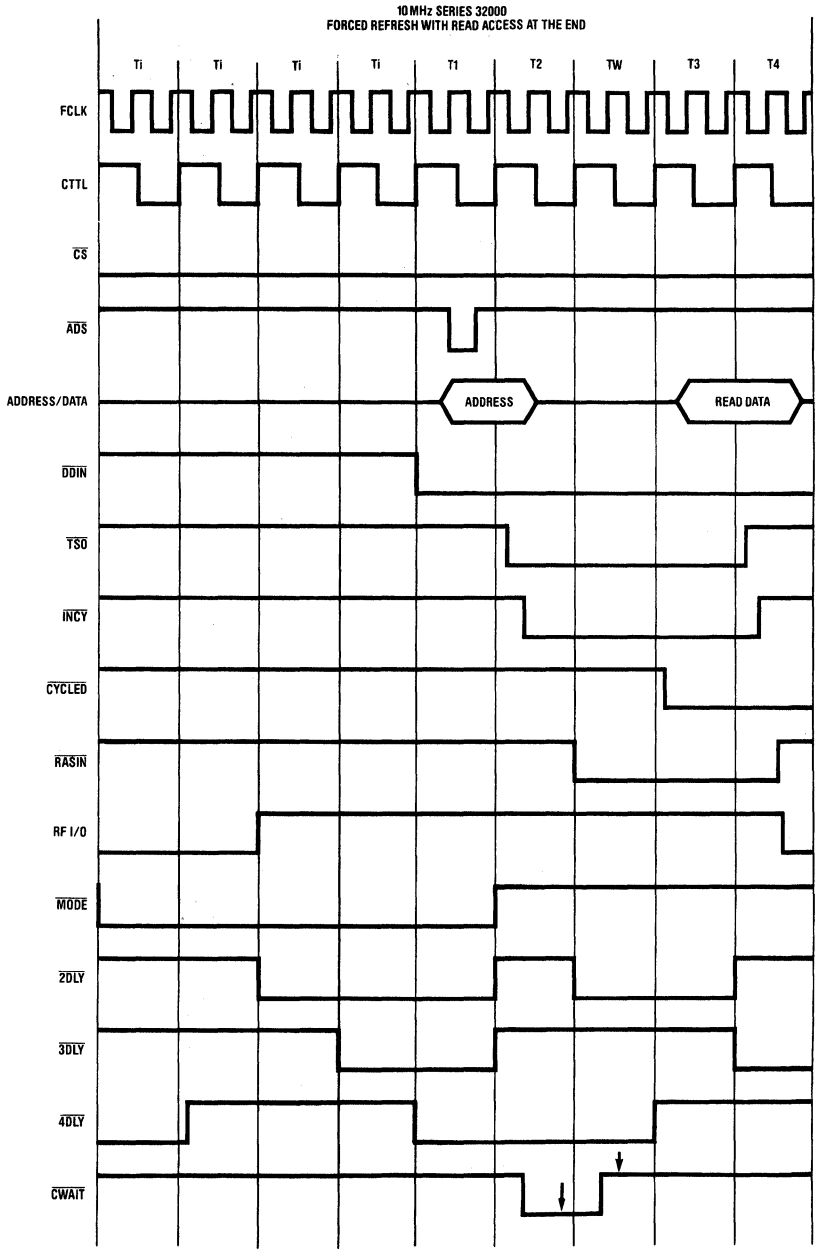
TL/F/8397–6

System Timing Diagrams (Continued)



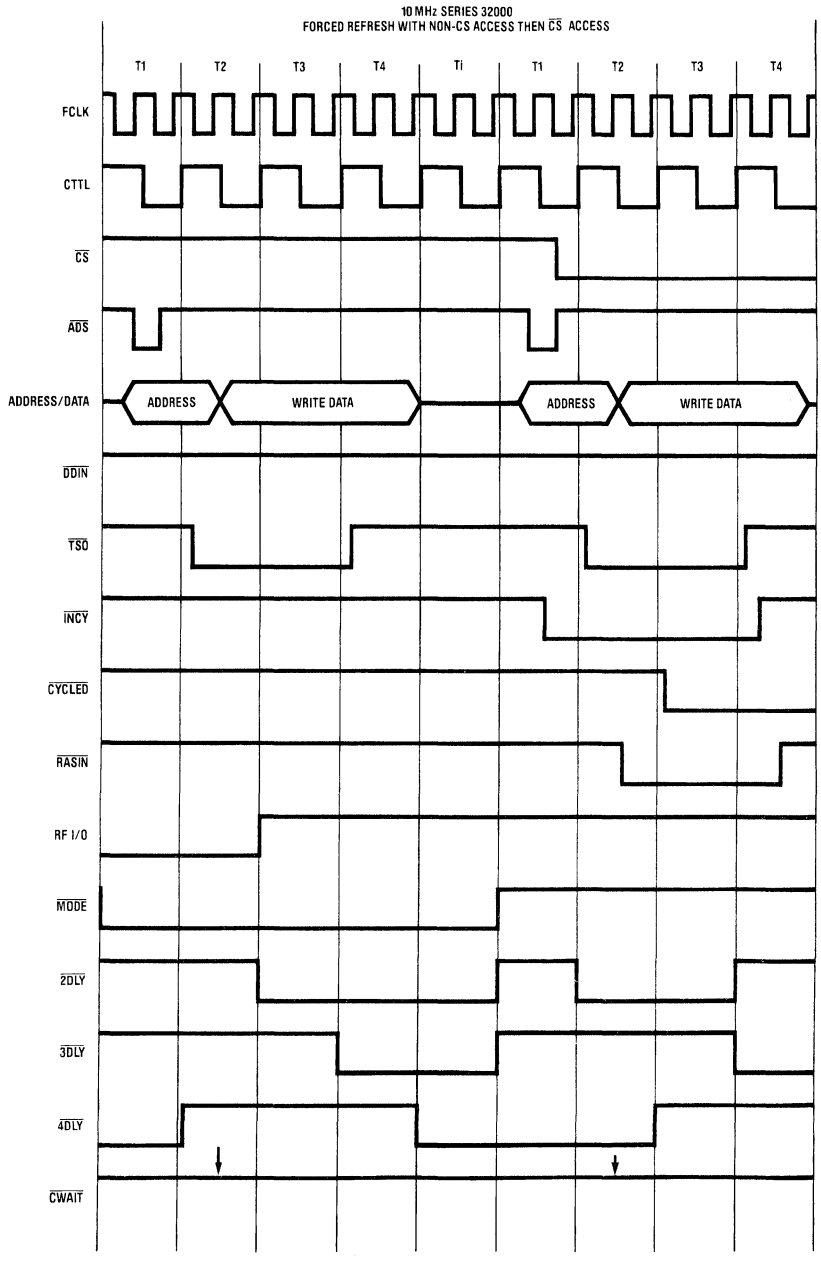
TL/F/8397–7

System Timing Diagrams (Continued)



TL/F/8397–8

System Timing Diagrams (Continued)



TL/F/8397–9

National Semiconductor Corporation

PRELIMINARY

DP8417/NS32817, 8418/32818, 8419/32819, 8419X/ 32819X 64k, 256k Dynamic RAM Controller/Drivers

General Description

The DP8417/8418/8419/8419X represent a family of 256k DRAM Controller/Drivers which are designed to provide "No-Waitstate" CPU interface to Dynamic RAM arrays of up to 2 Mbytes and larger. Each device offers slight functional variations of the DP8419 design which are tailored for different system requirements. All family members are fabricated using National's new oxide isolated Advanced Low power Schottky (ALS) process and use design techniques which enable them to significantly out-perform all other LSI or discrete alternatives in speed, level of integration, and power consumption.

Each device integrates the following critical 256k DRAM controller functions on a single monolithic device: ultra precise delay line; 9-bit refresh counter; fall-through row, column, and bank select input latches; Row/Column address muxing logic; on-board high capacitive-load $\overline{RAS}$, $\overline{CAS}$, and Write Enable & Address output drivers; and, precise control signal timing for all the above.

There are four device options of the basic DP8419 Controller. The DP8417 is pin and function compatible with the DP8419 except that its outputs are TRI-STATE®. The DP8418 changes one pin and is specifically designed to offer an optimum interface to 32 bit microprocessors. The DP8419X is functionally identical to the DP8419, but is available in a 52-pin DIP package which is upward pin compatible with National's new DP8429D 1 Mbit DRAM Controller/ Driver.

Each device is available in plastic DIP, Ceramic DIP, and Plastic Chip Carrier (PCC) packaging. (Continued)

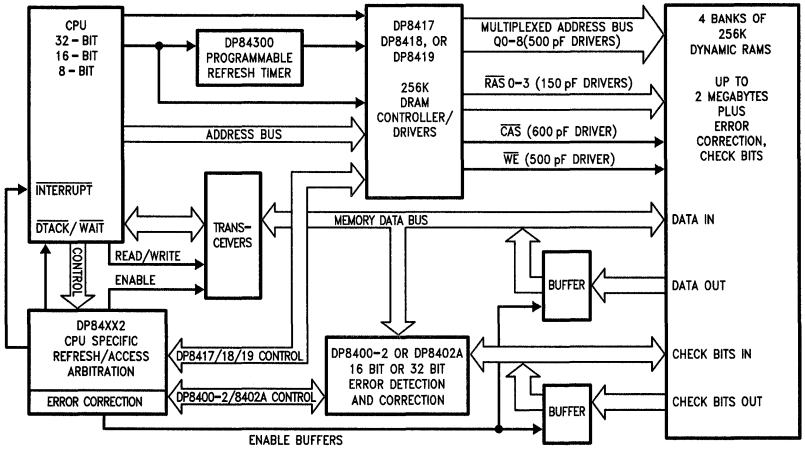
Operational Features

- Makes DRAM Interface and refresh tasks appear virtually transparent to the CPU, making DRAMs as easy to use as static RAMs
- Specifically designed to eliminate CPU wait states up to 10 MHz or beyond
- Eliminates 15 to 20 SSI/MSI components for significant board real estate reduction, system power savings and the elimination of chip-to-chip AC skewing
- On-board ultra precise delay line
- On-board high capacitive $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, and address drivers (specified driving 88 DRAMs directly)
- AC specified for directly addressing up to 8 Megabytes
- Low power/high speed bipolar oxide isolated process
- Upward pin and function compatible with new DP8428/ DP8429 1 Mbit DRAM controller drivers
- Downward pin and function compatible with DP8408A/ DP8409A 64k/256k DRAM controller/drivers
- 4 user selectable modes of operation for Access and Refresh (2 automatic, 2 external)

Contents

- System and Device Block Diagrams
- Recommended Companion Components
- Device Connection Diagrams and Pin Definitions
- Family Device Differences (DP8419 vs DP8409A, 8417, 8418, 8419X)
- Mode of Operation (Descriptions and Timing Diagrams)
- Application Description and Diagrams
- DC/AC Electrical Specifications, Timing Diagrams and Test Conditions

System Diagram



TL/F/8396-25

General Description (Continued)

In order to specify each device for "true" worst case operating conditions, all timing parameters are guaranteed while the chip is driving the capacitive load of 88 DRAMs including trace capacitance. The chip's delay timing logic makes use of a patented new delay line technique which keeps A.C. skew to ±3 ns over the full V_{CC} range of ±10% and temperature range of −55°C to +125°C. The DP8417, DP8418, DP8419, and DP8419X guarantee a maximum $\overline{RASIN}$ to CASOUT delay of 80 ns or 70 ns even while driving a 2 Mbyte memory array with error correction check bits included. Speed selected options of these devices are shown in the switching characteristics section of this document.

With its four independent $\overline{RAS}$ outputs and nine multiplexed address outputs, the DP8419 can support up to four banks of 16k, 64k or 256k DRAMs. Two bank select pins, B1 and B0, are decoded to activate one of the $\overline{RAS}$ signals during an access, leaving the three non-selected banks in the standby mode (less than one tenth of the operating power) with data outputs in TRI-STATE.

The DP8419 has two mode-select pins, allowing for two refresh modes and two access modes. Refresh and access timing may be controlled either externally or automatically. The automatic modes require a minimum of input control signals.

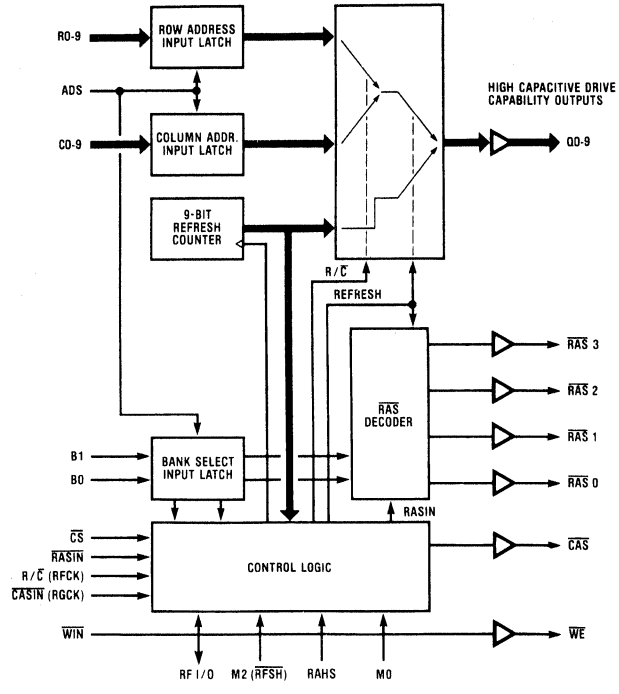
A refresh counter is on-chip and is multiplexed with the row and column inputs. Its contents appear at the address outputs of the DP8419 during any refresh, and are incremented at the completion of the refresh. Row/Column and bank address latches are also on-chip. However, if the address inputs to the DP8419 are valid throughout the duration of the access, these latches may be operated in the fall-through mode.

System Companion Components

Device #	Function
DP84300	Programmable Refresh Timer for DP84xx DRAM Controller
DP84412	NS32008/16/32 to DP8409A/17/18/19/28/29 Interface
DP84512	NS32332 to DP8417/18/19/28/29 Interface
DP84322	68000/08/10 to DP8409A/17/18/19/28/29 Interface (up to 8 MHz)
DP84422	68000/08/10 to DP8409A/17/18/19/28/29 Interface (up to 12.5 MHz)
DP84522	68020 to DP8417/18/19/28/29 Interface
DP84432	8086/88/186/188 to DP8409A/17/18/19/28/29 Interface
DP84532	80286 to DP8409A/17/18/19/28/29 Interface
DP8400-2	16-bit Expandable Error Checker/Corrector
DP8400-4	16-bit Expandable Error Checker/Corrector
DP8402A	32-bit Error Detector and Corrector (EDAC)

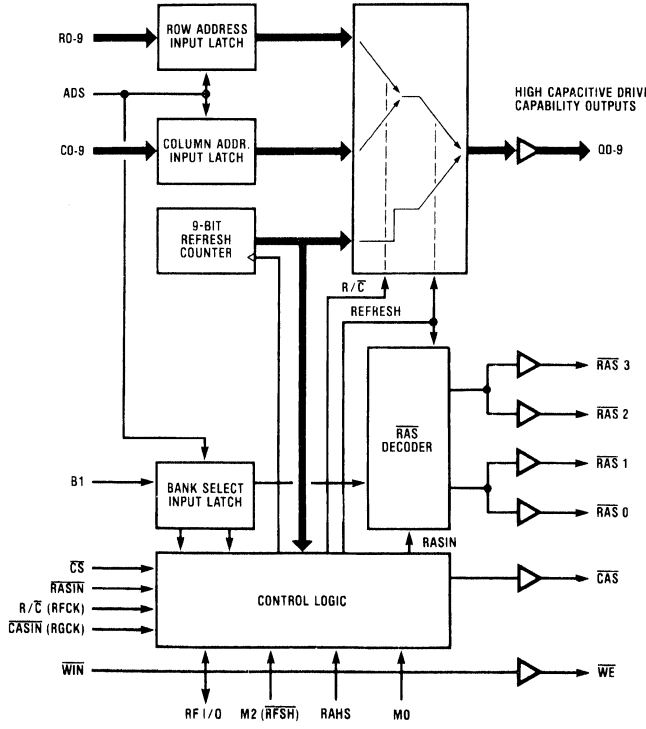
Block Diagrams

DP8417, 8419 and 8419X



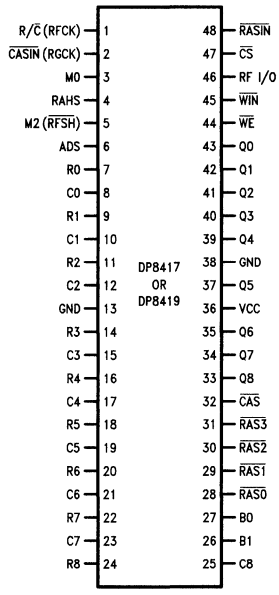
TL/F/8396–26

DP8418

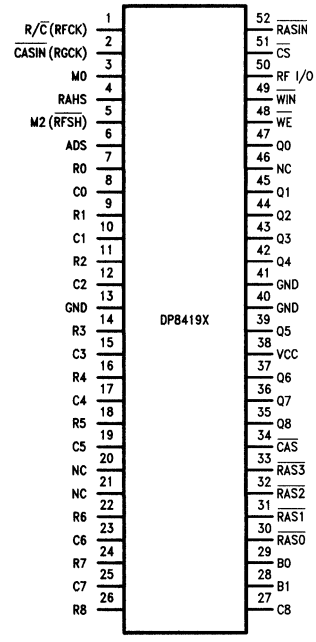


TL/F/8396–27

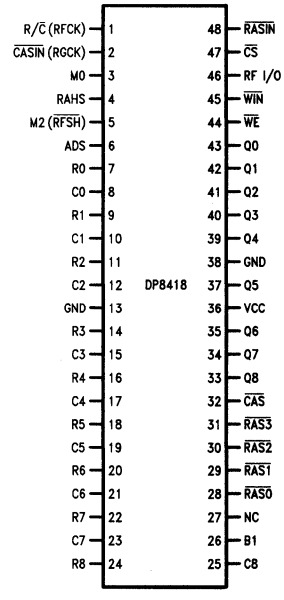
Connection Diagrams (Dual-In-Line Package)



TL/F/8396–28



TL/F/8396–29



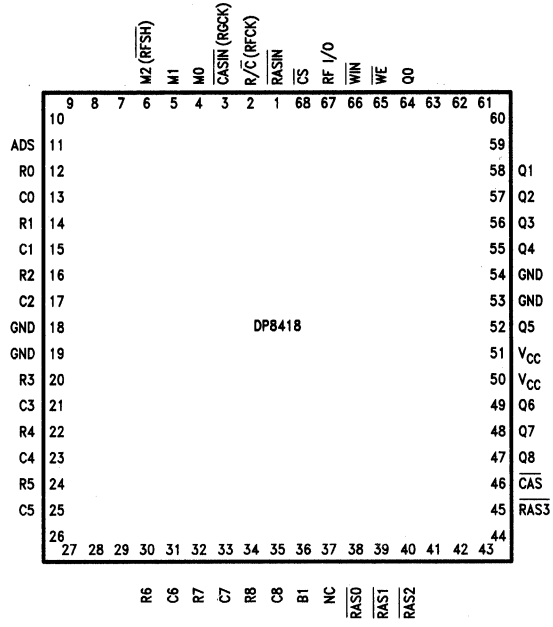
TL/F/8396–30

Order Number DP8417D-70, DP8417D-80, DP8417N-70, DP8417N-80, DP8418D-70, DP8418D-80, DP8418N-70, DP8418N-80, DP8419D-70, DP8419D-80, DP8419N-70, DP8419N-80, DP8419XD-70 or DP8419XD-80.
See NS Package Number D48A, D52A, or N48A

7

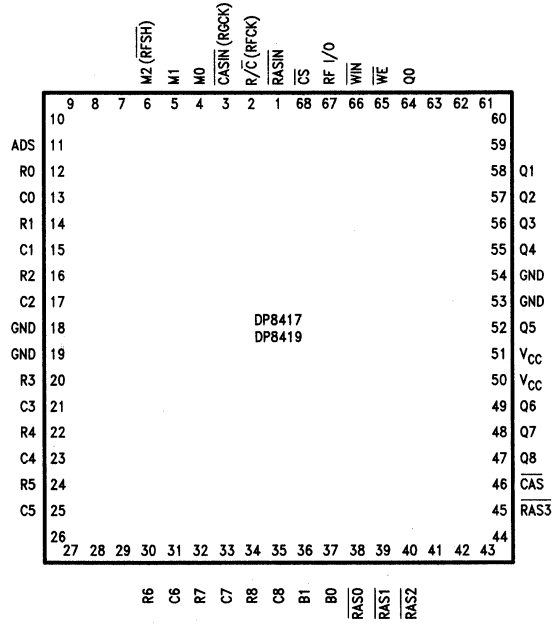
Connection Diagrams (Continued)

Plastic Chip Carrier Package



TL/F/8396–31

Plastic Chip Carrier Package



TL/F/8396–32

Order Number DP8417V-70, DP8417V-80, DP8418V-70, DP8418V-80, DP8419V-70 or DP8419V-80
See NS Package Number V68A

Family Device Differences

DP8417 vs DP8419

The DP8417 is identical to the DP8419 with the exception that its $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ and Q (Multiplexed Address) outputs are TRI-STATE when $\overline{CS}$ (Chip Select) is high and the chip is not in a refresh mode. This feature allows access to the same DRAM array through multiple DRAM Controller/Driver DP8417s. All AC specifications are the same as the DP8419 except t_{CSRLO} which is 34 ns for the DP8417 versus 5 ns for the DP8419. Separate delay specifications for the TRI-STATE timing paths are provided in the AC tables of this data sheet.

DP8418 vs DP8419

The DP8418 DYNAMIC RAM CONTROLLER/DRIVER is identical to the DP8419 with the exception of two functional differences incorporated to improve performance with 32-bit microprocessors.

1) Pin 26 (B1) is used to enable/disable a pair of $\overline{RAS}$ outputs, and pin 27 (B0 on the DP8419) is a no connect. When B1 is low, $\overline{RAS}$0 and $\overline{RAS}$1 are enabled such that they both go low during an access. When B1 is high, $\overline{RAS}$2 and $\overline{RAS}$3 are enabled. This feature is useful when driving words to 32 bits or more since each $\overline{RAS}$ would be driving only one half of the word. By distributing the load on each $\overline{RAS}$ line in this way, the DP8418 will meet the same AC specifications driving 2 banks of 32 DRAMs each as the DP8419 does driving 4 banks of 16 bits each.
2) The hidden refresh function available on the DP8419 has been disabled in order to reduce the amount of setup time necessary from $\overline{CS}$ going low to $\overline{RASIN}$ going low during an access of DRAM. This parameter, called t_{CSRL1}, is 5 ns for the DP8418 whereas it is 34 ns for the DP8419. The hidden refresh function only allows a very small increase in system performance, at best, at microprocessor frequencies of 10 MHz and above.

DP8419 vs DP8409A

The DP8419 High Speed DRAM Controller/Driver combines the most popular memory control features of the DP8408A/9A DRAM Controller/Driver with the high speed of bipolar oxide isolation processing.

The DP8419 retains the high capacitive-load drive capability of the DP8408A/9A as well as its most frequently used access and refresh modes, allowing it to directly replace the DP8408A/9A in applications using only modes 0, 1, 4 and 5. Thus, the DP8419 will allow most DP8408A/9A users to directly upgrade their system by replacing their old controller chip with the DP8419.

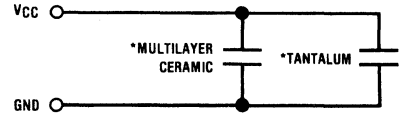
The highest priority of the DP8419 is speed. By peforming the DRAM address multiplexing, control signal timing and high-capacitive drive capability on a single chip, propagation delay skews are minimized. Emphasis has been placed on reducing delay variation over the specified supply and temperature ranges.

Except for the following, a DP8419 will operate essentially the same as a DP8409A.

1) The DP8419 has significantly faster AC performance.
2) The DP8419 can replace the DP8409A in applications which use modes 0, 1, 4, and 5. Modes 2, 3, 6, and 7 of the DP8409A are not available on the DP8419.
3) Pin 4 on the DP8419 is RAHS instead of M1, as on the DP8409A, and allows for two choices of t_{RAH} in mode 5.
4) RFI/O does not function as an end-of-count signal in Mode 0 on the DP8419 as it does on the DP8409A.
5) DP8419 address and control outputs do not TRI-STATE when $\overline{CS}$ is high as on the DP8409A. DP8419 control outputs are active high when $\overline{CS}$ is high (unless refreshing).

Pin Definitions

V_{CC}, GND, GND − V_{CC} = 5V ±10%. The three supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. There are two ground pins to reduce the low level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 9 address bits change in the same direction simultaneously. A recommended solution would be a 1 μF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected as close as possible to V_{CC} and GND to reduce lead inductance. See Figure below.



TL/F/8396–4

*Capacitor values should be chosen depending on the particular application.

R0–R8: Row Address Inputs.

C0–C8: Column Address Inputs.

Q0–Q8: Multiplexed Address Outputs - This address is selected from the Row Address Input Latch, the Column Address Input Latch or the Refresh Counter.

$\overline{RASIN}$: Row Address Strobe Input - $\overline{RASIN}$ directly controls the selected $\overline{RAS}$ output when in an access mode and all $\overline{RAS}$ outputs during hidden or external refresh.

R/$\overline{C}$ (RFCK) - In the auto-modes this pin is the external refresh clock input; one refresh cycle should be performed each clock period. In the external access mode it is Row/Column Select Input which enables either the row or column address input latch onto the output bus.

$\overline{CASIN}$ (RGCK) - In the auto-modes this pin is the $\overline{RAS}$ Generator Clock input. In external access mode it is the Column Address Strobe input which controls $\overline{CAS}$ directly once columns are enabled on the address outputs.

ADS: Address (Latch) Strobe Input - Row Address, Column Address, and Bank Select Latches are fall-through with ADS high; latching occurs on high-to-low transition of ADS.

$\overline{CS}$: Chip Select Input - When high, $\overline{CS}$ disables all accesses. Refreshing, however, in both modes 0 and 1 is not affected by this pin.

M0, M2 ($\overline{RFSH}$): Mode Control Inputs - These pins select one of the four available operational modes of the DP8419 (see Table III).

RFI/O: Refresh Input/Output - In the auto-modes this pin is the Refresh Request Output. It goes low following RFCK

Pin Definitions (Continued)

indicating that no hidden refresh was performed while RFCK was high. When this pin is set low by an external gate the on-chip refresh counter is reset to all zeroes.

$\overline{WIN}$: Write Enable Input.

$\overline{WE}$: Write Enable Output - $\overline{WE}$ follows $\overline{WIN}$ unconditionally.

RAHS: Row Address Hold Time Select - Selects the t_{RAH} to be generated by the DP8419 delay line to allow use with fast or slow DRAMs.

$\overline{CAS}$: Column Address Strobe Output - In mode 5 and in mode 4 with $\overline{CASIN}$ low before R/$\overline{C}$ goes low, $\overline{CAS}$ goes low automatically after the column address is valid on the address outputs. In mode 4 $\overline{CAS}$ follows $\overline{CASIN}$ directly after R/$\overline{C}$ goes low, allowing for nibble accessing. $\overline{CAS}$ is always high during refresh.

$\overline{RAS}$ 0–3: Row Address Strobe Outputs - The enabled $\overline{RAS}$ output (see Table II) follows $\overline{RASIN}$ directly during an access. During refresh, all $\overline{RAS}$ outputs are enabled.

B0, B1: Bank Select Inputs - These pins are decoded to enable one of the four $\overline{RAS}$ outputs during an access (see Table I and Table II).

TABLE I. DP8417, DP8419, DP8419X Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{RAS_n}$
B1	B0	
0	0	$\overline{RAS_0}$
0	1	$\overline{RAS_1}$
1	0	$\overline{RAS_2}$
1	1	$\overline{RAS_3}$

TABLE II. DP8418 Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{RAS_n}$
B1	NC	
0	X	$\overline{RAS_0}$ and $\overline{RAS_1}$
1	X	$\overline{RAS_2}$ and $\overline{RAS_3}$

Conditions for All Modes

INPUT ADDRESSING

The address block consists of a row-address latch, a column-address latch, and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses until after $\overline{CAS}$ goes low at the end of the memory cycle, ADS can be permanently high. Otherwise ADS must go low while the addresses are still valid.

DRIVE CAPABILITY

The DP8419 has timing parameters that are specified driving the typical capacitance (including traces) of 88, 5V-only DRAMs. Since there are 4 $\overline{RAS}$ outputs, each is specified driving one-fourth of the total memory. $\overline{CAS}$, $\overline{WE}$ and the address outputs are specified driving all 88 DRAMs.

The graph in *Figure 10* may be used to determine the slight variations in timing parameters, due to loading conditions other than 88 DRAMs.

Because of distributed trace capacitance and inductance and DRAM input capacitance, current spikes can be created, causing overshoots and undershoots at the DRAM inputs that can change the contents of the DRAMs or even destroy them. To reduce these spikes, a damping resistor (low inductance, carbon) should be inserted between the DP8419 outputs and the DRAMs, as close as possible to the DP8419. The damping resistor values may differ depending on how heavily an output is loaded. These resistors should be determined by the first prototypes (not wire-wrapped due to the larger distributed capacitance and inductance). Resistors should be chosen such that the transition on the control outputs is critically damped. Typical values will be from 15Ω to 100Ω, with the lower values being used with the larger memory arrays. Note that AC parameters are specified with 15Ω damping resistors. For more information see AN-305 "Precautions to Take When Driving Memories".

DP8419 DRIVING ANY 16k, 64k or 256k DRAMs

The DP8419 can drive any 16k, 64k or 256k DRAMs. All 16k DRAMs use basically the same configuration, including the 5V-only version. Hence, in most applications, different manufacturers' DRAMs are interchangeable (for the same supply-rail chips), and the DP8419 can drive them all (see *Figure 1a*).

There are three basic configurations for the 5V-only 64k DRAMs: a 128-row by 512-column array with an on-RAM refresh counter, a 128-row by 512-column array with no on-RAM refresh counter, and a 256-row by 256-column array with no on-RAM refresh counter. The DP8419 can drive all three configurations, and allows them all to be interchangeable (as shown in *Figures 1b* and *1c*), providing maximum flexibility in the choice of DRAMs. Since the 9-bit on-chip refresh counter can be used as a 7-bit refresh counter for the 128-row configuration, or as an 8-bit refresh counter for the 256-row configuration, the on-RAM refresh counter, if present, is never used.

256k DRAMs require all 18 of the DP8419's address inputs to select one memory location within the DRAM. $\overline{RAS}$-only refreshing with the nine-bit refresh-counter on the DP8419 makes $\overline{CAS}$ before $\overline{RAS}$ refreshing, available on 256k DRAMs, unnecessary.

READ, WRITE AND READ-MODIFY-WRITE CYCLES

The output signal, $\overline{WE}$, determines what type of memory access cycle the memory will perform. If $\overline{WE}$ is kept high while $\overline{CAS}$ goes low, a read cycle occurs. If $\overline{WE}$ goes low before $\overline{CAS}$ goes low, a write cycle occurs and data at DI (DRAM input data) is written into the DRAM as $\overline{CAS}$ goes low. If $\overline{WE}$ goes low later than t_{CWD} after $\overline{CAS}$ goes low, first a read occurs and DO (DRAM output data) becomes valid, then data DI is written into the same address in the DRAM as $\overline{WE}$ goes low. In this read-modify-write case, DI and DO cannot be linked together. $\overline{WE}$ always follows $\overline{WIN}$ directly to determine the type of access to be performed.

POWER-UP INITIALIZE

When V_{CC} is first applied to the DP8419, an initialize pulse clears the refresh counter and the internal control flip-flops.

Mode Features Summary

- 4 modes of operation: 2 access and 2 refresh
- Automatic or external control selected by the user
- Auto access mode provides $\overline{RAS}$, row to column change, and then $\overline{CAS}$ automatically
- Choice between two different values of t_{RAH} in auto-access mode
- $\overline{CAS}$ controlled independently in external control mode, allowing for nibble mode accessing
- Automatic refreshing can make refreshes transparent to the system
- $\overline{CAS}$ is inhibited during refresh cycles

DP8419 Mode Descriptions

MODE 0–EXTERNALLY CONTROLLED REFRESH

Figure 2 shows the Externally Controlled Refresh timing. In this mode the refresh counter contents are multiplexed to the address outputs. All $\overline{RAS}$ outputs are enabled to follow $\overline{RASIN}$ so that the row address indicated by the refresh counter is refreshed in all DRAM banks when $\overline{RASIN}$ goes low. The refresh counter increments when $\overline{RASIN}$ goes high. $\overline{RFSH}$ should be held low at least until $\overline{RASIN}$ goes high (they may go high simultaneously) so that the refresh address remains valid and all $\overline{RAS}$ outputs remain enabled throughout the refresh.

A burst refresh may be performed by holding $\overline{RFSH}$ low and toggling $\overline{RASIN}$ until all rows are refreshed. It may be useful in this case to reset the refresh counter just prior to beginning the refresh. The refresh counter resets to all zeroes when RFI/O is pulled low by an external gate. The refresh counter always counts to 511 before rolling over to zero. If there are 128 or 256 rows being refreshed then Q7 or Q8, respectively, going high may be used as an end-of-burst indicator.

In order that the refresh address is valid on the address outputs prior to the $\overline{RAS}$ lines going low, $\overline{RFSH}$ must go low before $\overline{RASIN}$. The setup time required is given by t_{RFLRL} in the Switching Characteristics. This parameter may be adjusted using *Figure 10* for loading conditions other than those specified.

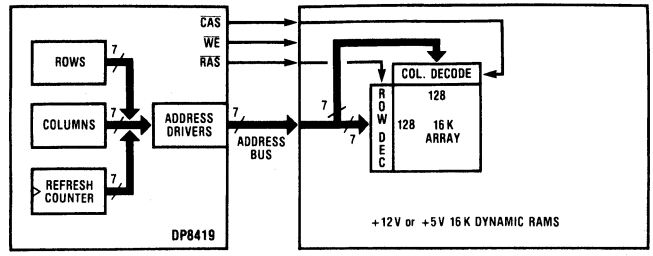
TABLE III. DP8419 Mode Select Options

Mode	($\overline{RFSH}$) M2	M0	Mode of Operation
0	0	0	Externally Controlled Refresh
1	0	1	Auto Refresh–Forced
4	1	0	Externally Controlled Access
5	1	1	Auto Access (Hidden Refresh)

7

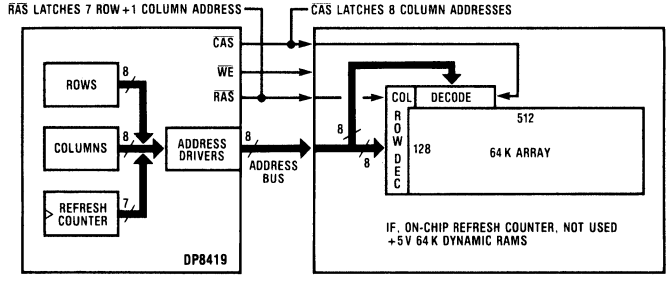
DP8419 Mode Descriptions (Continued)

DP8419 Interface Between System & DRAM Banks



TL/F/8396–5

FIGURE 1a. DP8419 with any 16k DRAMS

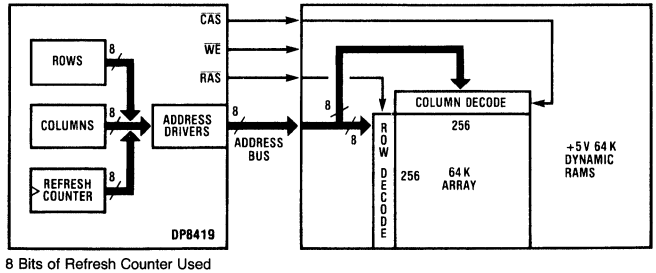


TL/F/8396–6

Only LS 7 Bits of Refresh Counter used for the 7 Row Addresses.

MSB not used but can toggle.

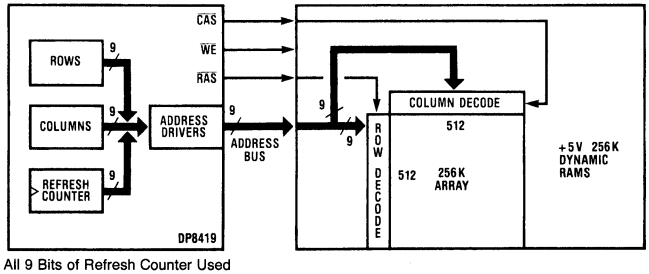
FIGURE 1b. DP8419 with 128 Row x 512 Column 64k DRAM



TL/F/8396–7

8 Bits of Refresh Counter Used

FIGURE 1c. DP8419 with 256 Row x 256 Column 64k DRAM

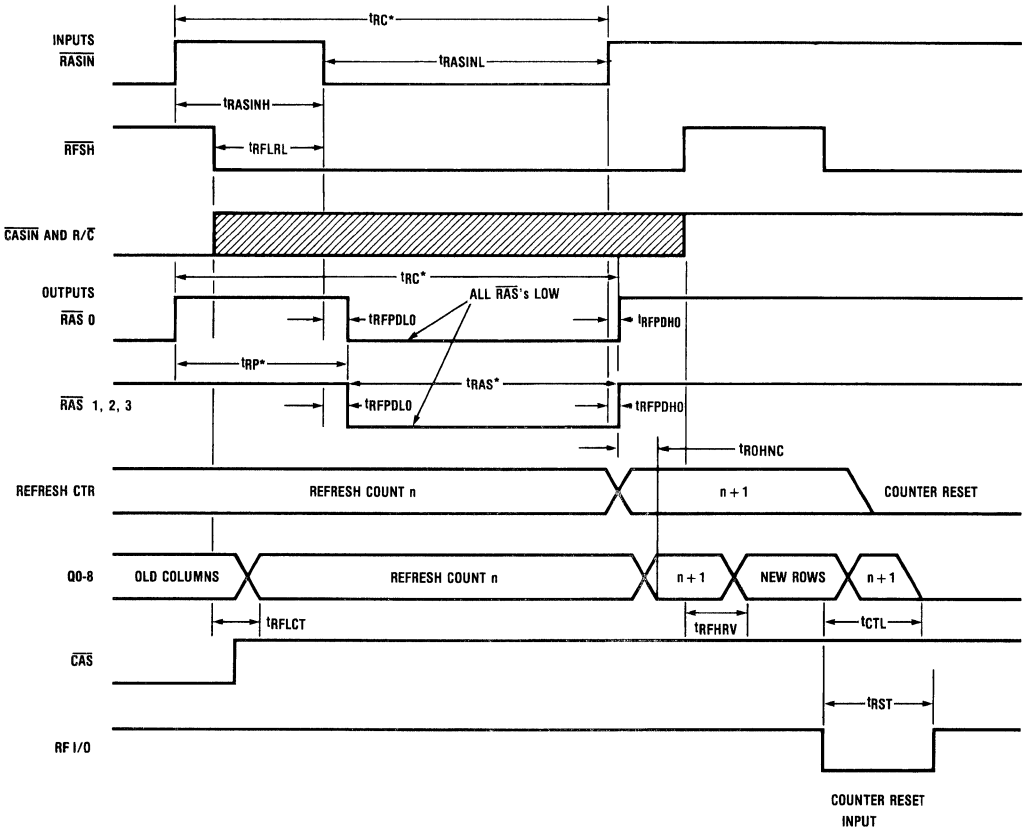


TL/F/8396–8

All 9 Bits of Refresh Counter Used

FIGURE 1d. DP8419 with 256k DRAMs

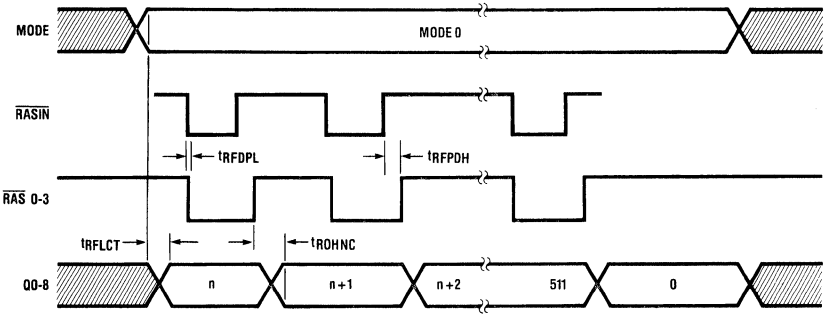
DP8419 Mode Descriptions (Continued)



TL/F/8396–9

*Indicates Dynamic RAM Parameters

FIGURE 2a. External Control Refresh Cycle (Mode 0)



TL/F/8396–10

FIGURE 2b. Burst Refresh Mode 0

DP8419 Mode Descriptions (Continued)

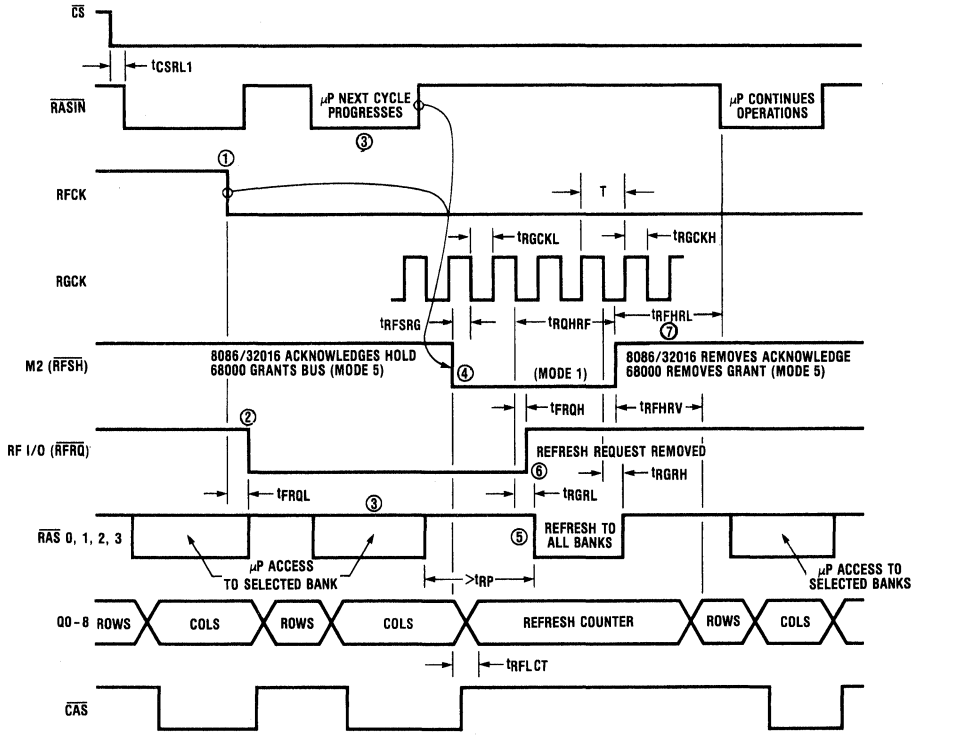
MODE 1–AUTOMATIC FORCED REFRESH

In Mode 1 the R/$\overline{C}$ (RFCK) pin becomes RFCK (refresh cycle clock) and the $\overline{CASIN}$ (RGCK) pin becomes RGCK ($\overline{RAS}$ generator clock). If RFCK is high and Mode 1 is entered then the chip operates as if in MODE 0 (externally controlled refresh), with all $\overline{RAS}$ outputs following $\overline{RASIN}$. This feature of Mode 1 may be useful for those who want to use Mode 5 (automatic access) with externally controlled refresh. By holding RFCK permanently high one need only toggle M2 ($\overline{RFSH}$) to switch from Mode 5 to external refresh. As with Mode 0, RFI/O may be pulled low by an external gate to reset the refresh counter.

When using Mode 1 as automatic refresh, RFCK must be an input clock signal. One refresh should occur each period of RFCK. If no refresh is performed while RFCK is high, then when RFCK goes low RFI/O immediately goes low to indicate that a refresh is requested. (RFI/O may still be used to reset the refresh counter even though it is also used as a refresh request pin, however, an open-collector gate should be used to reset the counter in this case since RFI/O is forced low internally for a request).

After receiving the refresh request the system must allow a forced refresh to take place while RFCK is low. External logic can monitor $\overline{RFRQ}$ (RFI/O) so that when $\overline{RFRQ}$ goes low this logic will wait for the access currently in progress to be completed before pulling M2 ($\overline{RFSH}$) low to put the DP8419 in mode 1. If no access is taking place when $\overline{RFRQ}$ occurs, then M2 may immediately go low. Once M2 is low, the refresh counter contents appear at the address outputs and $\overline{RAS}$ is generated to perform the refresh.

An external clock on RGCK is required to derive the refresh $\overline{RAS}$ signals. On the second falling edge of RGCK after M2 is low, all $\overline{RAS}$ lines go low. They remain low until two more falling edges of RGCK. Thus $\overline{RAS}$ remains high for one to two periods of RGCK after M2 goes low, and stays low for two periods. In order to obtain the minimum delay from M2 going low to $\overline{RAS}$ going low, M2 should go low t_{RFSRG} before the falling edge of RGCK.



TL/F/8396–11

① RFCK goes low
② $\overline{RFRQ}$ goes low if no hidden refresh occurred while RFCK was high
③ Next $\overline{RASIN}$ starts next access
④ µP acknowledges refresh request
⑤ Forced refresh $\overline{RAS}$ starts after > T (> t_{RP})
⑥ Forced refresh $\overline{RAS}$ ends $\overline{RFRQ}$
⑦ µP removes refresh acknowledge

FIGURE 3. DP8419 Performing a Forced Refresh (Mode 5 → 1 → 5) with Various Microprocessors

DP8419 Mode Descriptions (Continued)

The Refresh Request on RFI/O is terminated as $\overline{RAS}$ goes low. This signal may be used to end the refresh earlier than it normally would as described above. If M2 is pulled high while the $\overline{RAS}$ lines are low, then the $\overline{RAS}$s go high t_{RFRH} later. The designer must be careful, however, not to violate the minimum $\overline{RAS}$ low time of the DRAMs. He must also guarantee that the minimum $\overline{RAS}$ precharge time is not violated during a transition from mode 1 to mode 5 when an access is desired immediately following a refresh.

If the processor tries to access memory while the DP8419 is in mode 1, WAIT states should be inserted into the processor cycles until the DP8419 is back in mode 5 and the desired access has been accomplished (see *Figure 9*).

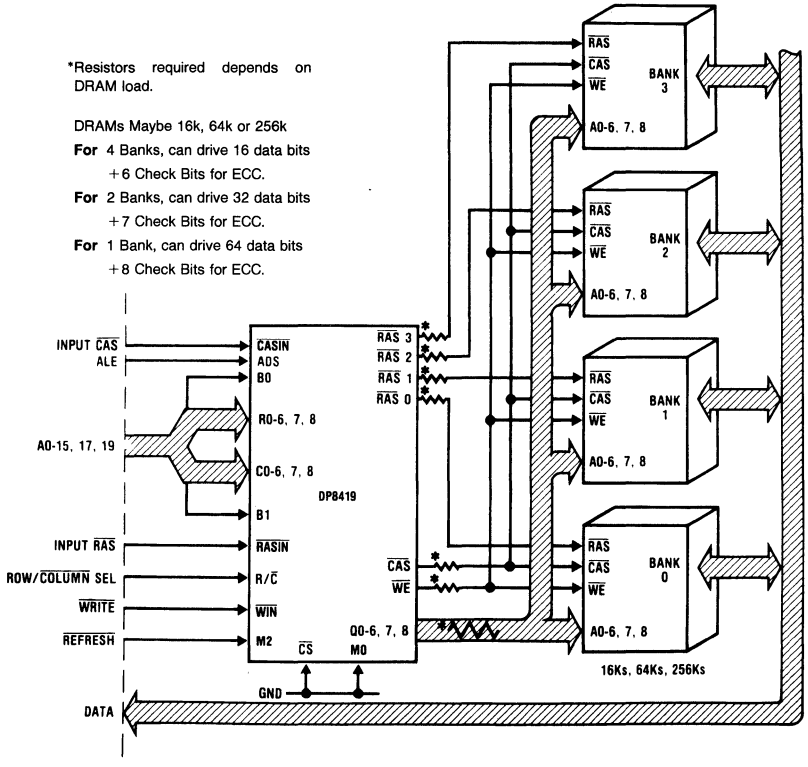
Instead of using WAIT states to delay accesses when refreshing, HOLD states could be used as follows. $\overline{RFRQ}$ could be connected to a HOLD or Bus Request input to the system. When convenient, the system acknowledges the $\overline{HOLD}$ or $\overline{Bus\ Request}$ by pulling M2 low. Using this scheme, $\overline{HOLD}$ will end as the $\overline{RAS}$ lines go low (RFI/O goes high). Thus, there must be sufficient delay from the time $\overline{HOLD}$ goes high to the DP8419 returning to mode 5, so that the $\overline{RAS}$ low time of the DRAMs isn't violated as described earlier (see *Figure 3* for mode 1 refresh with Hold states).

To perform a forced refresh the system will be inactive for about four periods of RGCK. For a frequency of 10 MHz, this is 400 ns. To refresh 128 rows every 2 ms an average of about one refresh per 16 μs is required. With a RFCK period of 16 μs and RGCK period of 100 ns, DRAM accesses are delayed due to refresh only 2.5% of the time. If using the Hidden Refresh available in mode 5 (refreshing with RFCK high) this percentage will be even lower.

MODE 4 - EXTERNALLY CONTROLLED ACCESS

In this mode all control signal outputs can be controlled directly by the corresponding control input. The enabled $\overline{RAS}$ output follows $\overline{RASIN}$, $\overline{CAS}$ follows $\overline{CASIN}$ (with R/$\overline{C}$ low), $\overline{WE}$ follows $\overline{WIN}$ and R/$\overline{C}$ determines whether the row or the column inputs are enabled to the address outputs (see *Figure 4*).

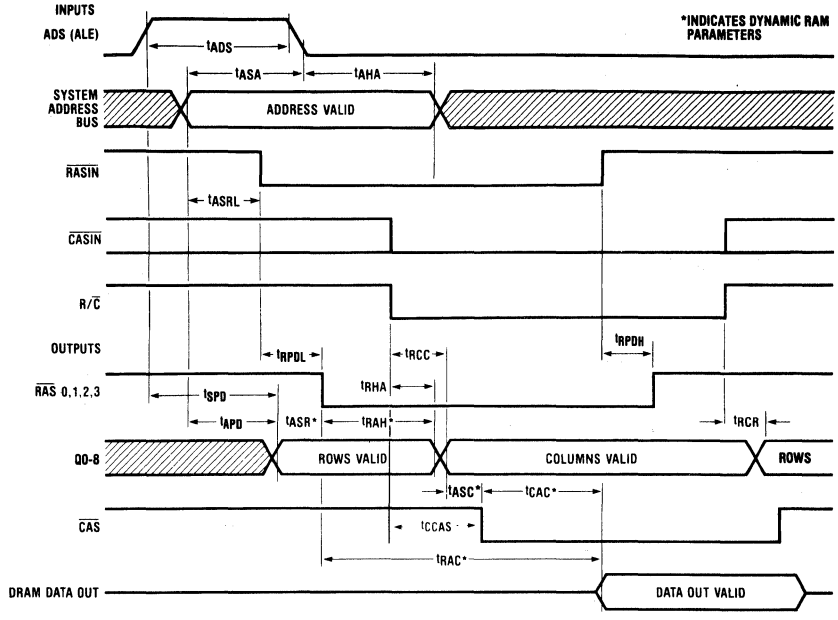
With R/$\overline{C}$ high, the row address latch contents are enabled onto the address bus. $\overline{RAS}$ going low strobes the row address into the DRAMs. After waiting to allow for sufficient row-address hold time (t_{RAH}) after $\overline{RAS}$ goes low, R/$\overline{C}$ can go low to enable the column address latch contents onto the address bus. When the column address is valid, $\overline{CAS}$ going low will strobe it into the DRAMs. $\overline{WIN}$ determines whether the cycle is a read, write or read-modify-write access. Refer to *Figures 5a* and *5b* for typical Read and Write timing using mode 4.



TL/F/8396–12

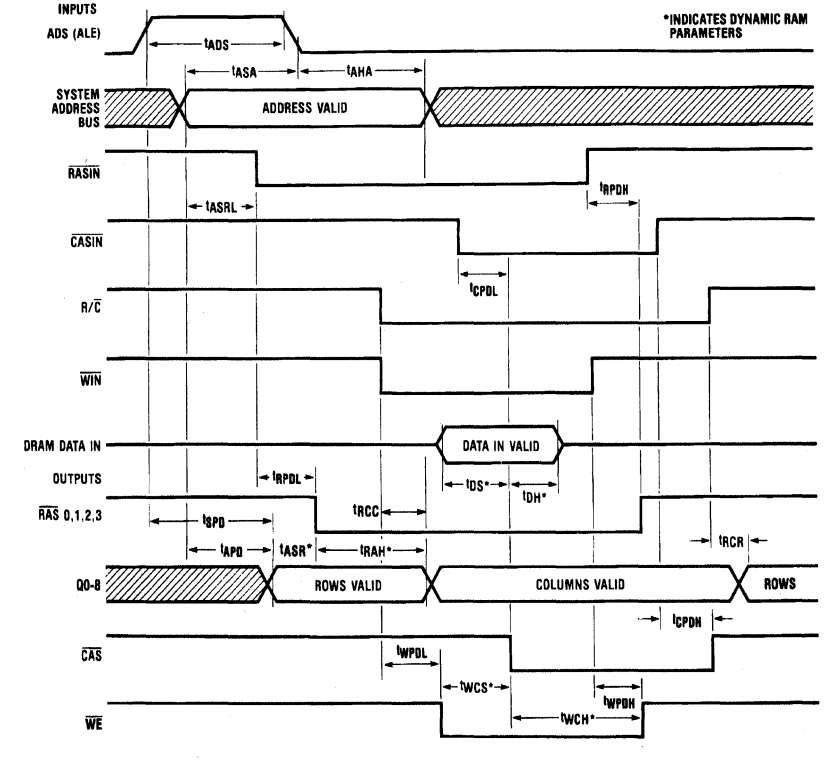
FIGURE 4. Typical Application of DP8419 Using External Control Access and Refresh in Modes 0 and 4

DP8419 Mode Descriptions (Continued)



TL/F/8396–13

FIGURE 5a. Read Cycle Timing (Mode 4)



TL/F/8396–14

FIGURE 5b. Write Cycle Timing (Mode 4)

DP8419 Mode Descriptions (Continued)

Page or Nibble mode may be performed by toggling $\overline{CASIN}$ once the initial access has been completed. In the case of page mode the column address must be changed before $\overline{CASIN}$ goes low to access a new memory location (see *Figure 5c*). Parameter t_{CPdif} has been specified in order that users may easily determine minimum $\overline{CAS}$ pulse widths when $\overline{CASIN}$ is toggling.

AUTOMATIC $\overline{CAS}$ GENERATION

$\overline{CAS}$ is held high when R/$\overline{C}$ is high even if $\overline{CASIN}$ is low. If $\overline{CASIN}$ is low when R/$\overline{C}$ goes low, $\overline{CAS}$ goes low automatically, t_{ASC} after the column address is valid. This feature eliminates the need for an externally derived $\overline{CASIN}$ signal to control $\overline{CAS}$ when performing a simple access (*Figure 5a* demonstrates Auto-$\overline{CAS}$ generation in mode 4). Page or nibble accessing may be performed as shown in *Figure 5c* even if $\overline{CAS}$ is generated automatically for the initial access.

FASTEST MEMORY ACCESS

The fastest mode 4 access is achieved by using the automatic $\overline{CAS}$ feature and external delay line to generate the required delay between $\overline{RASIN}$ and R/$\overline{C}$. The amount of delay required depends on the minimum t_{RAH} of the DRAMs being used. The DP8419 parameter t_{DIF1} has been specified in order that the delay between $\overline{RASIN}$ and R/$\overline{C}$ may be minimized.

$$t_{DIF1} = \text{MAXIMUM } (t_{RPDL} - t_{RHA})$$

where t_{RPDL} = $\overline{RASIN}$ to $\overline{RAS}$ delay

and t_{RHA} = row address held from R/$\overline{C}$ going low.

The delay between $\overline{RASIN}$ and R/$\overline{C}$ that guarantees the specified DRAM t_{RAH} is given by

$$\text{MINIMUM } \overline{RASIN} \text{ to R/}\overline{C} = t_{DIF1} + t_{RAH}.$$

Example

In an application using DRAMs that require a minimum t_{RAH} of 15 ns, the following demonstrates how the maximum $\overline{RASIN}$ to $\overline{CAS}$ time is determined.

With t_{DIF1} (from Switching Characteristics) = 7 ns,

$\overline{RASIN}$ to R/$\overline{C}$ delay = 7 ns + 15 ns = 22 ns.

A delay line of 25 ns will be sufficient.

With Auto-$\overline{CAS}$ generation, the maximum delay from R/$\overline{C}$ to $\overline{CAS}$ (loaded with 600 pF) is 46 ns. Thus the maximum $\overline{RASIN}$ to $\overline{CAS}$ time is 71 ns, under the given conditions.

With a maximum $\overline{RASIN}$ to $\overline{RAS}$ time (t_{RPDL}) of 20 ns, the maximum $\overline{RAS}$ to $\overline{CAS}$ time is about 51 ns. Most DRAMs with a 15 ns minimum t_{RAH} have a maximum t_{RCD} of about 60 ns. Thus, memory accesses are likely to be $\overline{RAS}$ limited instead of $\overline{CAS}$ limited. In other words, memory access time is limited by DRAM performance, not controller performance.

REFRESHING IN CONJUNCTION WITH MODE 4

If using mode 4 to access memory, mode 0 (externally controlled refresh) must be used for all refreshing.

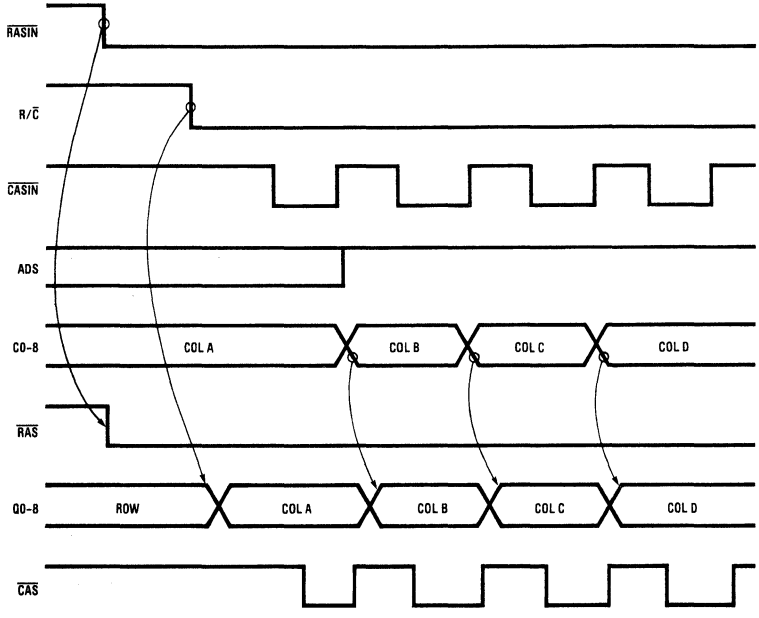
MODE 5 – AUTOMATIC ACCESS WITH HIDDEN REFRESHING CAPABILITY

Automatic-Access has two advantages over the externally controlled access (mode 4). First, $\overline{RAS}$, $\overline{CAS}$ and the row to column change are all derived internally from one input signal, $\overline{RASIN}$. Thus the need for an external delay line (see mode 4) is eliminated.

Secondly, since R/$\overline{C}$ and $\overline{CASIN}$ are not needed to generate the row to column change and $\overline{CAS}$, these pins can be used for the automatic refreshing function.

AUTOMATIC ACCESS CONTROL

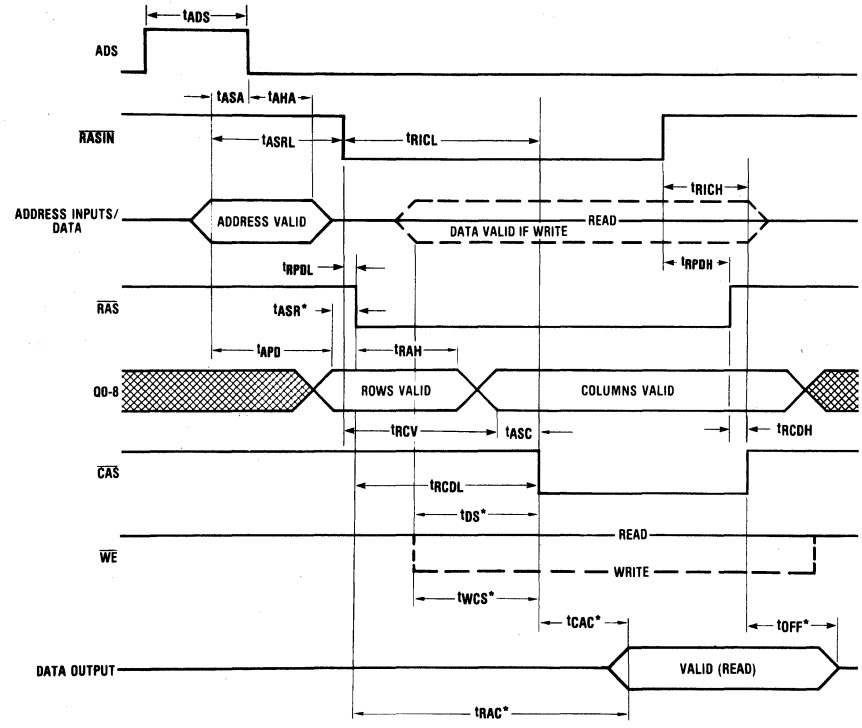
Mode 5 of the DP8419 makes accessing Dynamic RAM nearly as easy as accessing static RAM. Once row and column addresses are valid (latched on the DP8419 if necessary), $\overline{RASIN}$ going low is all that is required to perform the memory access.



TL/F/8396–15

FIGURE 5c. Page or Nibble Access in Mode 4

DP8419 Mode Descriptions (Continued)



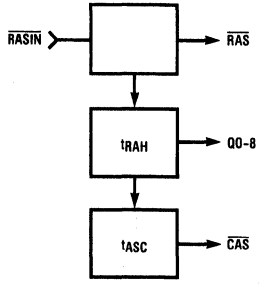
*Indicates Dynamic RAM Parameters

TL/F/8396–17

FIGURE 6. Mode 5 Timing

(Refer to *Figure 6*) In mode 5 the selected $\overline{RAS}$ follows $\overline{RASIN}$ immediately, as in mode 4, to strobe the row address into the DRAMs. The row address remains valid on the DP8419 address outputs long enough to meet the t_{RAH} requirement of the DRAMs (pin 4, RAHS, of the DP8419 allows the user two choices of t_{RAH}). Next, the column address replaces the row address on the address outputs and $\overline{CAS}$ goes low to strobe the columns into the DRAMs. $\overline{WIN}$ determines whether a read, write or read-modify-write is done.

The diagram below illustrates mode 5 automatic control signal generation.



TL/F/8396–16

REFRESHING IN CONJUNCTION WITH MODE 5

When using mode 5 to perform memory accesses, refreshing may be accomplished:

(a) externally (in mode 0 or mode 1)

(b) by a combination of mode 5 (hidden refresh) and mode 1 (auto-refresh)

or (c) by a combination of mode 5 and mode 0

(a) Externally Controlled Refreshing in Mode 0 or Mode 1

All refreshing may be accomplished using external refreshes in either mode 0 or mode 1 with R/$\overline{C}$ (RFCK) tied high (see mode 0 and mode 1 descriptions). If this is desired, the system determines when a refresh will be performed, puts the DP8419 in the appropriate mode, and controls the $\overline{RAS}$ signals directly with $\overline{RASIN}$. The on-chip refresh counter is enabled to the address outputs of the DP8419 when the refresh mode is entered, and increments when $\overline{RASIN}$ goes high at the completion of the refresh.

(b) Mode 5 Refreshing (hidden) with Mode 1 refreshing (auto)

(Refer to *Figure 7a*) If RFCK is tied to a clock (see mode 1 description), RFI/O becomes a refresh request output and goes low following RFCK going low if no refresh occurred while RFCK was high. Refreshes may be performed in mode 5 when the DP8419 is not selected for access ($\overline{CS}$ is high) and RFCK is high. If these conditions exist the refresh counter contents appear on the DP8419 address outputs and all $\overline{RAS}$ lines follow $\overline{RASIN}$ so that if $\overline{RASIN}$ goes low (an access other than through the DP8419 occurs), all $\overline{RAS}$ lines go low to perform the refresh. The DP8419 allows only one refresh of this type for each period of RFCK, since RFCK should be fast enough such that one refresh per period is sufficient to meet the DRAM refresh requirement.

DP8419 Mode Descriptions (Continued)

Once it is started, a hidden refresh will continue even if RFCK goes low. However, $\overline{CS}$ must be high throughout the refresh (until $\overline{RASIN}$ goes high).

These hidden refreshes are valuable in that they do not delay accesses. When determining the duty cycle of RFCK, the high time should be maximized in order to maximize the probability of hidden refreshes. If a hidden refresh doesn't happen, then a refresh request will occur on RFI/O when RFCK goes low. After receiving the request, the system must perform a refresh while RFCK is low. This may be done by going to mode 1 and allowing an automatic refresh (see mode 1 description). This refresh must be completed while RFCK is low, thus the RFCK low time is determined by the worst-case time required by the system to respond to a refresh request.

(c) Mode 5 Refresh (Hidden Refresh) with mode 0 Refresh (External Refresh)

This refresh scheme is identical to that in (b) except that after receiving a refresh request, mode 0 is entered to do the refresh (see mode 0 description). The refresh request is terminated (RFI/O goes high) as soon as mode 0 is entered. This method requires more control than using mode 1 (auto-refresh), however, it may be desirable if the mode 1 refresh time is considered to be excessive.

Example

Figure 7b demonstrates how a system designer would use the DP8419 in mode 5 based on certain characteristics of his system.

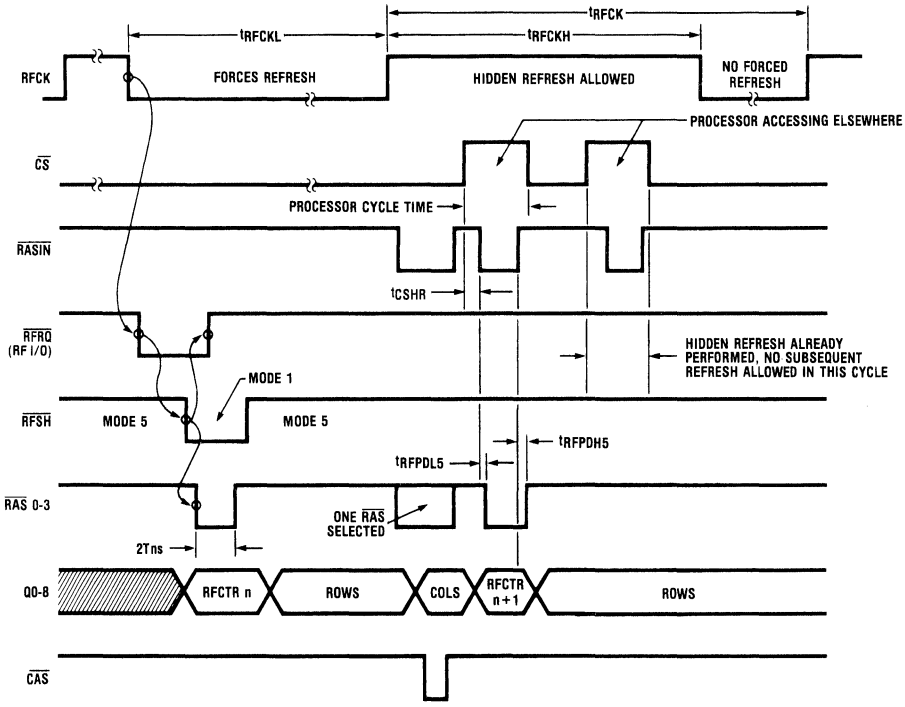
System Characteristics:

1) DRAM used has min t_{RAH} requirement of 15 ns and min t_{ASR} of 0 ns
2) DRAM address is valid from time T_V to the end of the memory cycle
3) four banks of twenty-two 256K memory chips each are being driven

Using the DP8419 (see *Figure 7b*):

1) Tie pin 4 (RAHS) high to guarantee a 15 ns minimum t_{RAH} which is sufficient for the DRAMs being used
2) Generate $\overline{RASIN}$ no earlier than time $T_V + t_{ASRL}$ (see switching characteristics), so that the row address is valid on the DRAM address inputs before $\overline{RAS}$ occurs
3) Tie ADS high since latching the DRAM address on the DP8419 is not necessary
4) Connect the first 18 system address bits to R0-R8 and C0-C8, and bits 19 and 20 to B0 and B1
5) Connect each $\overline{RAS}$ output of the DP8419 to the $\overline{RAS}$ inputs of the DRAMs of one bank of the memory array; connect Q0-Q8 of the DP8419 to A0-A8 of all DRAMs; connect $\overline{CAS}$ of the DP8419 to $\overline{CAS}$ of all the DRAMs

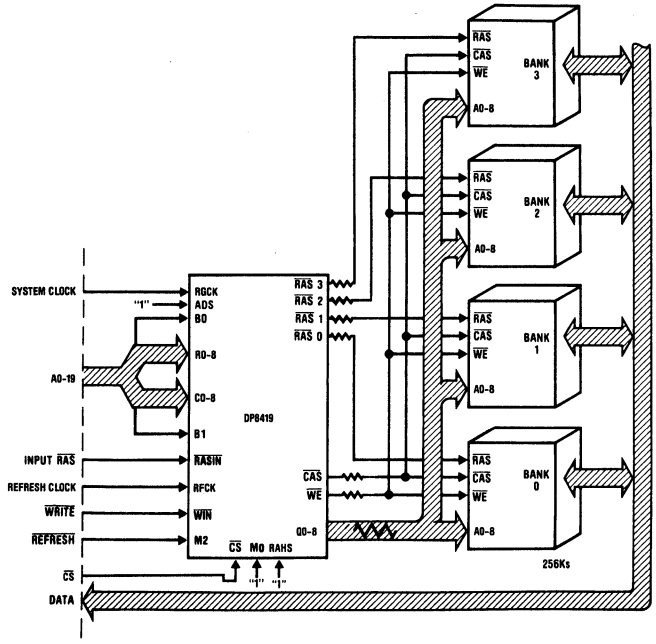
Figure 7c illustrates a similar example using the DP8418 to drive two 32-bit banks.



TL/F/8396–18

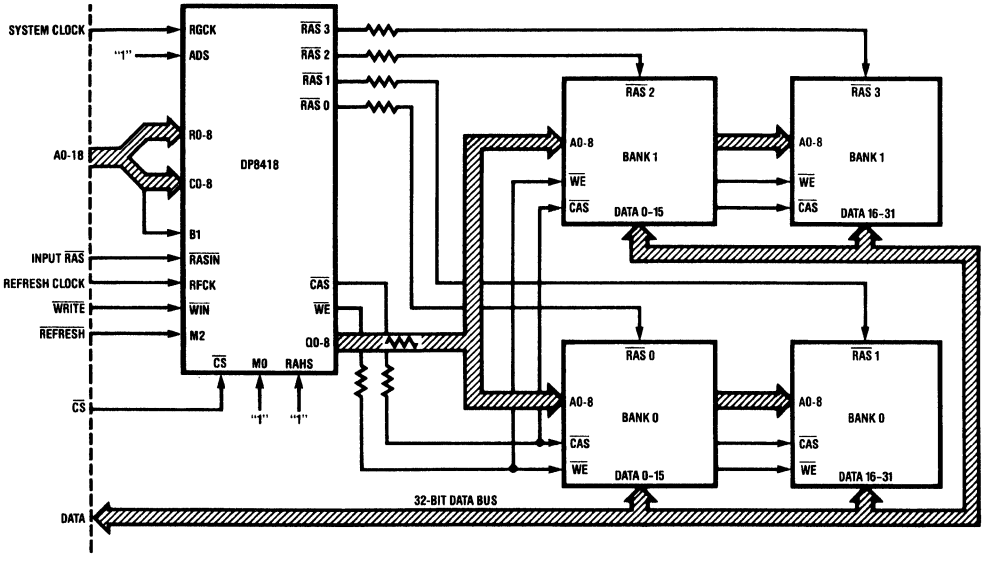
FIGURE 7a. Hidden Refreshing (Mode 5) and Forced Refreshing (Mode 1) Timing

DP8419 Mode Descriptions (Continued)



TL/F/8396–19

FIGURE 7b. Typical Application of DP8419 Using Modes 5 and 1



TL/F/8396–33

FIGURE 7c. Typical Application of DP8418 Using Modes 5 and 1

Applications

If one desires a memory interface containing the DP8419 that minimizes the number of external components required, modes 5 and 1 should be used. These two modes provide:

1) Automatic access to memory (in mode 5 only one signal, $\overline{RASIN}$, is required in order to access memory)
2) Hidden refresh capability (refreshes are performed automatically while in mode 5 when non-local accesses are taking place, as determined by $\overline{CS}$)
3) Refresh request capability (if no hidden refresh took place while RFCK was high, a refresh request is generated at the RFI/O pin when RFCK goes high)
4) Automatic forced refresh (If a refresh request is generated while in mode 5, as described above, external logic should switch the DP8419 into mode 1 to do an automatic forced refresh. No other external control signals need be issued. WAIT states can be inserted into the processor machine cycles if the system tries to access memory while the DP8419 is in mode 1 doing a forced refresh).

Some items to be considered when integrating the DP8419 into a system design are:

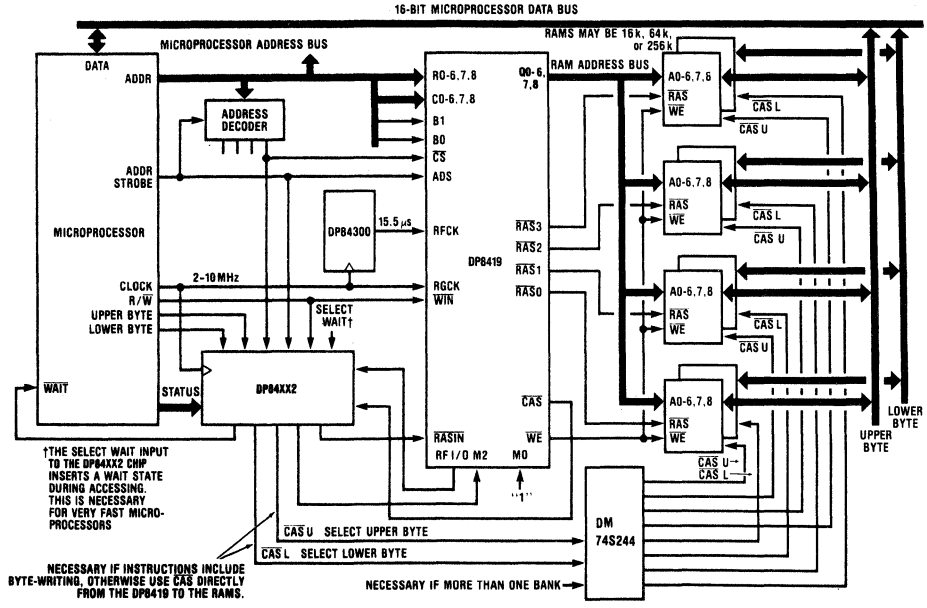
1) The system designer should ensure that a DRAM access not be in progress when a refresh mode is entered. Similarly, one should not attempt to start an access while a refresh is in progress. The parameter t_{RFHRL} specifies the minimum time from $\overline{RFSH}$ high to $\overline{RASIN}$ going low to initiate an access.
2) One should always guarantee that the DP8419 is enabled for access prior to initiating the access (see t_{CSRL1}).
3) One should bring $\overline{RASIN}$ low even during non-local access cycles when in mode 5 in order to maximize the chance of a hidden refresh occurring.
4) At lower frequencies (under 10 Mhz), it becomes increasingly important to differentiate between READ and WRITE cycles. $\overline{RASIN}$ generation during READ cycles can take place as soon as one knows that a processor READ access cycle has started. WRITE cycles, on the other hand, cannot start until one knows that the data to be written at the DRAM inputs will be valid a setup time before $\overline{CAS}$ (column address strobe) goes true at the DRAM inputs. Therefore, in general, READ cycles can be initiated earlier than WRITE cycles.
5) Many times it is possible to only add WAIT states during READ cycles and have no WAIT states during WRITE cycles. This is because it generally takes less time to write data into memory than to read data from memory.

The DP84XX2 family of inexpensive preprogrammed medium Programmable Array Logic devices (PALs) have been developed to provide an easy interface between various microprocessors and the DP84XX family of DRAM controller/drivers. These PALs interface to all the necessary control signals of the particular processor and the DP8419. The PAL controls the operation of the DP8419 in modes 5 and 1, while meeting all the critical timing considerations discussed above. The refresh clock, RFCK, may be divided down from the processor clock using an IC counter such as the DM74LS393 or the DP84300 programmable refresh timer. The DP84300 can provide RFCK periods ranging from 15.4 μs to 15.6 μs based on an input clock of 2 to 10 MHz. *Figure 8* shows a general block diagram for a system using the DP8419 in modes 1 and 5. *Figure 9* shows possible timing diagrams for such a system (using $\overline{WAIT}$ to prohibit access when refreshing). Although the DP84XX2 PALs are offered as standard peripheral devices for the DP84XX DRAM controller/drivers, the programming equations for these devices are provided so the user may make minor modification, for unique system requirements.

ADVANTAGES OF DP8419 OVER A DISCRETE DYNAMIC RAM CONTROLLER

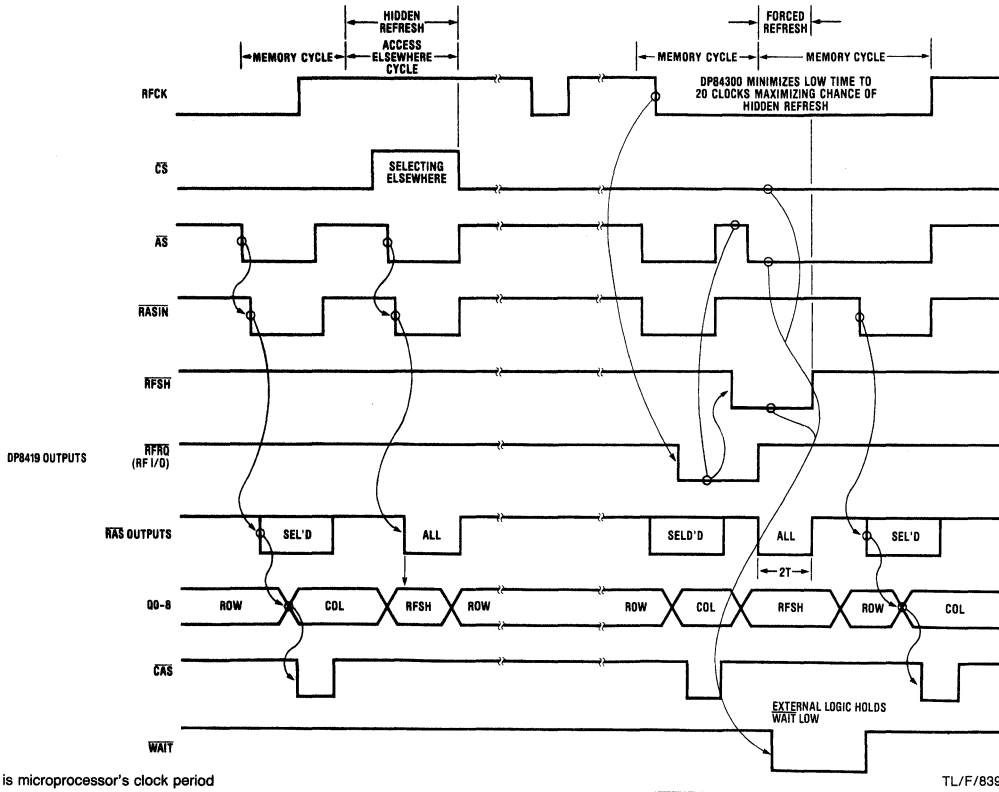
1) The DP8419 system solution takes up much less board space because everything is on one chip (latches, refresh counter, control logic, multiplexers, drivers, and internal delay lines).
2) Less effort is needed to design a memory system. The DP8419 has automatic modes (1 and 5) which require a minimum of external control logic. Also programmable array logic devices (PALs) have been designed which allow an easy interface to most popular microprocessors (Motorola 68000 family, National Semiconductor 32032 family, Intel 8086 family, and the Zilog Z8000 family).
3) Less skew in memory timing parameters because all critical components are on one chip (many discrete drivers specify a minimum on-chip skew under worst-case conditions, but this cannot be used if more then one driver is needed, such as would be the case in driving a large dynamic RAM array).
4) Our switching characteristics give the designer the critical timing specifications based on TTL output levels (low = 0.8V, high = 2.4V) at a specified load capacitance. All timing parameters are specified on the DP8419:
   A) driving 88 DRAM's over a temperature range of 0–70 degrees centigrade (no extra drivers are needed).
   B) under worst-case driving conditions with all outputs switching simultaneously (most discrete drivers on the market specify worst-case conditions with only one output switching at a time; this is not a true worst-case condition!).

Applications (Continued)



TL/F/8396–20

FIGURE 8. Connecting the DP8419 Between the 16-bit Microprocessor and Memory



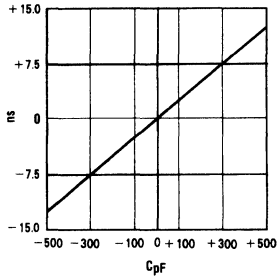
*T is microprocessor's clock period

TL/F/8396–21

FIGURE 9. DP8419 Auto Refresh, Access with $\overline{\text{WAIT}}$ States

Switching Characteristics

All AC parameters are specified with the equivalent load capacitances, including traces, of 88 DRAMs organized as 4 banks of 22 DRAMs each. Maximums are based on worst-case conditions including all outputs switching simultaneously. This, in many cases, results in the AC values shown in the DP84XX DRAM controller data sheet being much looser than true worst case (maximum) AC delays. The system designer should estimate the DP8419 load in his/her application, and modify the appropriate AC parameters using the graph in *Figure 10*. Two example calculations are provided below.



TL/F/8396–22

FIGURE 10. Change in Propagation Delay Relative to "True" (Application) Load Minus AC Specified Data Sheet Load

2 Examples

#1) A mode 4 user driving 2 16-bit banks of DRAM has the following approximate "true" loading conditions:

$\overline{CAS}$ - 300 pF

Q0-Q8 - 250 pF

$\overline{RAS}$ - 150 pF

max t_{RPDL} = 20 ns − 0 ns = 20 ns (since $\overline{RAS}$ loading is the same as that which is spec'ed)

max t_{CPDL} = 32 ns − 7 ns = 25 ns

max t_{CCAS} = 46 ns − 7 ns = 39 ns

max t_{RCC} = 41 ns − 6 ns = 35 ns

min t_{RHA} is not significantly effected since it does not involve an output transition

Other parameters are adjusted in a similar manner.

#2) A mode 5 user driving one 16-bit bank of DRAM has the following approximate "true" loading conditions:

$\overline{CAS}$ - 120 pF

Q0-Q8 - 100 pF

$\overline{RAS}$ - 120 pF

A. C. parameters should be adjusted as follows:

with RAHS = "1",

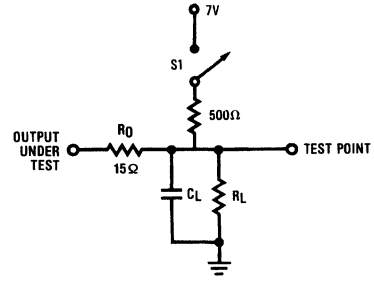
max t_{RICL} = 70 ns − 11 ns = 59 ns

max t_{RCDL} = 55 ns + 1 ns − 11 ns = 45 ns

(the + 1 ns is due to lighter $\overline{RAS}$ loading; the − 11 ns is due to lighter $\overline{CAS}$ loading)

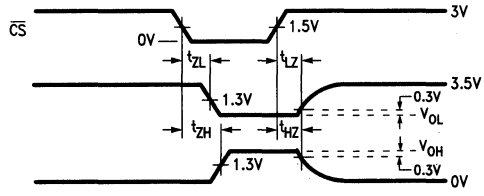
min t_{RAH} = 15 ns + 1 ns = 16 ns

The additional 1 ns is due to the fact that the $\overline{RAS}$ line is driving less (switching faster) than the load to which the 15 ns spec applies. The row address will remain valid for about the same time irregardless of address loading since it is considered to be not valid at the beginning of its transition.



TL/F/8396–23

FIGURE 11a. Output Load Circuit



TL/F/8396–34

FIGURE 11b. DP8417 TRI-STATE Waveforms

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply voltage, V_{CC}	7.0V
Storage Temperature Range	−65°C to +150°C
Input Voltage	5.5V
Output Current	150 mA
Lead Temp. (Soldering, 10 seconds)	300°C

Operating Conditions

		Min	Max	Units
V_{CC}	Supply Voltage	4.50	5.50	V
T_A	Ambient Temperature	0	+70	°C

Electrical Characteristics V_{CC} = 5.0V ±10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Note 2)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_C	Input Clamp Voltage	V_{CC} = Min, I_C = − 12 mA		− 0.8	− 1.2	V
I_{IH}	Input High Current for all Inputs	V_{IN} = 2.5V		2.0	100	μA
I_I RSI	Output Load Current for RFI/O	V_{IN} = 0.5V, Output high		−0.7	−1.5	mA
I_{IL1}	Input Low Current for all Inputs**	V_{IN} = 0.5V		−0.02	−0.25	mA
I_{IL2}	ADS, R/$\overline{C}$, $\overline{CS}$, M2, $\overline{RASIN}$	V_{IN} = 0.5V		−0.05	−0.5	mA
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_{OL1}	Output Low Voltage*	I_{OL} = 20 mA		0.3	0.5	V
V_{OL2}	Output Low Voltage for RFI/O	I_{OL} = 8 mA		0.3	0.5	V
V_{OH1}	Output High Voltage*	I_{OH} = − 1 mA	2.4	3.5		V
V_{OH2}	Output High Voltage for RFI/O	I_{OH} = − 100 μA	2.4	3.5		V
I_{1D}	Output High Drive Current*	V_{OUT} = 0.8V (Note 3)	−50	− 200		mA
I_{0D}	Output Low Drive Current*	V_{OUT} = 2.4V (Note 3)	50	200		mA
I_{CC}	Supply Current	V_{CC} = Max		150	240	mA

*Except RFI/O

**Except RFI/O, ADS, R/$\overline{C}$, $\overline{CS}$, M2, $\overline{RASIN}$

Switching Characteristics: DP8417, DP8418, DP8419, DP8419X

V_{CC} = 5.0V ±10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5), the output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11a* for test load. S1 is open unless otherwise noted. Maximum propagation delays are specified with all outputs switching.**

**** Preliminary**

Symbol	Parameter	Condition	*CL		**All C_L = 50 pF		Units
			Min	Max	Min	Max	
ACCESS							
t_{RICL0}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8417, 18, 19-80	57	97	42	85	ns
t_{RICL0}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8417, 18, 19-70	57	87	42	75	ns
t_{RICL1}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8417, 18, 19-80	48	80	35	68	ns
t_{RICL1}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8417, 18, 19-70	48	70	35	58	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ High Delay	*Figure 6*		37			ns
t_{RCDL0}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8417, 18, 19-80	43	80			ns
t_{RCDL0}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8417, 18, 19-70	43	72			ns
t_{RCDL1}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8417, 18, 19-80	34	63			ns
t_{RCDL1}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8417, 18, 19-70	34	55			ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ High Delay	*Figure 6*		22			ns
t_{RAH0}	Row Address Hold Time (RAHS = 0, Mode 5)	*Figure 6*	25		25		ns
t_{RAH1}	Row Address Hold Time (RAHS = 1, Mode 5)	*Figure 6*	15		15		ns
t_{ASC}	Column Address Set-up Time (Mode 5)	*Figure 6*	0		0		ns

Switching Characteristics: DP8417, DP8418, DP8419, DP8419X (Continued)

V_{CC} = 5.0V ±10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11a* for test load. S1 is open unless otherwise specified. Maximum propagation delays are specified with all outputs switching.**

**** Preliminary**

Symbol	Parameter	Condition	*CL		**All C_L = 50 pF		Units
			Min	Max	Min	Max	
ACCESS (Continued)							
t_{RCV0}	$\overline{RASIN}$ to Column Address Valid (RAHS = 0, Mode 5)	*Figure 6* DP8417, 18, 19-80		94			ns
t_{RCV0}	$\overline{RASIN}$ to Column Address Valid (RAHS = 0, Mode 5)	*Figure 6* DP8417, 18, 19-70		85			ns
t_{RCV1}	$\overline{RASIN}$ to Column Address Valid (RAHS = 1, Mode 5)	*Figure 6* DP8417, 18, 19-80		76			ns
t_{RCV1}	$\overline{RASIN}$ to Column Address Valid (RAHS = 1, Mode 5)	*Figure 6* DP8417, 18, 19-70		68			ns
t_{RPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Low Delay	*Figures 5a, 5b, 6*		21		18	ns
t_{RPDH}	$\overline{RASIN}$ to $\overline{RAS}$ High Delay	*Figures 5a, 5b, 6*		20		17	ns
t_{ASRL}	Address Set-up to $\overline{RASIN}$ low	*Figures 5a, 5b, 6*	13				ns
t_{APD}	Address Input to Output Delay	*Figures 5a, 5b, 6*		36		25	ns
t_{SPD}	Address Strobe High to Address Output Valid	*Figures 5a, 5b*		48			ns
t_{ASA}	Address Set-up Time to ADS	*Figures 5a, 5b, 6*	5				ns
t_{AHA}	Address Hold Time from ADS	*Figures 5a, 5b, 6*	10				ns
t_{ADS}	Address Strobe Pulse Width	*Figures 5a, 5b, 6*	26				ns
t_{WPD}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	*Figure 5b*		28			ns
t_{CPDL}	$\overline{CASIN}$ to $\overline{CAS}$ Low Delay (R/$\overline{C}$ low, Mode 4)	*Figure 5b*	21	32			ns
t_{CPDH}	$\overline{CASIN}$ to $\overline{CAS}$ High Delay (R/$\overline{C}$ low, Mode 4)	*Figure 5b*	16	33			ns
t_{CPdif}	t_{CPDL} - t_{CPDH}	See Mode 4 Description		11			ns
t_{RCC}	Column Select to Column Address Valid	*Figure 5a*		41			ns
t_{RCR}	Row Select to Row Address Valid	*Figures 5a, 5b*		45			ns
t_{RHA}	Row Address Held from Column Select	*Figure 5a*	7				ns
t_{CCAS}	R/$\overline{C}$ Low to $\overline{CAS}$ Low Delay ($\overline{CASIN}$ Low, Mode 4)	*Figure 5a* DP8417, 18, 19-80		50			ns
t	R/$\overline{C}$ Low to $\overline{CAS}$ Low Delay ($\overline{CASIN}$ Low, Mode 4)	*Figure 5a* DP8417, 18, 19-70		46			ns
t_{DIF1}	Maximum (t_{RPDL} - t_{RHA})	See Mode 4 Description		7			ns
t_{DIF2}	Maximum (t_{RCC} - t_{CPDL})			13			ns
REFRESH							
t_{RC}	Refresh Cycle Period	*Figure 2a*	100				ns
$t_{RASINL,H}$	Pulse Width of $\overline{RASIN}$ during Refresh	*Figure 2a*	50				ns
t_{RFPDL0}	$\overline{RASIN}$ to $\overline{RAS}$ Low Delay during Refresh (Mode 0)	*Figure 2a*		28			ns

Switching Characteristics: DP8417, DP8418, DP8419, DP8419X (Continued)

V_{CC} = 5.0V ±10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q8, C_L = 500 pF; $\overline{RAS}$0-$\overline{RAS}$3, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11a* for test load. S1 is open unless otherwise specified. Maximum propagation delays are specified with all outputs switching.**

Symbol	Parameter	Condition	*CL		All C_L = 50 pF		Units
			Min	Max	Min	Max	
REFRESH (Continued)							
t_{RFPDL5}	$\overline{RASIN}$ to $\overline{RAS}$ Low Delay during Hidden Refresh	*Figure 7*		38			ns
t_{RFPDH0}	$\overline{RASIN}$ to $\overline{RAS}$ High Delay during Refresh (Mode 0)	*Figure 2a*		35			ns
t_{RFPDH5}	$\overline{RASIN}$ to $\overline{RAS}$ High Delay during Hidden Refresh	*Figure 7*		44			ns
t_{RFLCT}	$\overline{RFSH}$ Low to Counter Address Valid	*Figures 2a, 3* $\overline{CS}$ = X		38			ns
t_{RFLRL}	$\overline{RFSH}$ Low Set-up to $\overline{RASIN}$ Low (Mode 0), to get Minimum t_{ASR} = 0	*Figure 2a*	12				ns
t_{RFHRL}	$\overline{RFSH}$ High Setup to Access $\overline{RASIN}$ Low	*Figure 3*	25				ns
t_{RFHRV}	$\overline{RFSH}$ High to Row Address Valid	*Figure 3*		43			ns
t_{ROHNC}	$\overline{RAS}$ High to New Count Valid	*Figure 2a*		42			ns
t_{RST}	Counter Reset Pulse Width	*Figure 2a*	46				ns
t_{CTL}	RFI/O Low to Counter Outputs All Low	*Figure 2a*		80			ns
$t_{RFCKL,H}$	Minimum Pulse Width of RFCK	*Figure 7*	100				ns
T	Period of $\overline{RAS}$ Generator Clock	*Figure 3*	30				ns
t_{RGCKL}	Minimum Pulse Width Low of RGCK	*Figure 3*	15				ns
t_{RGCKH}	Minimum Pulse Width High of RGCK	*Figure 3*	15				ns
t_{FRQL}	RFCK Low to Forced $\overline{RFRQ}$ (RFI/O) Low	*Figure 3* C_L = 50 pF RL = 35k		66			ns
t_{FRQH}	RGCK Low to Forced $\overline{RFRQ}$ High	*Figure 3* C_L = 50 pF RL = 35k		55			ns
t_{RGRL}	RGCK Low to $\overline{RAS}$ Low	*Figure 3*	21	41			ns
t_{RGRH}	RGCK Low to $\overline{RAS}$ High	*Figure 3*	23	48			ns

Switching Characteristics: DP8417, DP8418, DP8419, DP8419X (Continued)

V_{CC} = 5.0V ±10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q8, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11a* for test load. S1 is open unless otherwise specified. Maximum propagation delays are specified with all outputs switching.**

Symbol	Parameter	Condition	*CL		All C_L = 50 pF		Units
			Min	Max	Min	Max	
REFRESH (Continued)							
t_{RQHRF}	$\overline{RFSH}$ Hold Time from RGCK	*Figure 3*	2T				ns
t_{RFRH}	$\overline{RFSH}$ High to $\overline{RAS}$ High (Ending Forced Refresh early)	(See Mode 1 Description)		42			ns
t_{RFSRG}	$\overline{RFSH}$ Low Set-up to RGCK Low (Mode 1)	(See Mode 1 Description) *Figure 3*	12				ns
t_{CSHR}	$\overline{CS}$ High to $\overline{RASIN}$ Low for Hidden Refresh	*Figure 7*	10				ns
t_{RKRL}	RFCK High to $\overline{RASIN}$ low for hidden Refresh		50				ns
DP8419, DP8419X ONLY							
t_{CSRL1}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Mode 5 with Auto Refresh Mode)	*Figure 3*	34				ns
t_{CSRL0}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Modes 4 or 5 with externally controlled Refresh)	(See Mode 5 Description)	5				ns
DP8418 ONLY							
t_{CSRL1}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Mode 5 with Auto Refresh Mode)	*Figure 3*	5				ns
t_{CSRL0}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Modes 4 or 5 with externally controlled Refresh)	(See Mode 5 Description)	5				ns
DP8417 ONLY — PRELIMINARY							
t_{CSRL1}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Mode 5 with Auto Refresh Mode)	*Figure 3*	34				ns
t_{CSRL0}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Modes 4 or 5 with externally controlled Refresh)	(See Mode 5 Description)	34				ns
TRI-STATE							
t_{ZH}	$\overline{CS}$ Low to Output High from Hi-Z	S1 Open *Figure 11G*		50			ns
t_{HZ}	$\overline{CS}$ High to Output Hi-Z from High	S1 Open *Figure 11G*				50	ns
t_{ZL}	$\overline{CS}$ Low to Output Low from Hi-Z	S1 Closed *Figure 11G*		50			ns
t_{HZ}	$\overline{CS}$ High to Output Hi-Z from Low	S1 Closed *Figure 11G*				50	ns

Input Capacitance T_A = 25°C (Note 2)

Symbol	Parameter	Condition	Min	Typ	Max	Units
C_{IN}	Input Capacitance ADS, R/$\overline{C}$, $\overline{CS}$, M2, $\overline{RASIN}$			8		pF
C_{IN}	Input Capacitance All Other Inputs			5		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: This test is provided as a monitor of Driver output source and sink current capability. Caution should be exercised in testing this parameter. In testing these parameters, a 15Ω resistor should be placed in series with each output under test. One output should be tested at a time and test time should not exceed 1 second.

Note 4: Input pulse 0V to 3.0V, $t_R = t_F = 2.5$ ns, $f = 2.5$ MHz, $t_{PW} = 200$ ns. Input reference point on AC measurements is 1.5V Output reference points are 2.4V for High and 0.8V for Low.

Note 5: The load capacitance on RF I/O should not exceed 50 pF.

National Semiconductor Corporation

PRELIMINARY

DP84512 Dynamic RAM Controller Interface Circuit for the NS32332

General Description

This is a PAL (Programmable Array Logic) device designed to allow an easy interface between the National Semiconductor NS32332 microprocessor and the National Semiconductor DP8417/18/19/28/29 dynamic RAM controller.

This PAL supplies all the control signals needed to perform memory read, burst read, write, and refresh operations up to a frequency of 15 MHz.

Features

- Provides a 3-chip solution for the NS32332/DP8418 (or DP8428) dynamic RAM interface (1 PAL, DP8418 and clock divider)
- Works with all speed versions of the NS32332 up to 15 MHz
- Allows operation of NS32332 at 12 MHz with no WAIT states with standard 120 ns 256k or 1M DRAMs
- Controls DP8417/18/19/28/29 mode 5 accesses and mode 0 forced refreshes automatically
- Allows READ accesses in burst mode
- CPU WAIT states are automatically inserted during contention between DRAM accesses and DRAM refreshes
- Uses standard National Semiconductor PALs (i.e., DMPAL16R4A, the user may want to use faster versions of these PALs at higher CPU operating frequencies)
- The PAL programming equations are provided with comments for easy user modification to his specific requirements

National Semiconductor Corporation

DP8428/NS32828, DP8429/NS32829
1 Megabit High Speed Dynamic RAM Controller/Drivers

General Description

The DP8428 and DP8429 1M DRAM Controller/Drivers are designed to provide "No-Waitstate" CPU interface to Dynamic RAM arrays of up to 8 Mbytes and larger. The DP8428 and DP8429 are tailored for 32-bit and 16-bit system requirements, respectively. Both devices are fabricated using National's new oxide isolated Advanced Low power Schottky (ALS) process and use design techniques which enable them to significantly out-perform all other LSI or discrete alternatives in speed, level of integration, and power consumption.

Each device integrates the following critical 1M DRAM controller functions on a single monolithic device: ultra precise delay line; 9 bit refresh counter; fall-through row, column, and bank select input latches; Row/Column address muxing logic; on-board high capacitive-load $\overline{RAS}$, $\overline{CAS}$, Write Enable and Address output drivers; and, precise control signal timing for all the above.

In order to specify each device for "true" worst case operating conditions, all timing parameters are guaranteed while the chip is driving the capacitive load of 88 DRAMs including trace capacitance. The chip's delay timing logic makes use of a patented new delay line technique which keeps AC skew to ± 3 ns over the full V_{CC} range of $\pm 10\%$ and temperature range of $-55°C$ to $+125°C$. The DP8428 and DP8429 guarantee a maximum RASIN to CASOUT delay of 80 ns or 70 ns even while driving an 8 Mbyte memory array with error correction check bits included. Two speed selected options of these devices are shown in the switching characteristics section of this document. (Continued)

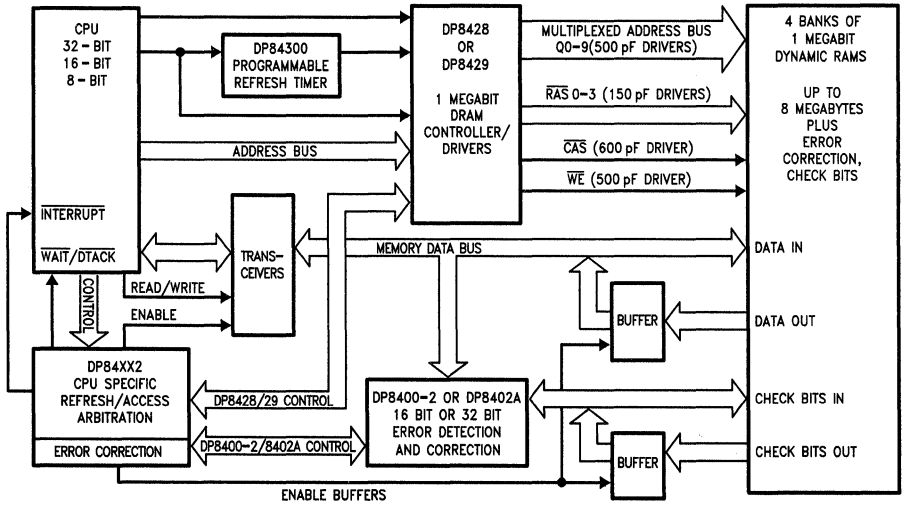
Features

- Makes DRAM interface and refresh tasks appear virtually transparent to the CPU making DRAMs as easy to use as static RAMs
- Specifically designed to eliminate CPU wait states up to 10 MHz or beyond
- Eliminates 20 discrete components for significant board real estate reduction, system power savings and the elimination of chip-to-chip AC skewing
- On-board ultra precise delay line
- On-board high capacitive $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$ and Address drivers (specified driving 88 DRAMs directly)
- AC specified for directly addressing up to 8 Mbytes
- Low power/high speed bipolar oxide isolated process
- Downward pin and function compatible with 256k DRAM Controller/Drivers DP8409A, DP8417, DP8418, and DP8419

Contents

- System and Device Block Diagrams
- Recommended Companion Components
- Device Connection Diagrams and Pin Definitions
- Device Differences—DP8428 vs DP8429
- Mode of Operation (Descriptions and Timing Diagrams)
- Application Description and Diagrams
- DC/AC Electrical Specifications, Timing Diagrams and Test Conditions

System Diagram



TL/F/8649-1

General Description (Continued)

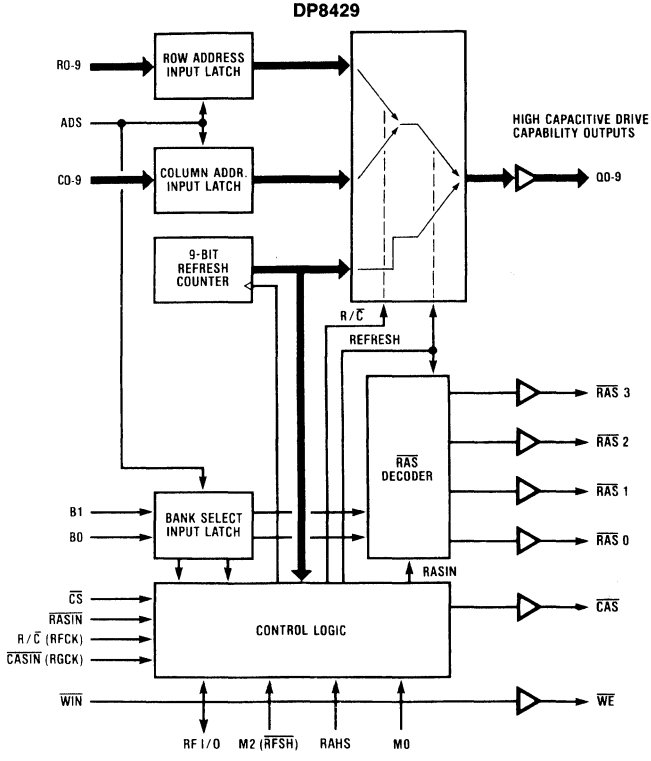
With its four independent $\overline{RAS}$ outputs and ten multiplexed address outputs, the DP8429 can support up to four banks of 64k, 256k or 1M DRAMs. Two bank select pins, B1 and B0, are decoded to activate one of the $\overline{RAS}$ signals during an access, leaving the three non-selected banks in the standby mode (less than one tenth of the operating power) with data outputs in TRI-STATE®. The DP8428's one Bank Select pin, B1, enables 2 banks automatically during an access in order to provide an optimum interface for 32-bit microprocessors.

The DP8428 and DP8429 each have two mode-select pins, allowing for two refresh modes and two access modes. Refresh and access timing may be controlled either externally or automatically. The automatic modes require a minimum of input control signals.

A refresh counter is on-chip and is multiplexed with the row and column inputs. Its contents appear at the address outputs of the DP8428 or DP8429 during any refresh, and are incremented at the completion of the refresh. Row, Column and bank address latches are also on-chip. However, if the address inputs to the DP8428 or DP8429 are valid throughout the duration of the access, these latches may be operated in the fall-through mode.

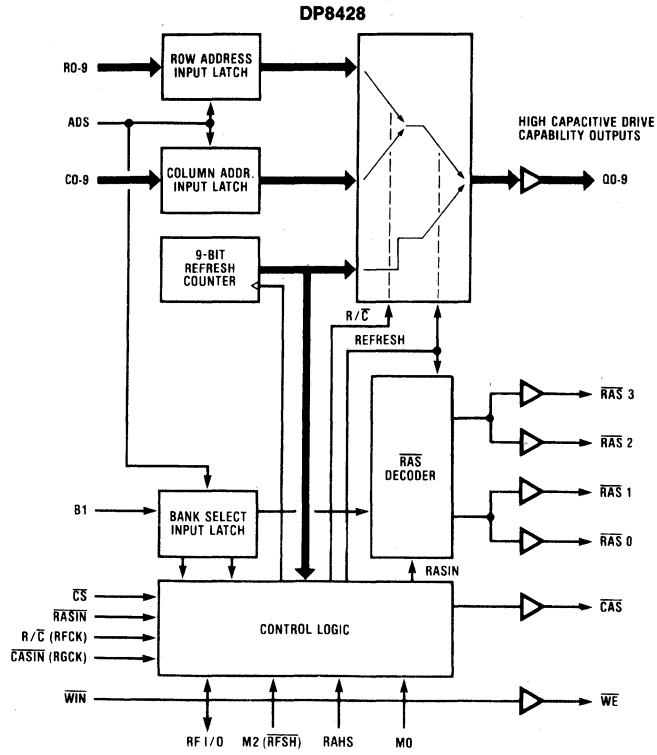
Each device is available in either the 52 pin Ceramic DIP, or the low cost JEDEC standard 68 pin Plastic Chip Carrier (PCC) package.

Functional Block Diagrams



TL/F/8649–2

Functional Block Diagrams (Continued)



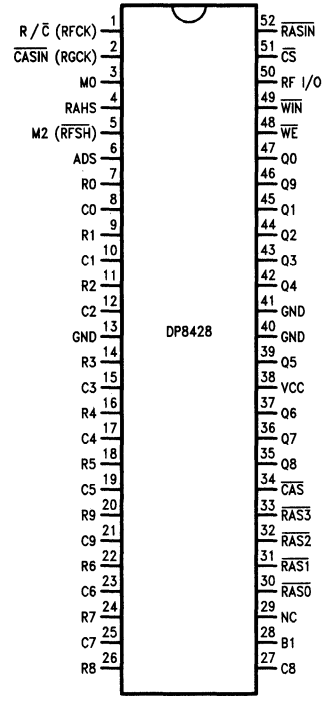
TL/F/8649-3

System Companion Components

Device #	Function
DP84300	Programmable Refresh Timer for DP84xx DRAM Controller
DP84412	NS32008/16/32 to DP8409A/17/18/19/28/29 Interface
DP84512	NS32332 to DP8417/18/19/28/29 Interface
DP84322	68000/08/10 to DP8409A/17/18/19/28/29 Interface (up to 8 MHz)
DP84422	68000/08/10 to DP8409A/17/18/19/28/29 Interface (up to 12.5 MHz)
DP84522	68020 to DP8417/18/19/28/29 Interface
DP84432	8086/88/186/188 to DP8409A/17/18/19/28/29 Interface
DP84532	80286 to DP8409A/17/18/19/28/29 Interface
DP8400-2	16-Bit Expandable Error Checker/Corrector (E2C2)
DP8402A	32-Bit Error Detector And Corrector (EDAC)

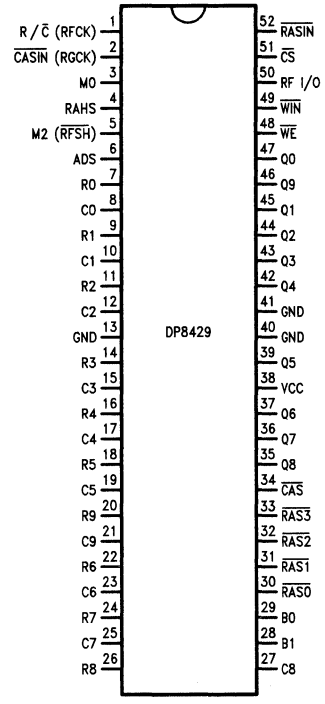
Connection Diagrams

Dual-In-Line Package



TL/F/8649–4

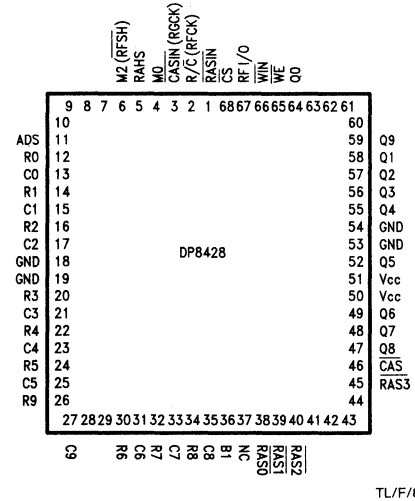
Dual-In-Line Package



TL/F/8649–5

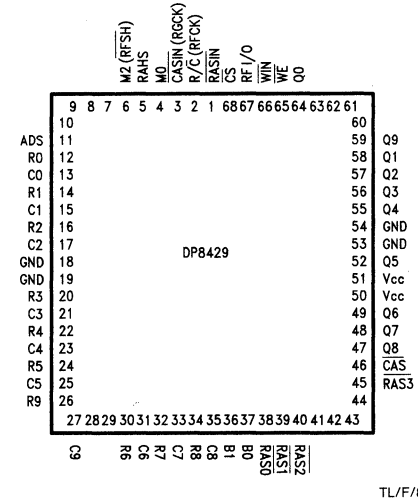
Order Number DP8428D-70, DP8428D-80 or
DP8429D-70, DP8429D-80
See NS Package Number D52A

Plastic Chip Carrier Package



TL/F/8649–6

Plastic Chip Carrier Package



TL/F/8649–7

Order Number DP8428V-70, DP8428V-80 or
DP8429V-70, DP8429V-80
See NS Package Number V68A

7

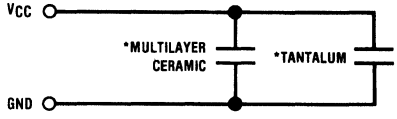
DP8428 vs DP8429

The DP8428 DYNAMIC RAM CONTROLLER/DRIVER is identical to the DP8429 with the exception of two functional differences incorporated to improve performance with 32-bit microprocessors.

1) Pin 28 (B1) is used to enable/disable a pair of $\overline{RAS}$ outputs, and pin 29 (B0 on the DP8429) is a no connect. When B1 is low, $\overline{RAS0}$ and $\overline{RAS1}$ are enabled such that they both go low during an access. When B1 is high, $\overline{RAS2}$ and $\overline{RAS3}$ are enabled. This feature is useful when driving words of 32 bits or more since each $\overline{RAS}$ would be driving only one half of the word. By distributing the load on each $\overline{RAS}$ line in this way, the DP8428 will meet the same AC specifications driving 2 banks of 32 DRAMs each as the DP8429 does driving 4 banks of 16 bits each.
2) The hidden refresh function available on the DP8429 has been disabled on the DP8428 in order to reduce the amount of setup time necessary from $\overline{CS}$ going low to $\overline{RASIN}$ going low during an access of DRAM. This parameter, called t_{CSRL1}, is 5 ns for the DP8428 whereas it is 34 ns for the DP8429. The hidden refresh function allowed only a very small increase in system performance, at microprocessor frequencies of 10 MHz and above.

Pin Definitions

V_{CC}, GND, GND − V_{CC} = 5V ±10%. The three supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. There are two ground pins to reduce the low level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 10 address bits change in the same direction simultaneously. A recommended solution would be a 1 μF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected as close as possible to GND and V_{CC} to reduce lead inductance. See Figure below.



TL/F/8649–8

*Capacitor values should be chosen depending on the particular application.

R0–R9: Row Address Inputs.

C0–C9: Column Address Inputs.

Q0–Q9: Multiplexed Address Outputs – This address is selected from the Row Address Input Latch, the Column Address Input Latch or the Refresh Counter.

$\overline{RASIN}$: Row Address Strobe Input – $\overline{RASIN}$ directly controls the selected $\overline{RAS}$ output when in an access mode and all $\overline{RAS}$ outputs during hidden or external refresh.

R/$\overline{C}$ (RFCK) – In the auto-modes this pin is the external refresh clock input; one refresh cycle should be performed each clock period. In the external access mode it is Row/Column Select Input which enables either the row or column address input latch onto the output bus.

$\overline{CASIN}$ (RGCK) – In the auto-modes this pin is the $\overline{RAS}$ Generator Clock input. In external access mode it is the Column Address Strobe input which controls $\overline{CAS}$ directly once columns are enabled on the address outputs.

ADS: Address (Latch) Strobe Input – Row Address, Column Address, and Bank Select Latches are fall-through with ADS high; latching occurs on high-to-low transition of ADS.

$\overline{CS}$: Chip Select Input – When high, $\overline{CS}$ disables all accesses. Refreshing, however, in both modes 0 and 1 is not affected by this pin.

M0, M2 ($\overline{RFSH}$): Mode Control Inputs – These pins select one of the four available operational modes of the DP8429 (see Table III).

RFI/0: Refresh Input/Output – In the auto-modes this pin is the Refresh Request Output. It goes low following RFCK indicating that no hidden refresh was performed while RFCK was high. When this pin is set low by an external gate the on-chip refresh counter is reset to all zeroes.

$\overline{WIN}$: Write Enable Input.

$\overline{WE}$: Write Enable Output – $\overline{WE}$ follows $\overline{WIN}$ unconditionally.

RAHS: Row Address Hold Time Select – Selects the t_{RAH} to be guaranteed by the DP8428 or DP8429 delay line to allow for the use of fast or slow DRAMs.

$\overline{CAS}$: Column Address Strobe Output – In mode 5 and in mode 4 with $\overline{CASIN}$ low before R/$\overline{C}$ goes low, $\overline{CAS}$ goes low automatically after the column address is valid on the address outputs. In mode 4 $\overline{CAS}$ follows $\overline{CASIN}$ directly after R/$\overline{C}$ goes low, allowing for nibble accessing. $\overline{CAS}$ is always high during refresh.

$\overline{RAS}$ 0–3: Row Address Strobe Outputs – The enabled $\overline{RAS}$ output (see Table II) follows $\overline{RASIN}$ directly during an access. During refresh, all $\overline{RAS}$ outputs are enabled.

Pin Definitions (Continued)

B0, B1: Bank Select Inputs – These pins are decoded to enable one or two of the four $\overline{RAS}$ outputs during an access (see Table I and Table II).

TABLE I. DP8429 Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{RAS}_n$
B1	B0	
0	0	$\overline{RAS}_0$
0	1	$\overline{RAS}_1$
1	0	$\overline{RAS}_2$
1	1	$\overline{RAS}_3$

TABLE II. DP8428 Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{RAS}_n$
B1	NC	
0	X	$\overline{RAS}_0$ & $\overline{RAS}_1$
1	X	$\overline{RAS}_2$ & $\overline{RAS}_3$

Conditions for All Modes

INPUT ADDRESSING

The address block consists of a row-address latch, a column-address latch, and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses until after $\overline{CAS}$ goes low at the end of the memory cycle, ADS can be permanently high. Otherwise ADS must go low while the addresses are still valid.

DRIVE CAPABILITY

The DP8429 has timing parameters that are specified driving the typical capacitance (including traces) of 88, 5V-only DRAMs. Since there are 4 $\overline{RAS}$ outputs, each is specified driving one-fourth of the total memory. $\overline{CAS}$, $\overline{WE}$ and the address outputs are specified driving all 88 DRAMs.

The graph in *Figure 10* may be used to determine the slight variations in timing parameters, due to loading conditions other than 88 DRAMs.

Because of distributed trace capacitance and inductance and DRAM input capacitance, current spikes can be created, causing overshoots and undershoots at the DRAM inputs that can change the contents of the DRAMs or even destroy them. To reduce these spikes, a damping resistor (low inductance, carbon) should be inserted between the DP8429 outputs and the DRAMs, as close as possible to the DP8429. The damping resistor values may differ depending on how heavily an output is loaded. These resistors should be determined by the first prototypes (not wire-wrapped due to the larger distributed capacitance and inductance). Resistors should be chosen such that the transition on the control outputs is critically damped. Typical values will be from 15Ω to 100Ω, with the lower values being used with the larger memory arrays. Note that AC parameters are specified with 15Ω damping resistors. For more information see AN-305 "Precautions to Take When Driving Memories".

DP8429 DRIVING ANY 256k or 1M DRAMS

The DP8429 can drive any 256k or 1M DRAMs. 256k DRAMs require 18 of the DP8429's address inputs to select one memory location within the DRAM. $\overline{RAS}$-only refreshing with the nine-bit refresh-counter on the DP8429 makes $\overline{CAS}$ before $\overline{RAS}$ refreshing, available on 256k DRAMs, unnecessary (see *Figure 1a*).

1 Mbit DRAMs require the use of all 10 of the DP8429 Address Outputs (see *Figure 1b*).

READ, WRITE AND READ-MODIFY-WRITE CYCLES

The output signal, $\overline{WE}$, determines what type of memory access cycle the memory will perform. If $\overline{WE}$ is kept high while $\overline{CAS}$ goes low, a read cycle occurs. If $\overline{WE}$ goes low before $\overline{CAS}$ goes low, a write cycle occurs and data at DI (DRAM input data) is written into the DRAM as $\overline{CAS}$ goes low. If $\overline{WE}$ goes low later than t_{CWD} after $\overline{CAS}$ goes low, first a read occurs and DO (DRAM output data) becomes valid, then data DI is written into the same address in the DRAM as $\overline{WE}$ goes low. In this read-modify-write case, DI and DO cannot be linked together. $\overline{WE}$ always follows $\overline{WIN}$ directly to determine the type of access to be performed.

POWER-UP INITIALIZE

When V_{CC} is first applied to the DP8429, an initialize pulse clears the refresh counter and the internal control flip-flops.

Mode Features Summary

- 4 modes of operation: 2 access and 2 refresh
- Automatic or external selected by the user
- Auto access mode provides $\overline{RAS}$, row to column change, and then $\overline{CAS}$ automatically.
- Choice between two different values of t_{RAH} in auto-access mode
- $\overline{CAS}$ controlled independently in external control mode, allowing for nibble mode accessing
- Automatic refreshing can make refreshes transparent to the system
- $\overline{CAS}$ is inhibited during refresh cycles

DP8428/DP8429 Mode Descriptions

MODE 0–EXTERNALLY CONTROLLED REFRESH

Figure 2 shows the Externally Controlled Refresh timing. In this mode the refresh counter contents are multiplexed to the address outputs. All $\overline{RAS}$ outputs are enabled to follow $\overline{RASIN}$ so that the row address indicated by the refresh counter is refreshed in all DRAM banks when $\overline{RASIN}$ goes low. The refresh counter increments when $\overline{RASIN}$ goes high. $\overline{RFSH}$ should be held low at least until $\overline{RASIN}$ goes high (they may go high simultaneously) so that the refresh address remains valid and all $\overline{RAS}$ outputs remain enabled throughout the refresh.

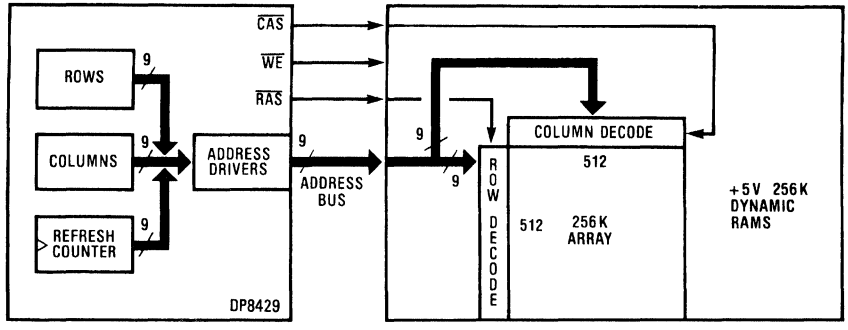
A burst refresh may be performed by holding $\overline{RFSH}$ low and toggling $\overline{RASIN}$ until all rows are refreshed. It may be useful in this case to reset the refresh counter just prior to beginning the refresh. The refresh counter resets to all zeroes when RFI/O is pulled low by an external gate. The refresh counter always counts to 511 before rolling over to zero. If there are 128 or 256 rows being refreshed then Q7 or Q8, respectively, going high may be used as an end-of-burst indicator.

In order that the refresh address is valid on the address outputs prior to the $\overline{RAS}$ lines going low, $\overline{RFSH}$ must go low before $\overline{RASIN}$. The setup time required is given by t_{RFLRL} in the Switching Characteristics. This parameter may be adjusted using *Figure 10* for loading conditions other than those specified.

TABLE III. DP8428/DP8429 Mode Select Options

Mode	($\overline{RFSH}$) M2	M0	Mode of Operation
0	0	0	Externally Controlled Refresh
1	0	1	Auto Refresh–Forced
4	1	0	Externally Controlled Access
5	1	1	Auto Access (Hidden Refresh)

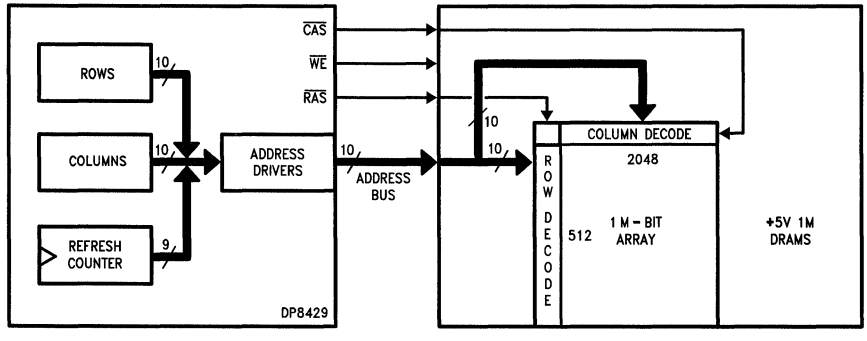
DP8428/DP8429 Interface Between System and DRAM Banks



TL/F/8649–12

All 9 Bits of Refresh Counter Used

FIGURE 1a. DP8428/DP8429 with 256k DRAMs

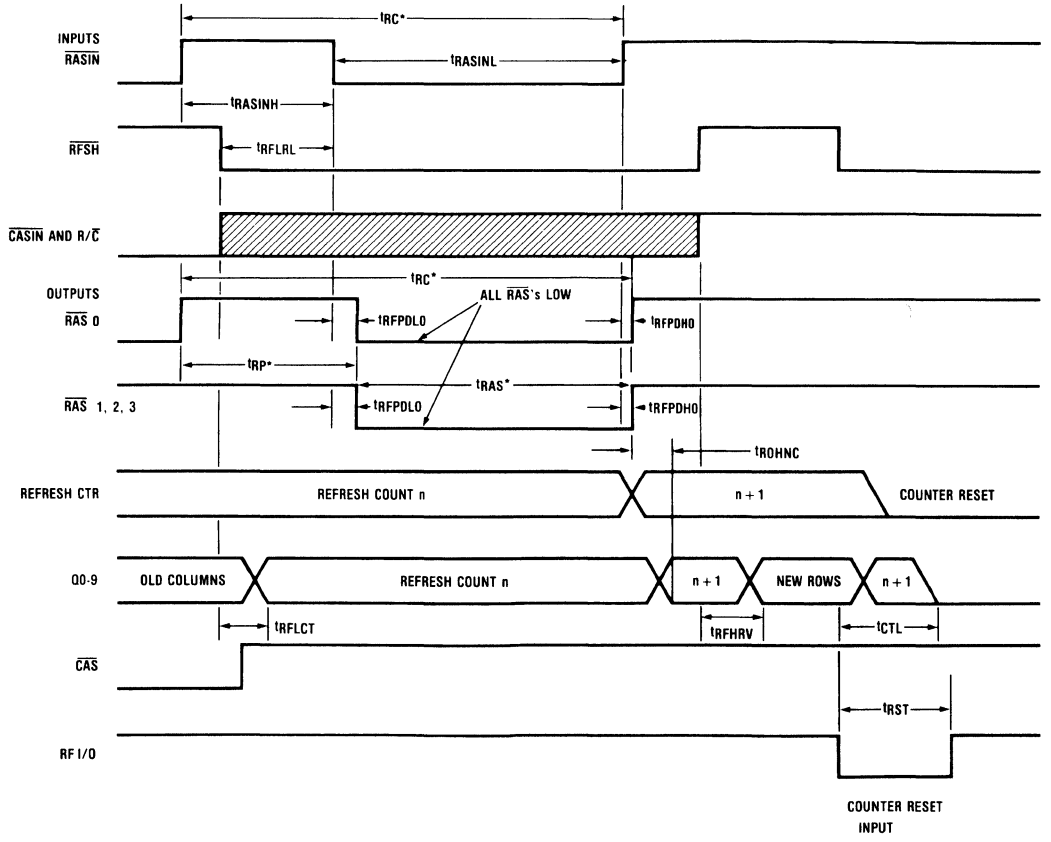


TL/F/8649–25

All 9 Bits of Refresh Counter Used

FIGURE 1b. DP8428/DP8429 with 1M DRAMs

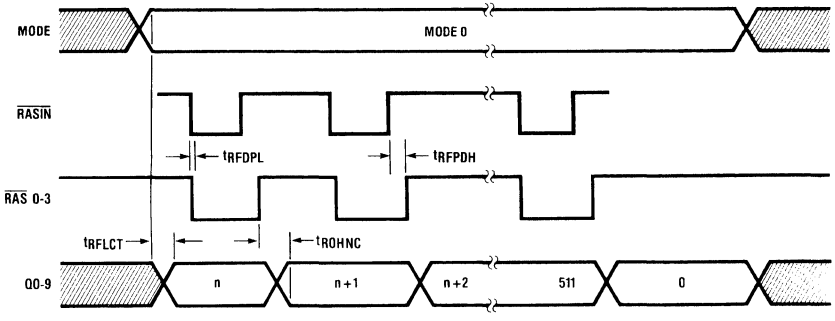
DP8428/DP8429 Mode Descriptions (Continued)



*Indicates Dynamic RAM Parameters

TL/F/8649–13

FIGURE 2a. External Control Refresh Cycle (Mode 0)



TL/F/8649–14

FIGURE 2b. Burst Refresh Mode 0

DP8428/DP8429 Mode Descriptions (Continued)

MODE 1–AUTOMATIC FORCED REFRESH

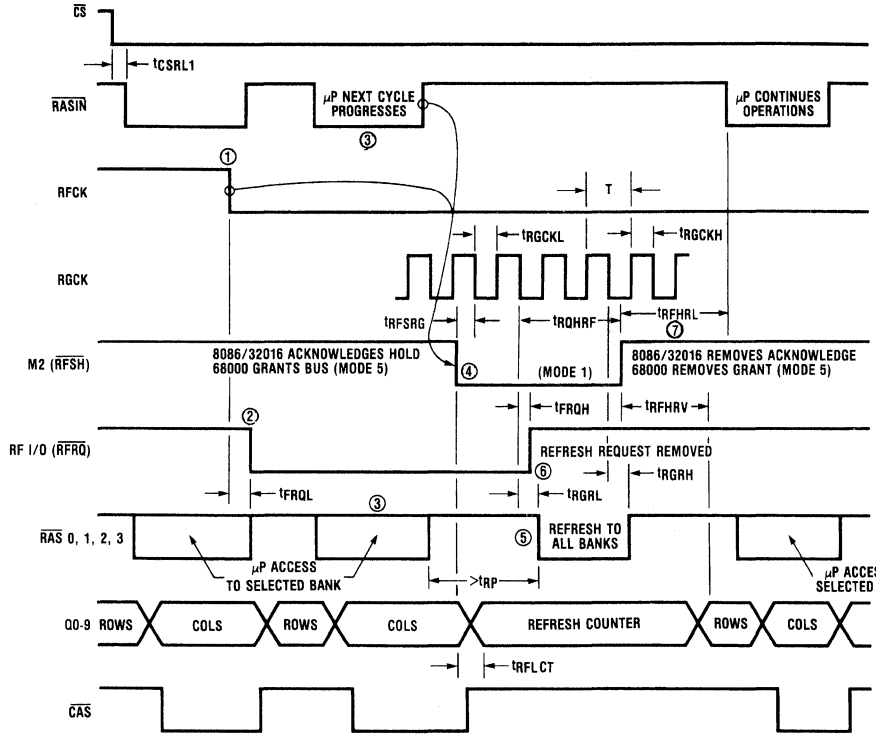
In Mode 1 the R/$\overline{C}$ ($\overline{RFCK}$) pin becomes RFCK (refresh cycle clock) and the $\overline{CASIN}$ (RGCK) pin becomes RGCK ($\overline{RAS}$ generator clock). If RFCK is high and Mode 1 is entered then the chip operates as if in MODE 0 (externally controlled refresh), with all $\overline{RAS}$ outputs following $\overline{RASIN}$. This feature of Mode 1 may be useful for those who want to use Mode 5 (automatic access) with externally controlled refresh. By holding RFCK permanently high one need only toggle M2 ($\overline{RFSH}$) to switch from Mode 5 to external refresh. As with Mode 0, RFI/O may be pulled low by an external gate to reset the refresh counter.

When using Mode 1 as automatic refresh, RFCK must be an input clock signal. One refresh should occur each period of RFCK. If no refresh is performed while RFCK is high, then when RFCK goes low RFI/O immediately goes low to indicate that a refresh is requested. (RFI/O may still be used to reset the refresh counter even though it is also used as a refresh request pin, however, an open-collector gate should be used to reset the counter in this case since RFI/O is forced low internally for a request).

After receiving the refresh request the system must allow a forced refresh to take place while RFCK is low. External logic can monitor $\overline{RFRQ}$ (RFI/O) so that when $\overline{RFRQ}$ goes low this logic will wait for the access currently in progress to be completed before pulling M2 ($\overline{RFSH}$) low to put the DP8429 in mode 1. If no access is taking place when $\overline{RFRQ}$ occurs, then M2 may immediately go low. Once M2 is low, the refresh counter contents appear at the address outputs and $\overline{RAS}$ is generated to perform the refresh.

An external clock on RGCK is required to derive the refresh $\overline{RAS}$ signals. On the second falling edge of RGCK after M2 is low, all $\overline{RAS}$ lines go low. They remain low until two more falling edges of RGCK. Thus $\overline{RAS}$ remains high for one to two periods of RGCK after M2 goes low, and stays low for two periods. In order to obtain the minimum delay from M2 going low to $\overline{RAS}$ going low, M2 should go low t_{RFSRG} before the falling edge of RGCK.

The Refresh Request on RFI/O is terminated as $\overline{RAS}$ goes low. This signal may be used to end the refresh earlier than it normally would as described above. If M2 is pulled high



TL/F/8649–15

① RFCK goes low
② $\overline{RFRQ}$ goes low if no hidden refresh occurred while RFCK was high
③ Next $\overline{RASIN}$ starts next access
④ μP acknowledges refresh request
⑤ Forced refresh $\overline{RAS}$ starts after > T (> t_{RP})
⑥ Forced refresh $\overline{RAS}$ ends $\overline{RFRQ}$
⑦ μP removes refresh acknowledge

FIGURE 3. DP8428/DP8429 Performing a Forced Refresh (Mode 5 → 1 → 5) with Various Microprocessors

DP8428/DP8429 Mode Descriptions (Continued)

while the $\overline{RAS}$ lines are low, then the $\overline{RAS}$s go high t_{RFRH} later. The designer must be careful, however, not to violate the minimum $\overline{RAS}$ low time of the DRAMs. He must also guarantee that the minimum $\overline{RAS}$ precharge time is not violated during a transition from mode 1 to mode 5 when an access is desired immediately following a refresh.

If the processor tries to access memory while the DP8429 is in mode 1, WAIT states should be inserted into the processor cycles until the DP8429 is back in mode 5 and the desired access has been accomplished (see *Figure 9*).

Instead of using WAIT states to delay accesses when refreshing, HOLD states could be used as follows. $\overline{RFRQ}$ could be connected to a HOLD or Bus Request input to the system. When convenient, the system acknowledges the $\overline{HOLD}$ or Bus Request by pulling M2 low. Using this scheme, $\overline{HOLD}$ will end as the $\overline{RAS}$ lines go low (RFI/O goes high). Thus, there must be sufficient delay from the time $\overline{HOLD}$ goes high to the DP8429 returning to mode 5, so that the $\overline{RAS}$ low time of the DRAMs isn't violated as described earlier (see *Figure 3* for mode 1 refresh with Hold states).

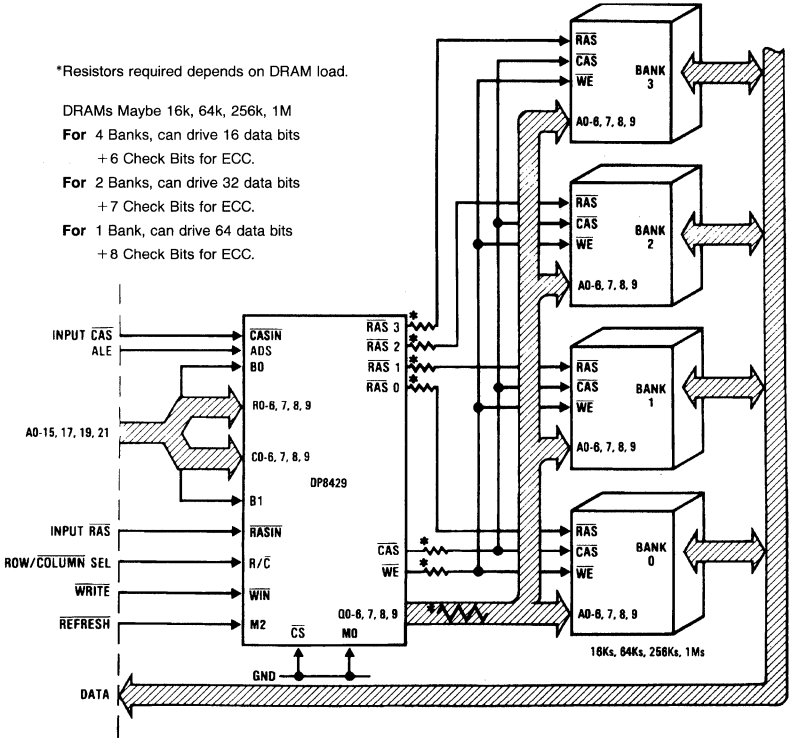
To perform a forced refresh the system will be inactive for about four periods of RGCK. For a frequency of 10 MHz, this is 400 ns. To refresh 128 rows every 2 ms an average of about one refresh per 16 μs is required. With a RFCK period of 16 μs and RGCK period of 100 ns, DRAM accesses are delayed due to refresh only 2.5% of the time. If using the Hidden Refresh available in mode 5 (refreshing with RFCK high) this percentage will be even lower.

MODE 4 - EXTERNALLY CONTROLLED ACCESS

In this mode all control signal outputs can be controlled directly by the corresponding control input. The enabled $\overline{RAS}$ output follows $\overline{RASIN}$, $\overline{CAS}$ follows $\overline{CASIN}$ (with R/$\overline{C}$ low), $\overline{WE}$ follows $\overline{WIN}$ and R/$\overline{C}$ determines whether the row or the column inputs are enabled to the address outputs (see *Figure 4*).

With R/$\overline{C}$ high, the row address latch contents are enabled onto the address bus. $\overline{RAS}$ going low strobes the row address into the DRAMs. After waiting to allow for sufficient row-address hold time (t_{RAH}) after $\overline{RAS}$ goes low, R/$\overline{C}$ can go low to enable the column address latch contents onto the address bus. When the column address is valid, $\overline{CAS}$ going low will strobe it into the DRAMs. $\overline{WIN}$ determines whether the cycle is a read, write or read-modify-write access. Refer to *Figures 5a* and *5b* for typical Read and Write timing using mode 4.

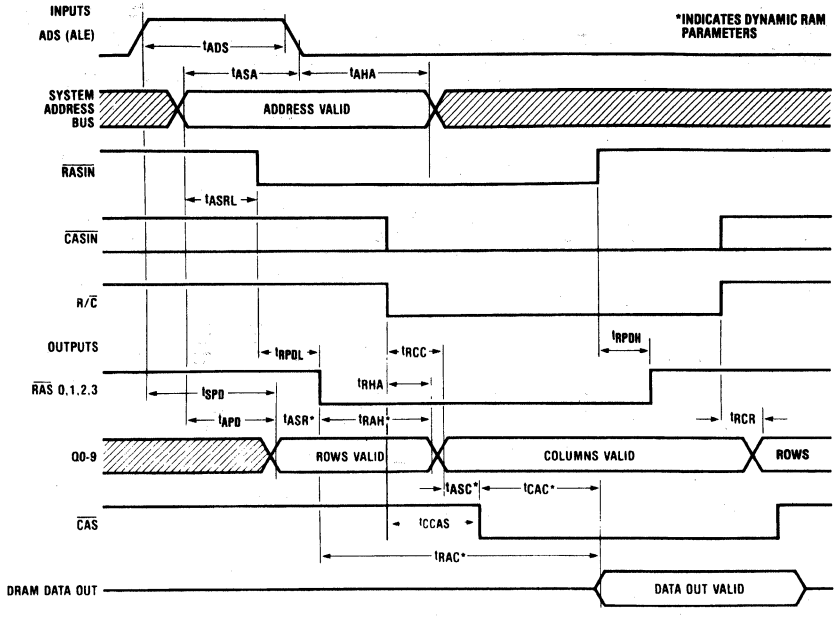
Page or Nibble mode may be performed by toggling $\overline{CASIN}$ once the initial access has been completed. In the case of page mode the column address must be changed before



TL/F/8649–16

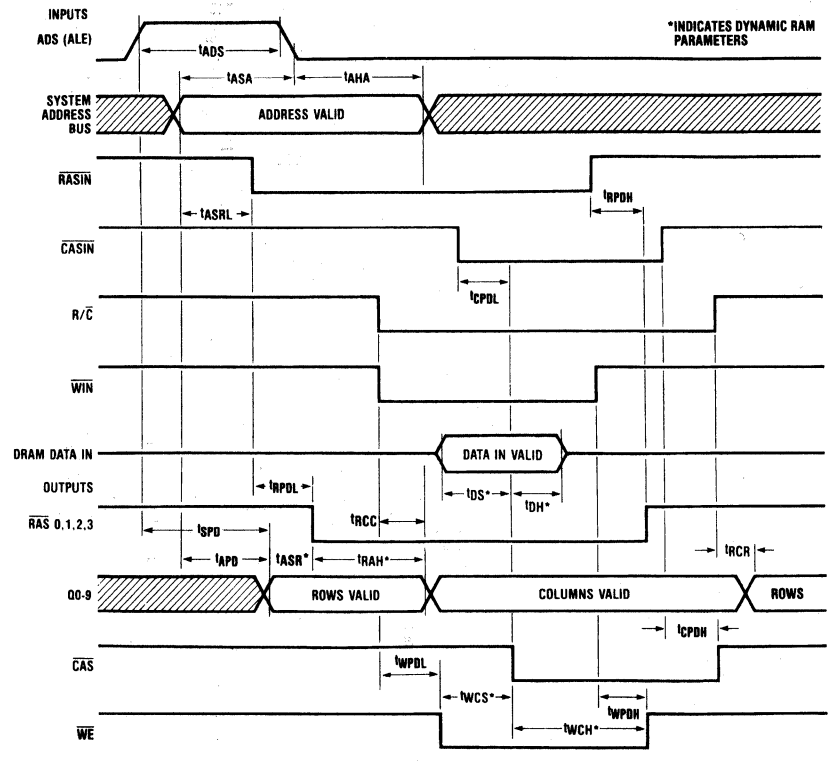
FIGURE 4. Typical Application of DP8429 Using External Control Access and Refresh in Modes 0 and 4

DP8428/DP8429 Mode Descriptions (Continued)



TL/F/8649–17

FIGURE 5a. Read Cycle Timing (Mode 4)



TL/F/8649–18

FIGURE 5b. Write Cycle Timing (Mode 4)

DP8428/DP8429 Mode Descriptions (Continued)

$\overline{CASIN}$ goes low to access a new memory location (see *Figure 5c*). Parameter t_{CPdif} has been specified in order that users may easily determine minimum $\overline{CAS}$ pulse widths when $\overline{CASIN}$ is toggling.

AUTOMATIC $\overline{CAS}$ GENERATION

$\overline{CAS}$ is held high when R/$\overline{C}$ is high even if $\overline{CASIN}$ is low. If $\overline{CASIN}$ is low when R/$\overline{C}$ goes low, $\overline{CAS}$ goes low automatically, t_{ASC} after the column address is valid. This feature eliminates the need for an externally derived $\overline{CASIN}$ signal to control $\overline{CAS}$ when performing a simple access (*Figure 5a* demonstrates Auto-$\overline{CAS}$ generation in mode 4). Page or nibble accessing may be performed as shown in *Figure 5c* even if $\overline{CAS}$ is generated automatically for the initial access.

FASTEST MEMORY ACCESS

The fastest Mode 4 access is achieved by using the automatic $\overline{CAS}$ feature and external delay line to generate the required delay between $\overline{RASIN}$ and R/$\overline{C}$. The amount of delay required depends on the minimum t_{RAH} of the DRAMs being used. The DP8429 parameter t_{DIF1} has been specified in order that the delay between $\overline{RASIN}$ and R/$\overline{C}$ may be minimized.

$$t_{DIF1} = \text{MAXIMUM } (t_{RPDL} - t_{RHA})$$

where t_{RPDL} = $\overline{RASIN}$ to $\overline{RAS}$ delay

and t_{RHA} = row address held from R/$\overline{C}$ going low.

The delay between $\overline{RASIN}$ and R/$\overline{C}$ that guarantees the specified DRAM t_{RAH} is given by

$$\text{MINIMUM } \overline{RASIN} \text{ to R/}\overline{C} = t_{DIF1} + t_{RAH}.$$

Example

In an application using DRAMs that require a minimum t_{RAH} of 15 ns, the following demonstrates how the maximum $\overline{RASIN}$ to $\overline{CAS}$ time is determined.

With t_{DIF1} (from Switching Characteristics) = 7 ns,

$\overline{RASIN}$ to R/$\overline{C}$ delay = 7 ns + 15 ns = 22 ns.

A delay line of 25 ns will be sufficient.

With Auto-$\overline{CAS}$ generation, the maximum delay from R/$\overline{C}$ to $\overline{CAS}$ (loaded with 600 pF) is 46 ns. Thus the maximum $\overline{RASIN}$ to $\overline{CAS}$ time is 71 ns, under the given conditions.

With a maximum $\overline{RASIN}$ to $\overline{RAS}$ time (t_{RPDL}) of 20 ns, the maximum $\overline{RAS}$ to $\overline{CAS}$ time is about 51 ns. Most DRAMs with a 15 ns minimum t_{RAH} have a maximum t_{RCD} of about 60 ns. Thus memory accesses are likely to be $\overline{RAS}$ limited instead of $\overline{CAS}$ limited. In other words, memory access time is limited by DRAM performance, not controller performance.

REFRESHING IN CONJUNCTION WITH MODE 4

If using mode 4 to access memory, mode 0 (externally controlled refresh) must be used for all refreshing.

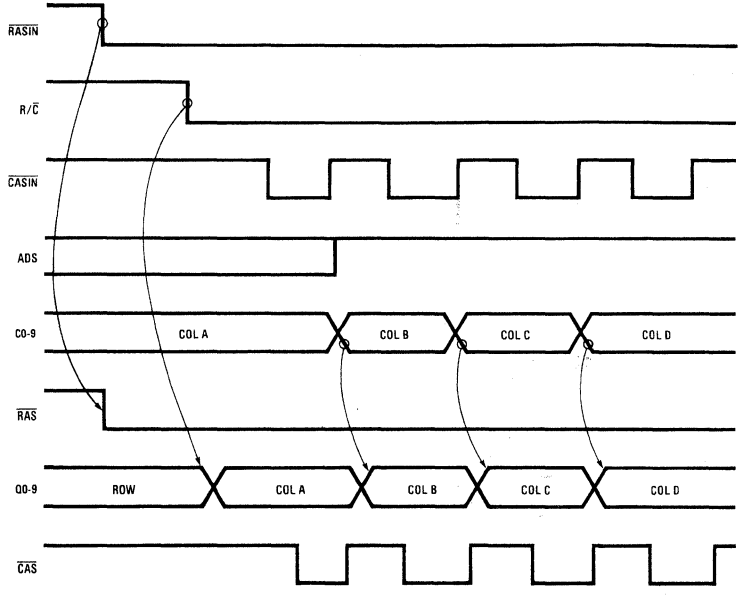
MODE 5 – AUTOMATIC ACCESS WITH HIDDEN REFRESHING CAPABILITY

Automatic-Access has two advantages over the externally controlled access (mode 4). First, $\overline{RAS}$, $\overline{CAS}$ and the row to column change are all derived internally from one input signal, $\overline{RASIN}$. Thus the need for an external delay line (see mode 4) is eliminated.

Secondly, since R/$\overline{C}$ and $\overline{CASIN}$ are not needed to generate the row to column change and $\overline{CAS}$, these pins can be used for the automatic refreshing function.

AUTOMATIC ACCESS CONTROL

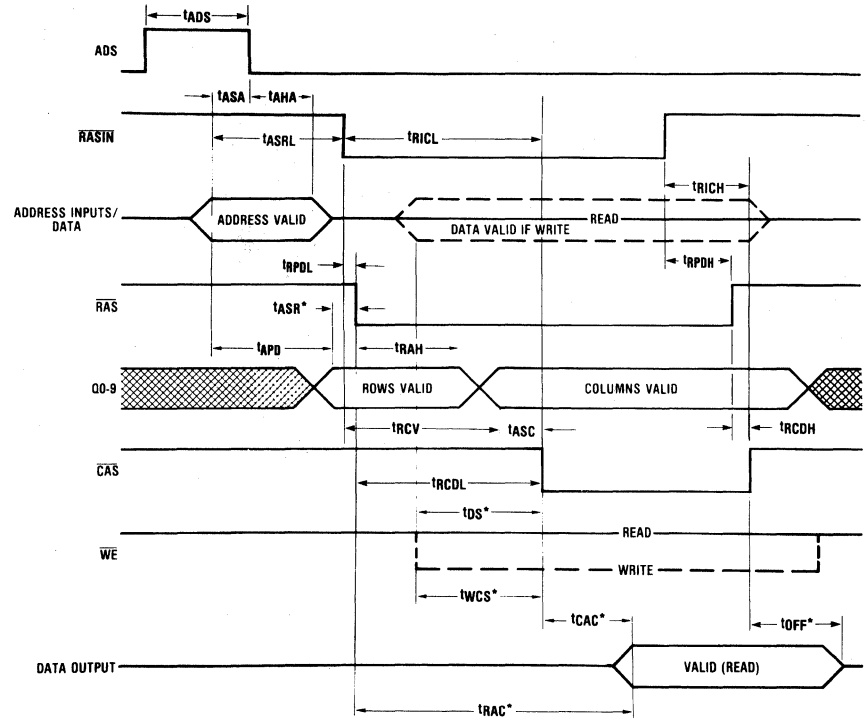
Mode 5 of the DP8429 makes accessing Dynamic RAM nearly as easy as accessing static RAM. Once row and column addresses are valid (latched on the DP8429 if necessary), $\overline{RASIN}$ going low is all that is required to perform the memory access.



TL/F/8649–19

FIGURE 5c. Page or Nibble Access in Mode 4

DP8428/DP8429 Mode Descriptions (Continued)



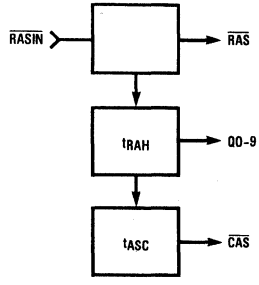
*Indicates Dynamic RAM Parameters

TL/F/8649–20

FIGURE 6. Mode 5 Timing

(Refer to *Figure 6*) In mode 5 the selected $\overline{RAS}$ follows $\overline{RASIN}$ immediately, as in mode 4, to strobe the row address into the DRAMs. The row address remains valid on the DP8429 address outputs long enough to meet the t_{RAH} requirement of the DRAMs (pin 4, RAHS, of the DP8429 allows the user two choices of t_{RAH}). Next, the column address replaces the row address on the address outputs and $\overline{CAS}$ goes low to strobe the columns into the DRAMs. $\overline{WIN}$ determines whether a read, write or read-modify-write is done.

The diagram below illustrates mode 5 automatic control signal generation.



TL/F/8649–21

REFRESHING IN CONJUNCTION WITH MODE 5

When using mode 5 to perform memory accesses, refreshing may be accomplished:

(a) externally (in mode 0 or mode 1)

(b) by a combination of mode 5 (hidden refresh) and mode 1 (auto-refresh)

or (c) by a combination of mode 5 and mode 0

(a) Externally Controlled Refreshing in Mode 0 or Mode 1

All refreshing may be accomplished using external refreshes in either mode 0 or mode 1 with R/$\overline{C}$ (RFCK) tied high (see mode 0 and mode 1 descriptions). If this is desired, the system determines when a refresh will be performed, puts the DP8429 in the appropriate mode, and controls the $\overline{RAS}$ signals directly with $\overline{RASIN}$. The on-chip refresh counter is enabled to the address outputs of the DP8429 when the refresh mode is entered, and increments when $\overline{RASIN}$ goes high at the completion of the refresh.

(b) Mode 5 Refreshing (hidden) with Mode 1 refreshing (auto)

(Refer to *Figure 7a*) If RFCK is tied to a clock (see mode 1 description), RFI/O becomes a refresh request output and goes low following RFCK going low if no refresh occurred while RFCK was high. Refreshes may be performed in mode 5 when the DP8429 is not selected for access ($\overline{CS}$ is high) and RFCK is high. If these conditions exist the refresh counter contents appear on the DP8429 address outputs and all $\overline{RAS}$ lines follow $\overline{RASIN}$ so that if $\overline{RASIN}$ goes low (an access other than through the DP8429 occurs), all $\overline{RAS}$ lines go low to perform the refresh. The DP8429 allows only one refresh of this type for each period of RFCK, since RFCK should be fast enough such that one refresh per period is sufficient to meet the DRAM refresh requirement.

DP8428/DP8429 Mode Descriptions (Continued)

Once it is started, a hidden refresh will continue even if RFCK goes low. However, $\overline{CS}$ must be high throughout the refresh (until $\overline{RASIN}$ goes high).

These hidden refreshes are valuable in that they do not delay accesses. When determining the duty cycle of RFCK, the high time should be maximized in order to maximize the probability of hidden refreshes. If a hidden refresh doesn't happen, then a refresh request will occur on RFI/O when RFCK goes low. After receiving the request, the system must perform a refresh while RFCK is low. This may be done by going to mode 1 and allowing an automatic refresh (see mode 1 description). This refresh must be completed while RFCK is low, thus the RFCK low time is determined by the worst-case time required by the system to respond to a refresh request.

(c) Mode 5 Refresh (Hidden Refresh) with mode 0 Refresh (External Refresh)

This refresh scheme is identical to that in (b) except that after receiving a refresh request, mode 0 is entered to do the refresh (see mode 0 description). The refresh request is terminated (RFI/O goes high) as soon as mode 0 is entered. This method requires more control than using mode 1 (auto-refresh), however, it may be desirable if the mode 1 refresh time is considered to be excessive.

Example

Figure 7b demonstrates how a system designer would use the DP8429 in mode 5 based on certain characteristics of his system.

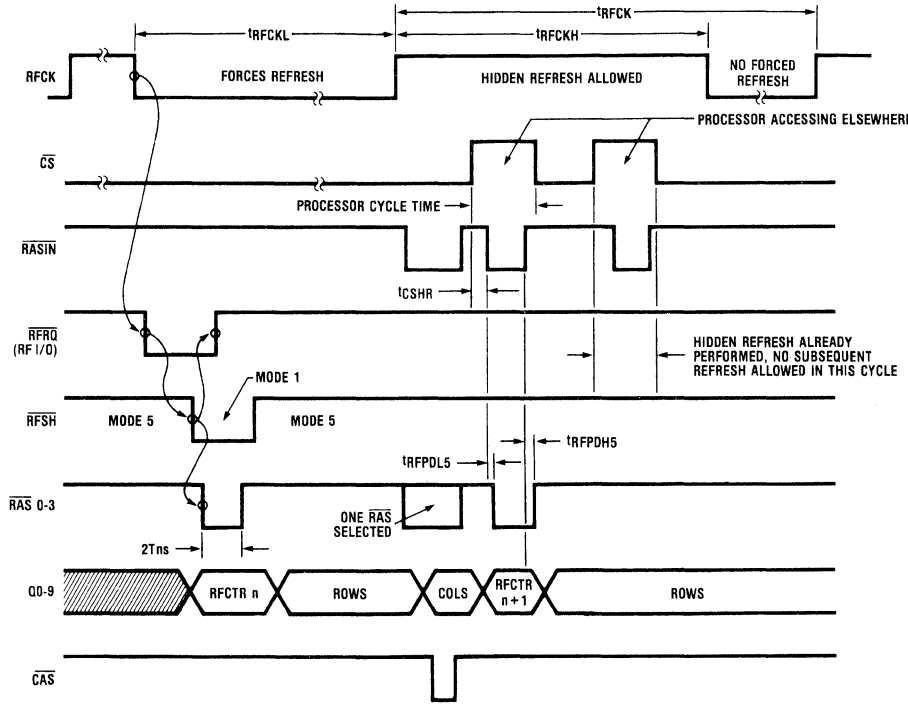
System Characteristics:

1) DRAM used has min t_{RAH} requirement of 15 ns and min t_{ASR} of 0 ns
2) DRAM address is valid from time T_V to the end of the memory cycle
3) four banks of twenty-two 256k memory chips each are being driven

Using the DP8429 (see *Figure 7b*):

1) Tie pin 4 (RAHS) high to guarantee a 15 ns minimum t_{RAH} which is sufficient for the DRAMs being used
2) Generate $\overline{RASIN}$ no earlier than time $T_V + t_{ASRL}$ (see switching characteristics), so that the row address is valid on the DRAM address inputs before $\overline{RAS}$ occurs
3) Tie ADS high since latching the DRAM address on the DP8429 is not necessary
4) Connect the first 20 system address bits to R0-R9 and C0-C9, and bits 21 and 22 to B0 and B1
5) Connect each $\overline{RAS}$ output of the DP8429 to the $\overline{RAS}$ inputs of the DRAMs of one bank of the memory array; connect Q0-Q9 of the DP8429 to A0-A9 of all DRAMs; connect $\overline{CAS}$ of the DP8429 to $\overline{CAS}$ of all the DRAMs

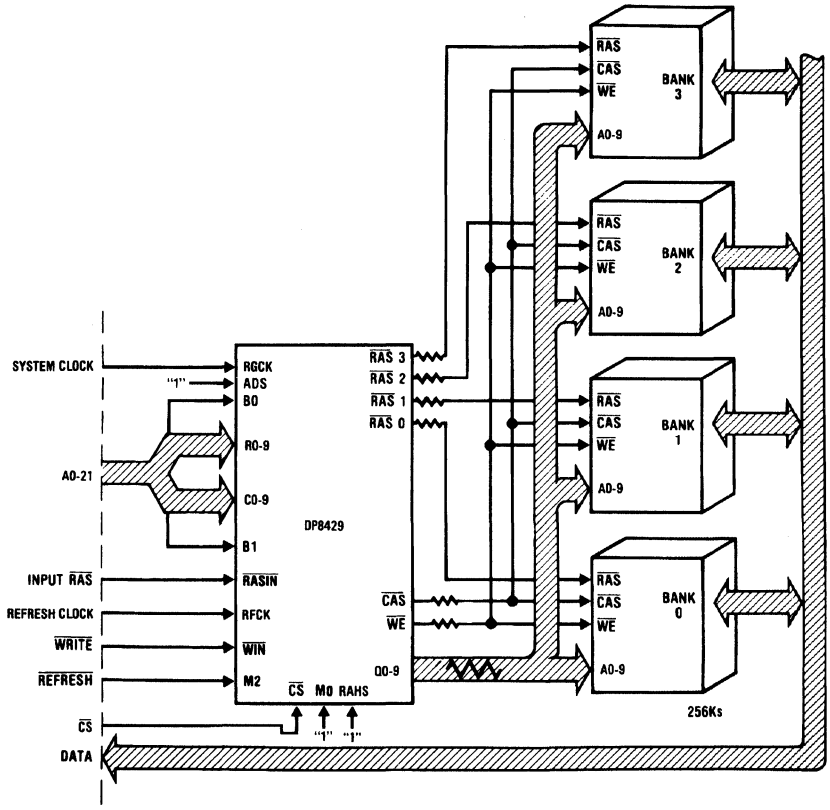
Figure 7c illustrates a similar example using the DP8428 to drive two 32-bit banks.



TL/F/8649–22

FIGURE 7a. Hidden Refreshing (Mode 5) and Forced Refreshing (Mode 1) Timing

DP8428/DP8429 Mode Descriptions (Continued)



TL/F/8649–23

FIGURE 7b. Typical Application of DP8429 Using Modes 5 and 1

Applications

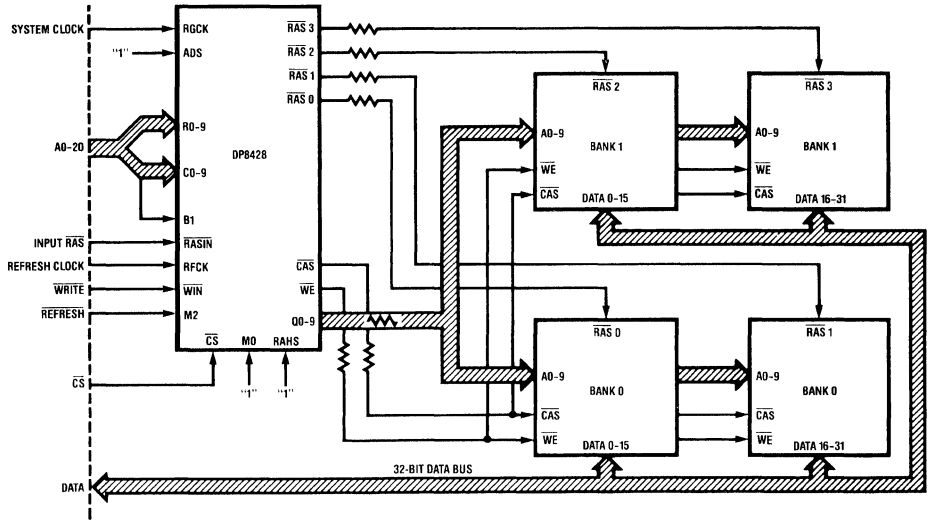
If one desires a memory interface containing the DP8429 that minimizes the number of external components required, modes 5 and 1 should be used. These two modes provide:

1) Automatic access to memory (in mode 5 only one signal, $\overline{RASIN}$, is required in order to access memory)
2) Hidden refresh capability (refreshes are performed automatically while in mode 5 when non-local accesses are taking place, as determined by $\overline{CS}$)
3) Refresh request capability (if no hidden refresh took place while RFCK was high, a refresh request is generated at the RFI/O pin when RFCK goes high)
4) Automatic forced refresh (If a refresh request is generated while in mode 5, as described above, external logic should switch the DP8429 into mode 1 to do an automatic forced refresh. No other external control signals need be issued. WAIT states can be inserted into the processor machine cycles if the system tries to access memory while the DP8429 is in mode 1 doing a forced refresh).

Some items to be considered when integrating the DP8429 into a system design are:

1) The system designer should ensure that a DRAM access not be in progress when a refresh mode is entered. Similarly, one should not attempt to start an access while a refresh is in progress. The parameter t_{RFHRL} specifies the minimum time from $\overline{RFSH}$ high to $\overline{RASIN}$ going low to initiate an access.
2) One should always guarantee that the DP8429 is enabled for access prior to initiating the access (see t_{CSRL1}).
3) One should bring $\overline{RASIN}$ low even during non-local access cycles when in mode 5 in order to maximize the chance of a hidden refresh occurring.
4) At lower frequencies (under 10 Mhz), it becomes increasingly important to differentiate between READ and WRITE cycles. $\overline{RASIN}$ generation during READ cycles can take place as soon as one knows that a processor READ access cycle has started. WRITE cycles, on the other hand, cannot start until one knows that the data to be written at the DRAM inputs will be valid a setup time before $\overline{CAS}$ (column address strobe) goes true at the DRAM inputs. Therefore, in general, READ cycles can be initiated earlier than WRITE cycles.
5) Many times it is possible to only add WAIT states during READ cycles and have no WAIT states during WRITE cycles. This is because it generally takes less time to write data into memory than to read data from memory.

Applications (Continued)



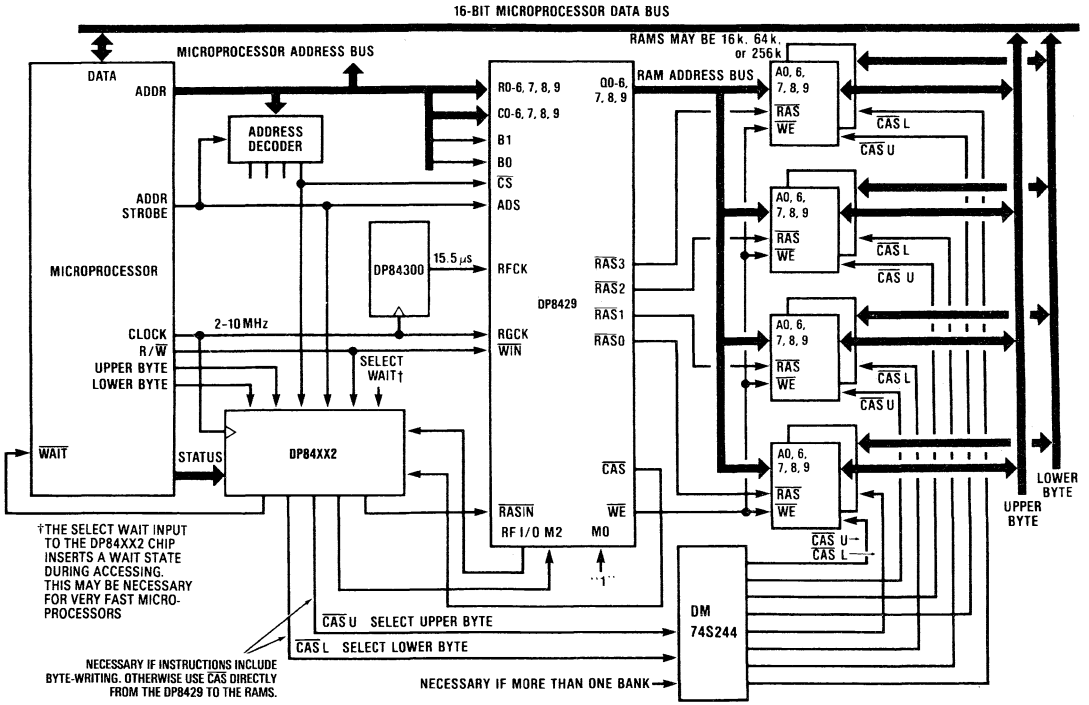
TL/F/8649–24

FIGURE 7c. Typical Application of DP8428 Using Modes 5 and 1

The DP84XX2 family of inexpensive preprogrammed medium Programmable Array Logic devices (PALs) have been developed to provide an easy interface between various microprocessors and the DP84XX family of DRAM controller/drivers. These PALs interface to all the necessary control signals of the particular processor and the DP8429. The PAL controls the operation of the DP8429 in modes 5 and 1, while meeting all the critical timing considerations discussed above. The refresh clock, RFCK, may be divided down from the processor clock using an IC counter such as the DM74LS393 or the DP84300 programmable refresh timer. The DP84300 can provide RFCK periods ranging from 15.4 μs to 15.6 μs based on an input clock of 2 to 10 MHz. *Figure 8* shows a general block diagram for a system using the DP8429 in modes 1 and 5. *Figure 9* shows possible timing diagrams for such a system (using $\overline{WAIT}$ to prohibit access when refreshing). Although the DP84XX2 PALs are offered as standard peripheral devices for the DP84XX DRAM controller/drivers, the programming equations for these devices are provided so the user may make minor modifications for unique system requirements.

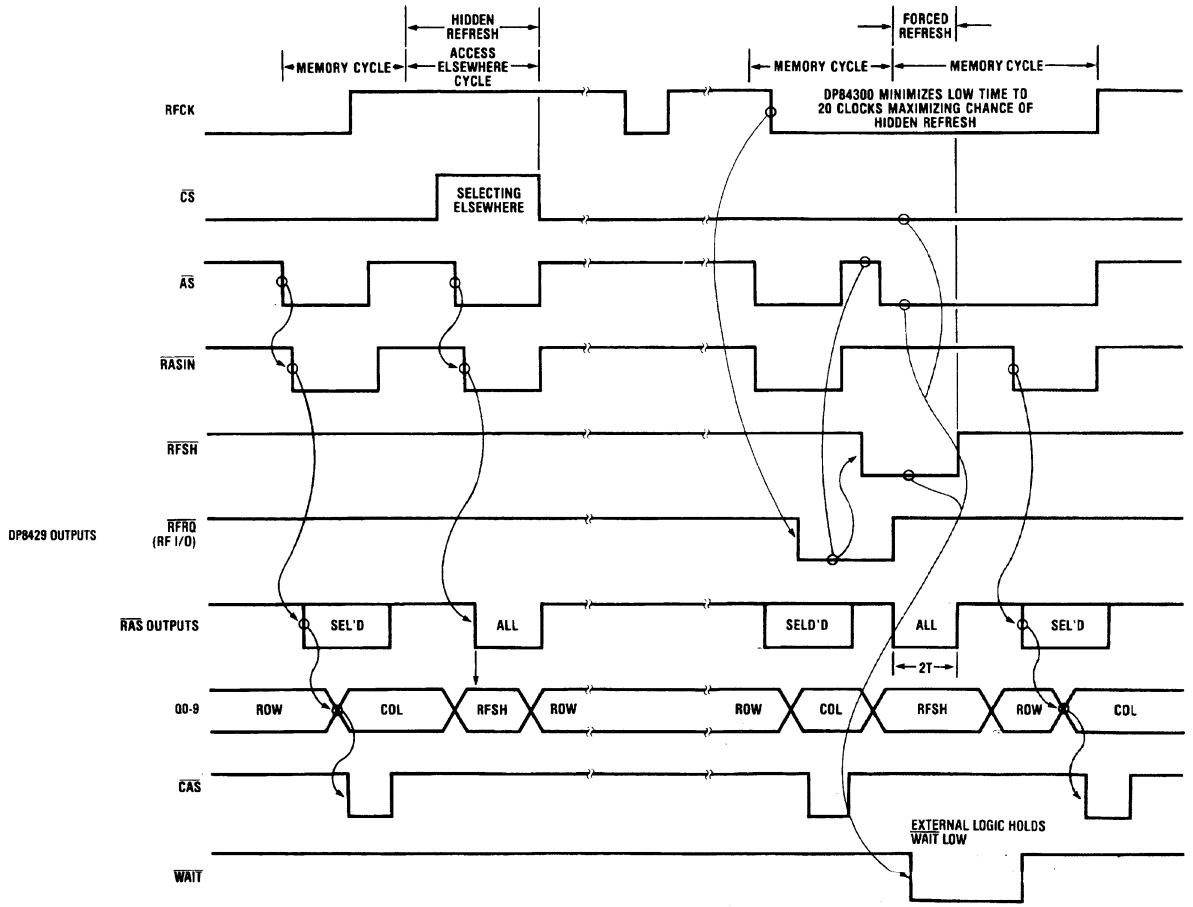
ADVANTAGES OF DP8429 OVER A DISCRETE DYNAMIC RAM CONTROLLER

1) The DP8429 system solution takes up much less board space because everything is on one chip (latches, refresh counter, control logic, multiplexers, drivers, and internal delay lines).
2) Less effort is needed to design a memory system. The DP8429 has automatic modes (1 and 5) which require a minimum of external control logic. Also programmable array logic devices (PALs) have been designed which allow an easy interface to most popular microprocessors (Motorola 68000 family, National Semiconductor 32032 family, Intel 8086 family, and the Zilog Z8000 family).
3) Less skew in memory timing parameters because all critical components are on one chip (many discrete drivers specify a minimum on-chip skew under worst-case conditions, but this cannot be used if more then one driver is needed, such as would be the case in driving a large dynamic RAM array).
4) Our switching characteristics give the designer the critical timing specifications based on TTL output levels (low = 0.8V, high = 2.4V) at a specified load capacitance. All timing parameters are specified on the DP8429:
   A) driving 88 DRAM's over a temperature range of 0–70 degrees centigrade (no extra drivers are needed).
   B) under worst-case driving conditions with all outputs switching simultaneously (most discrete drivers on the market specify worst-case conditions with only one output switching at a time; this is not a true worst-case condition!).



TL/F/8649–26

FIGURE 8. Connecting the DP8429 Between the 16-bit Microprocessor and Memory



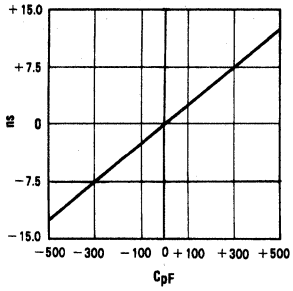
TL/F/8649–27

*T is microprocessor's clock period

FIGURE 9. DP8429 Auto Refresh, Access with $\overline{\text{WAIT}}$ States

Switching Characteristics

All A. C. parameters are specified with the equivalent load capacitances, including traces, of 88 DRAMs organized as 4 banks of 22 DRAMs each. Maximums are based on worst-case conditions including all outputs switching simultaneously. This, in many cases, results in the AC valves shown in the DP84XX DRAM controller data sheet being much looser than true worst case maximum AC delays. The system designer should estimate the DP8429 load in his/her application, and modify the appropriate A. C. parameters using the graph in *Figure 10*. Two example calculations are provided below.



TL/F/8649–28

FIGURE 10. Change in Propagation Delay relative to "true" (application) load minus AC specified data sheet load

Examples

1) A mode 4 user driving 2 banks of DRAM has the following loading conditions:

   $\overline{CAS}$ - 300 pF

   Q0–Q9 - 250 pF

   $\overline{RAS}$ - 150 pF

A.C. parameters should be adjusted in accordance with *Figure 10* and the specifications given for the 88 DRAM load as follows:

max t_{RPDL} = 20 ns − 0 ns = 20 ns (since $\overline{RAS}$ loading is the same as that which is spec'ed)

max t_{CPDL} = 32 ns − 7 ns = 25 ns

max t_{CCAS} = 46 ns − 7 ns = 39 ns

max t_{RCC} = 41 ns − 6 ns = 35 ns

min t_{RHA} is not significantly effected since it does not involve an output transition

Other parameters are adjusted in a similar manner.

2) A mode 5 user driving one bank of DRAM has the following loading conditions:

   $\overline{CAS}$ - 120 pF

   Q0–Q9 - 100 pF

   $\overline{RAS}$ - 120 pF

A. C. parameters should be adjusted as follows:

with RAHS = "1",

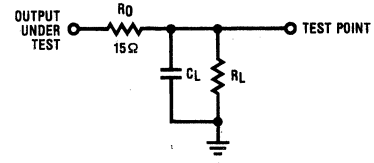
max t_{RICL} = 70 ns − 11 ns = 59 ns

max t_{RCDL} = 55 ns + 1 ns − 11 ns = 45 ns

(the + 1 ns is due to lighter $\overline{RAS}$ loading; the − 11 ns is due to lighter $\overline{CAS}$ loading)

min t_{RAH} = 15 ns + 1 ns = 16 ns

The additional 1 ns is due to the fact that the $\overline{RAS}$ line is driving less (switching faster) than the load to which the 15 ns spec applies. The row address will remain valid for about the same time irregardless of address loading since it is considered to be not valid at the beginning of its transition.



TL/F/8649–29

FIGURE 11. Output Load Circuit

Absolute Maximum Ratings (Note 1)

Specifications for Military/Aerospace products are not contained in this datasheet. Refer to the associated reliability electrical test specifications document.

Supply Voltage, V_{CC}	7.0V
Storage Temperature Range	−65°C to +150°C
Input Voltage	5.5V
Output Current	150 mA
Lead Temp. (Soldering, 10 seconds)	300°C

Operating Conditions

		Min	Max	Units
V_{CC}	Supply Voltage	4.50	5.50	V
T_A	Ambient Temperature	0	+70	°C

Electrical Characteristics V_{CC} = 5.0V ±10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Note 2)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_C	Input Clamp Voltage	V_{CC} = Min, I_C = −12 mA		−0.8	−1.2	V
I_{IH}	Input High Current for all Inputs	V_{IN} = 2.5V		2.0	100	μA
I_I RSI	Output Load Current for RFI/O	V_{IN} = 0.5V, Output high		−0.7	−1.5	mA
I_{IL1}	Input Low Current for all Inputs**	V_{IN} = 0.5V		−0.02	−0.25	mA
I_{IL2}	ADS, R/$\overline{C}$, $\overline{CS}$, M2, $\overline{RASIN}$	V_{IN} = 0.5V		−0.05	−0.5	mA
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_{OL1}	Output Low Voltage*	I_{OL} = 20 mA		0.3	0.5	V
V_{OL2}	Output Low Voltage for RFI/O	I_{OL} = 8 mA		0.3	0.5	V
V_{OH1}	Output High Voltage*	I_{OH} = −1 mA	2.4	3.5		V
V_{OH2}	Output High Voltage for RFI/O	I_{OH} = −100 μA	2.4	3.5		V
I_{1D}	Output High Drive Current*	V_{OUT} = 0.8V (Note 3)	−50	−200		mA
I_{0D}	Output Low Drive Current*	V_{OUT} = 2.4V (Note 3)	50	200		mA
I_{CC}	Supply Current	V_{CC} = Max		150	240	mA

*Except RFI/O

**Except RFI/O, ADS, R/$\overline{C}$, $\overline{CS}$, M2, $\overline{RASIN}$

Switching Characteristics: DP8428 and DP8429

V_{CC} = 5.0V ± 10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5), the output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q9, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11* for test load. Maximum propagation delays are specified with all outputs switching.**

**** Preliminary**

Symbol	Access Parameter	Condition	*CL		**All C_L = 50 pF		Units
			Min	Max	Min	Max	
t_{RICL0}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8428-80/29-80	57	97	42	85	ns
t_{RICL0}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8428-70/29-70	57	87	42	75	ns
t_{RICL1}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8428-80/29-80	48	80	35	68	ns
t_{RICL1}	$\overline{RASIN}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8428-70/29-70	48	70	35	58	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ High Delay	*Figure 6*		37			ns
t_{RCDL0}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8428-80/29-80	43	80			ns
t_{RCDL0}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 0)	*Figure 6* DP8428-70/29-70	43	72			ns

Switching Characteristics: DP8428 and DP8429 (Continued)

V_{CC} = 5.0V ± 10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5), the output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q9, C_L = 500 pF; $\overline{RAS}$0-$\overline{RAS}$3, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11* for test load. Maximum propagation delays are specified with all outputs switching.**

**** Preliminary**

Symbol	Access Parameter	Condition	*CL		**All C_L = 50 pF		Units
			Min	Max	Min	Max	
t_{RCDL1}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8428-80/29-80	34	63			ns
t_{RCDL1}	$\overline{RAS}$ to $\overline{CAS}$ Low Delay (RAHS = 1)	*Figure 6* DP8428-70/29-70	34	55			ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ High Delay	*Figure 6*		22			ns
t_{RAH0}	Row Address Hold Time (RAHS = 0, Mode 5)	*Figure 6*	25		25		ns
t_{RAH1}	Row Address Hold Time (RAHS = 1, Mode 5)	*Figure 6*	15		15		ns
t_{ASC}	Column Address Set-up Time (Mode 5)	*Figure 6*	0		0		ns
t_{RCV0}	$\overline{RASIN}$ to Column Address Valid (RAHS = 0, Mode 5)	*Figure 6* DP8428-80/29-80		94			ns
t_{RCV0}	$\overline{RASIN}$ to Column Address Valid (RAHS = 0, Mode 5)	*Figure 6* DP8428-70/29-70		85			ns
t_{RCV1}	$\overline{RASIN}$ to Column Address Valid (RAHS = 1, Mode 5)	*Figure 6* DP8428-80/29-80		76			ns
t_{RCV1}	$\overline{RASIN}$ to Column Address Valid (RAHS = 1, Mode 5)	*Figure 6* DP8428-70/29-70		68			ns
t_{RPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Low Delay	*Figures 5a, 5b, 6*		21		18	ns
t_{RPDH}	$\overline{RASIN}$ to $\overline{RAS}$ High Delay	*Figures 5a, 5b, 6*		20		17	ns
t_{ASRL}	Address Set-up to $\overline{RASIN}$ low	*Figures 5a, 5b, 6*	13				ns
t_{APD}	Address Input to Output Delay	*Figures 5a, 5b, 6*		36		25	ns
t_{SPD}	Address Strobe High to Address Output Valid	*Figures 5a, 5b*		48			ns
t_{ASA}	Address Set-up Time to ADS	*Figures 5a, 5b, 6*	5				ns
t_{AHA}	Address Hold Time from ADS	*Figures 5a, 5b, 6*	10				ns
t_{ADS}	Address Strobe Pulse Width	*Figures 5a, 5b, 6*	26				ns
t_{WPD}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	*Figure 5b*		28			ns
t_{CPDL}	$\overline{CASIN}$ to $\overline{CAS}$ Low Delay (R/$\overline{C}$ low, Mode 4)	*Figure 5b*	21	32			ns
t_{CPDH}	$\overline{CASIN}$ to $\overline{CAS}$ High Delay (R/$\overline{C}$ low, Mode 4)	*Figure 5b*	16	33			ns
t_{CPdif}	t_{CPDL} - t_{CPDH}	See Mode 4 Description		11			ns
t_{RCC}	Column Select to Column Address Valid	*Figure 5a*		41			ns
t_{RCR}	Row Select to Row Address Valid	*Figures 5a, 5b*		45			ns
t_{RHA}	Row Address Held from Column Select	*Figure 5a*	7				ns
t_{CCAS}	R/$\overline{C}$ Low to $\overline{CAS}$ Low Delay ($\overline{CASIN}$ Low, Mode 4)	*Figure 5a* DP8428-80/29-80		50			ns
t_{CCAS}	R/$\overline{C}$ Low to $\overline{CAS}$ Low Delay ($\overline{CASIN}$ Low, Mode 4)	*Figure 5a* DP8428-70/29-70		46			ns
t_{DIF1}	Maximum (t_{RPDL} - t_{RHA})	See Mode 4 Description		7			ns
t_{DIF2}	Maximum (t_{RCC} - t_{CPDL})			13			ns

Switching Characteristics: DP8428 and DP8429 (Continued)

V_{CC} = 5.0V ± 10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q9, C_L = 500 pF; $\overline{RAS}$0-$\overline{RAS}$3, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11* for test load. Maximum propagation delays are specified with all outputs switching.**

****Preliminary**

Symbol	Refresh Parameter	Condition	*CL		**All C_L = 50 pF		Units
			Min	Max	Min	Max	
t_{RC}	Refresh Cycle Period	*Figure 2a*	100				ns
$t_{RASINL,H}$	Pulse Width of $\overline{RASIN}$ during Refresh	*Figure 2a*	50				ns
t_{RFPDL0}	$\overline{RASIN}$ to $\overline{RAS}$ Low Delay during Refresh (Mode 0)	*Figure 2a*		28			ns
t_{RFPDL5}	$\overline{RASIN}$ to $\overline{RAS}$ Low Delay during Hidden Refresh	*Figure 7*		38			ns
t_{RFPDH0}	$\overline{RASIN}$ to $\overline{RAS}$ High Delay during Refresh (Mode 0)	*Figure 2a*		35			ns
t_{RFPDH5}	$\overline{RASIN}$ to $\overline{RAS}$ High Delay during Hidden Refresh	*Figure 7*		44			ns
t_{RFLCT}	$\overline{RFSH}$ Low to Counter Address Valid	*Figures 2a, 3* $\overline{CS}$ = X		38			ns
t_{RFLRL}	$\overline{RFSH}$ Low Set-up to $\overline{RASIN}$ Low (Mode 0), to get Minimum t_{ASR} = 0	*Figure 2a*	12				ns
t_{RFHRL}	$\overline{RFSH}$ High Setup to Access $\overline{RASIN}$ Low	*Figure 3*	25				ns
t_{RFHRV}	$\overline{RFSH}$ High to Row Address Valid	*Figure 3*		43			ns
t_{ROHNC}	$\overline{RAS}$ High to New Count Valid	*Figure 2a*		42			ns
t_{RST}	Counter Reset Pulse Width	*Figure 2a*	46				ns
t_{CTL}	RFI/O Low to Counter Outputs All Low	*Figure 2a*		80			ns
$t_{RFCKL,H}$	Minimum Pulse Width of RFCK	*Figure 7*	100				ns
T	Period of $\overline{RAS}$ Generator Clock	*Figure 3*	30				ns
t_{RGCKL}	Minimum Pulse Width Low of RGCK	*Figure 3*	15				ns
t_{RGCKH}	Minimum Pulse Width High of RGCK	*Figure 3*	15				ns
t_{FRQL}	RFCK Low to Forced $\overline{RFRQ}$ (RFI/O) Low	*Figure 3* C_L = 50 pF RL = 35k		66			ns
t_{FRQH}	RGCK Low to Forced $\overline{RFRQ}$ High	*Figure 3* C_L = 50 pF RL = 35k		55			ns

Switching Characteristics: DP8428 and DP8429 (Continued)

V_{CC} = 5.0V ± 10%, 0°C ≤ T_A ≤ 70°C unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs, including trace capacitance.

*** These values are Q0-Q9, C_L = 500 pF; $\overline{RAS0}$–$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF; RL = 500Ω unless otherwise noted. See *Figure 11* for test load. Maximum propagation delays are specified with all outputs switching.**

****Preliminary**

Symbol	Refresh Parameter	Condition	*CL		**All C_L = 50 pF		Units
			Min	Max	Min	Max	
t_{RGRL}	RGCK Low to $\overline{RAS}$ Low	*Figure 3*	21	41			ns
t_{RGRH}	RGCK Low to $\overline{RAS}$ High	*Figure 3*	23	48			ns
t_{RQHRF}	$\overline{RFSH}$ Hold Time from RGCK	*Figure 3*	2T				ns
t_{RFRH}	$\overline{RFSH}$ High to $\overline{RAS}$ High (Ending Forced Refresh early)	(See Mode 1 Description)		42			ns
t_{RFSRG}	$\overline{RFSH}$ Low Set-up to RGCK Low (Mode 1)	(See Mode 1 Description) *Figure 3*	12				ns
t_{CSHR}	$\overline{CS}$ High to $\overline{RASIN}$ Low for Hidden Refresh	*Figure 7*	10				ns
t_{CSRL1} for DP8429	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Mode 5 with Auto Refresh Mode)	*Figure 3*	34				ns
t_{CSRL1} for DP8428	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Mode 5 with Auto Refresh Mode)	*Figure 3*	5				ns
t_{CSRL0}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low (Using Modes 4 or 5 with externally controlled Refresh)	(See Mode 5 Description)	5				ns
t_{RKRL}	RFCK High to $\overline{RASIN}$ low for hidden Refresh		50				ns

Input Capacitance T_A = 25°C (Note 2)

Symbol	Parameter	Condition	Min	Typ	Max	Units
C_{IN}	Input Capacitance ADS, R/$\overline{C}$, $\overline{CS}$, M2, $\overline{RASIN}$			8		pF
C_{IN}	Input Capacitance All Other Inputs			5		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for T_A = 25°C and V_{CC} = 5.0V.

Note 3: This test is provided as a monitor of Driver output source and sink current capability. Caution should be exercised in testing this parameter. In testing these parameters, a 15Ω resistor should be placed in series with each output under test. One output should be tested at a time and test time should not exceed 1 second.

Note 4: Input pulse 0V to 3.0V, t_R = t_F = 2.5 ns, f = 2.5 MHz, t_{PW} = 200 ns. Input reference point on AC measurements is 1.5V Output reference points are 2.4V for High and 0.8V for Low.

Note 5: The load capacitance on RF I/O should not exceed 50 pF.

Section 8
Development Tools

Section 8 Contents

SYS32/20 Development System 8-3
VR32 Target/Development System 8-4
ISE32 NS32032 In-System Emulator 8-8
ISE16 NS32016 In-System Emulator 8-17
DB32000 Development Board 8-28
DB32016 Development Board 8-33
ICM-3332 Integrated Computer Module 8-39
ICM-3216 Integrated Computer Module 8-46

PRELIMINARY
May 1986

SYS32/20 Development System

- High Performance, 10-MHz, no-wait state, NS32032 add-in card for IBM-XT/AT or Compatible. (Note 1)
- 2-Mbyte RAM Configuration (not expandable)
- 4-Mbyte RAM Configuration
- XT or AT Compatible (360-Kbyte or 1.2-Mbyte Floppies)
- Operating System derived from AT&T UNIX™ System V.2 (on-line manual pages) (Note 1)
- Series 32000® GNX Language Tools (Note 1)
- Optional Compilers (Note 2)
- Optional System V.2 Documentation (Note 2)
- Optional BSD Utilities (Note 2)
- Optional Tools for Documenters (TFD) (Note 2)
- Optional Driver for Ethernet (Note 2)
- Easy Installation
- Adapts to wide variety of Winchester disk sizes.

Note 1: Minimum configuration for development system.

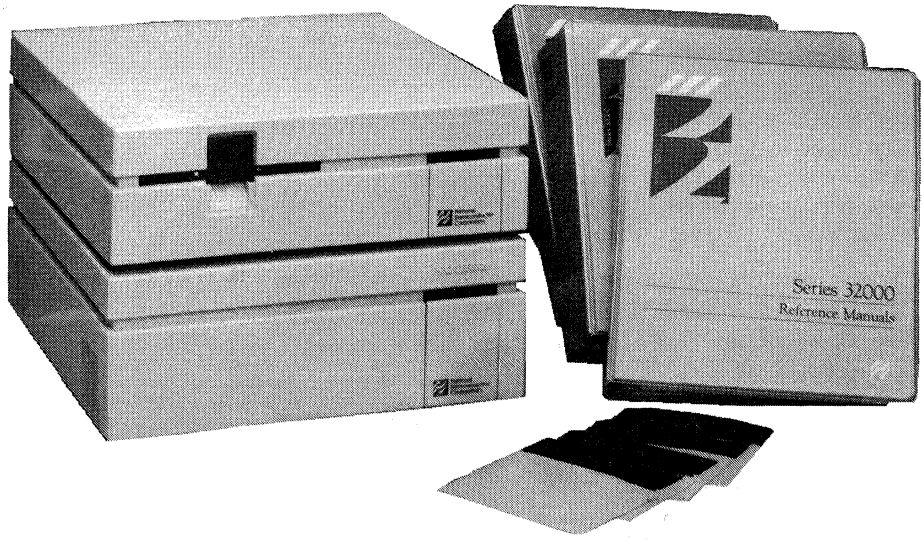
Note 2: Requires purchase of add-in card with operating system and GNX tools.

MINIMUM PC CONFIGURATION

- 20-Mbyte hard disk (30-Mbyte or larger is recommended)
- Slots - 2-Mbyte Configuration requires 1½ long slot, 4-Mbyte Configuration (depending on placement) may require 2 long slots.
- XT, AT or Compatible
- PC-DOS 3.1 or later
- 512-Kbyte RAM

National Semiconductor Corp.

VR32™ Target/Development System For The Series 32000® Microprocessor Family



TL/R/8521–1

- **Series 32000 Microprocessor Family based**
- **Time-shared support for two users**
- **System V/Series 32000 UNIX™ operating system**
- **Demand Paged Virtual Memory support**
- **Easy to use, proven programming environment**
- **1 Megabyte RAM, user expandable**
- **Reconfigurable hardware/software supports user customized systems**
- **40 Megabyte hard disk**
- **Industry standard MULTIBUS® based with three user available slots**
- **C and FORTRAN high level languages supported**
- **Supports emulation of Series 32000 Microprocessor Family**
- **1 megabyte Floppy Disk**
- **Optional PASCAL compiler**
- **Optional Streamer Tape**

Product Overview

The VR32 Target/Development System (T/DS) is a single-user development environment that provides software and hardware tools for the development of applications using National Semiconductor's Series 32000 Microprocessor Family components. Applications that require customized hardware and software environments can be satisfied efficiently and economically with the user-configurable features of the VR32 T/DS.

The VR32 T/DS includes two modules: the processor module which is Multibus based and contains the main CPU board, 1 Megabyte of system memory, floppy/hard disk controller board, power supply, and up to three additional user slots; and the disk module which contains the 40 Megabyte hard disk and 1 Megabyte floppy disk drives. An optional 20 Megabyte streamer tape cartridge module can also be added to provide either backup or program off-loading for the system. Software support is included for the optional streamer tape module. The terminal for use with the system is user-supplied.

Product Overview (Continued)

The VR32 T/DS uses National Semiconductor's reconfigurable System V/Series 32000 operating system, a port of AT&T's validated UNIX System V Release 2.0 Version 2 for the Series 32000 Microprocessor Family. Its reconfigurability features provide the system designer with the ability to add input/output driver software without the requirement of an AT&T UNIX source license. The enhancements made to the System V/Series 32000 operating system fully utilize the advanced 32-bit architecture of the Series 32000 Microprocessor Family.

Hardware Description

Processor Module

The processor module of the VR32 houses the main CPU board, the system memory, disk-tape controller board, power supply, and an industry-standard Multibus card cage (the disk drives are contained in a separate module, as is the optional streamer tape cartridge drive). The card cage has three slots available for any additional Multibus-compatible boards the user may wish to install into the system. The power supply provides power to the processor module.

Main CPU Board

The VR32's main processor board, which occupies one of the slots, utilizes National Semiconductor's Series 32000 Microprocessor Family. This chip set includes the NS32016 Central Processor Unit (CPU), NS32082 Memory Management Unit (MMU), NS32201 Timing Control Unit (TCU), NS32081 Floating Point Unit (FPU), and NS32202 Interrupt Control Unit (ICU). 512 Kilobytes of dual-ported dynamic random access memory (RAM) reside on-board. An additional 512 kilobytes of system memory reside on a separate board. The standard I/O interfaces provided include a parallel interface port with 24 parallel I/O lines which can be used with any Centronics-compatible printer interface. Also included are two serial RS232 interface ports, each capable of communication at 9600 baud. One serial port is connected to a user terminal for communication to the VR32 T/DS. The second serial port can be used with such development tools as the ISE32™ (NS32032 In-System Emulator) or even to another VR32 T/DS.

Disk-Tape Controller

The disk-tape controller board which occupies a second Multibus slot handles all communications between the processor board and mass storage devices (disks, tape). Commands for specific functions are passed from the processor board to the controller board over the system bus. Results (status and data) from the mass storage devices are then returned over the system bus back to the processor board, to be evaluated by the NS32016 CPU or stored in system memory. The controller board provides the interface for the VR32's 40 Megabyte Winchester disk drive, 1 Megabyte 5¼" floppy disk drive, and optional streamer tape cartridge drive module.

Disk Module

The disk module contains the two disk drives that come standard with the VR32 T/DS System. A 40 Megabyte Winchester hard disk drive and a 1 Megabyte 5¼" floppy disk drive are installed in a chassis that can be stacked on top of the processor module. A power supply provides the required voltages and currents for the disk drives. Cables are included to connect the disk module to the processor module, creating a complete computer system. The terminal for use with the system is user-supplied.

Hardware Support

Terminals: Support is provided for a wide variety of terminals via the "terminfo" facility in the System V/Series 32000 operating system. These include the DEC VT100, Televideo, and Hazeltine families.

Printer Interface: Both parallel and serial printer support is provided for a variety of printers including the Centronics 700 and 300 series. The parallel interface is configured for standard Centronics-compatible devices.

PROM Programming: Support is provided for the Data I/O System 19.

Emulation: The ISE™ products for the Series 32000 Microprocessor Family are fully supported. These include the ISE16™, In-System Emulator for the NS32016 and the ISE32, In-System Emulator for the NS32032.

Software Description

System V/Series 32000 Operating System

The System V/Series 32000 operating system utilized on the VR32 T/DS is a port of AT&T's UNIX System V (Release 2.0, Version 2) operating system. The features provided by System V/Series 32000 are an advanced, proven programming environment to fully support the Series 32000 Microprocessor Family, including Demand-Paged Virtual Memory.

The System V/Series 32000 operating system is a general purpose, multi-tasking, interactive operating system designed to make the programmer's and documenter's computing environment simple, efficient, and productive. The System V/Series 32000 includes all the tools to compile, assemble, link, and download object code to any Series 32000 based product. Object files created conform to a superset of the AT&T common object file format (COFF), jointly defined by AT&T and National Semiconductor, and specifically intended to fully support the advanced features of the Series 32000 architecture.

Features of the System V/Series 32000 operating system include:

— Demand-Paged Virtual Memory

— User-configurable I/O environment

Software Description (Continued)

- — Hierarchical, tree-structured file system
- — Flexible command language
- — Ability to execute sequential, asynchronous, and background processes
- — Powerful text editors
- — File and Record locking
- — Support for high-level languages including C, FORTRAN, and an optional Pascal Compiler
- — Series 32000 assembler
- — Inter-system communications facilities
- — High level language symbolic debugger

User-Configurable I/O System

The System V/Series 32000 operating system implemented on the VR32 allows the I/O system to be redefined as required by the user's application. The operating system has been designed with I/O interface components available to the user at the binary code level. Additional hardware in the form of Multibus boards can be added to the system as the application requires (see **Hardware Description**). The software driver generated by the user can then be linked into the system and recognized by the operating system as a valid system I/O driver. This eliminates the need for an AT&T UNIX source license and enables the VR32 to operate as both a development system and application-specific target system.

Sample I/O Drivers

Several I/O driver programs used in the VR32 T/DS are provided in source form to assist in the development of additional I/O drivers. Program source code is provided for the following drivers:

- — Console driver
- — Disk/Tape driver
- — Multiport asynchronous RS232 communication driver
- — Parallel printer driver

File System

The file system of the operating system consists of a highly uniform set directories and files in a tree-like hierarchical structure which are addressable to one billion bytes.

Command Language

User communication with the operating system is normally carried out with the aid of a program called the shell. A shell is both a command language interpreter and a programming language that provides an interface to the operating system.

Document Preparation

System V/Series 32000 has many text processing and document preparation facilities. Included are powerful full-screen editor, text formatters, text processing macro packages, special processors for mathematical expressions and tabular material, and numerous supporting utilities.

Source Code Control System

The Source Code Control System (SCCS) in the System V/Series 32000 operating system is an integrated set of commands designed to aid software development projects or document preparation by controlling changes to source code or files of text. SCCS provides facilities for storing, updating, and retrieving all versions of source code modules or documents, and for recording the time, author, and reason for change.

File Transfer

"uucp" (UNIX to UNIX copy) is a series of programs designed to permit communication between systems running under the other UNIX operating systems either by dial-up modem or hard-wired communication lines.

Physical Specification

The standard VR32 T/DS consists of the Processor Module, Floppy/Hard Disk Module, the required interconnect cables, and supporting manuals.

Processor Module

This is a horizontal desk unit with front mounted controls and indicators, and rear mounted I/O connections.

Height —	5.23 inches (13.3 cm)
Width —	14.08 inches (35.6 cm)
Depth —	16.25 inches (41.3 cm)
Color —	Beige
Module Weight —	21 pounds (10 kg)
Shipping Weight —	29 pounds (13 kg)

Floppy/Hard Disk

This, like the Processor Module, is also a horizontal desk unit with front mounted controls and indicators, and rear mounted I/O connections.

Height —	4.4 inches (11.2 cm)
Width —	14.08 inches (35.6 cm)
Depth —	16.25 inches (41.3 cm)
Color —	Beige
Module Weight —	21 pounds (10 kg)
Shipping Weight —	29 pounds (13 kg)

Environmental

	Operating	Non-Operating
Temperature	41°F to 104°F 5°C to 40°C	−40°F to 151°F −40°C to 66°C
Relative Humidity		
Max Wet Bulb	8% to 95% (non-condensing)	
Max Altitude	8,000 feet	30,000 feet

Physical Specification (Continued)

Electrical

Processor Module

FCC: Class A

AC Voltage: 90–132 VAC; 47–63 Hz
180–253 VAC; 47–63 Hz

Average Power Consumed: 310 Watts

Floppy/Hard Disk Module

FCC: Same as Processor Module

AC Voltage: Same as Processor Module

Average Power Consumed: 110 W

Maximum Surge Power: 140 W

Order Information

Systems

NSS-VR32-1001	Complete system including processor module, disk module, System V/Series 32000 operating system, cables, and manuals.
NSS-VR32-1001E	Same as above configured for European power.

Peripherals

NSS-VR32-2001	20 Megabyte streamer tape cartridge module.
NSS-VR32-2001E	Same as above configured for European power.
NSS-VR32-3001	Pascal compiler.
NSS-ISE16	ISE16; In-System Emulator for NS32016.
NSS-ISE16E	Same as above configured for European power.
NSS-ISE32	ISE32; In-System Emulator for NS32032.
NSS-ISE32E	Same as above configured for European power.

Documentation

NSS-VR32-2100	Set of system manuals, consisting of System V/Series 32000 software manuals, systems reference manuals, and ISE and symbolic debug manuals.
NSS-VR32-2102	ISE16 and System V/Series 32000 symbolic debug manuals.
NSS-VR32-2103	ISE32 and System V/Series 32000 symbolic debug manuals.

National Semiconductor Corp.

ISE32™ NS32032 In-System Emulator

TL/R/8522–1

- **Operation up to 10 MHz***
- **Emulation of NS32032 Central Processing Unit, NS32082 Memory Management Unit, NS32201 Timing Control Unit**
- **Host resident debuggers**
- **Generalized event driven system**
- **Memory mapping, up to 128 kbytes**
- **Read/write protection of 4 kbyte memory blocks**
- **Program flow tracing, up to 1023 non-sequential fetches**
- **Complete bus activity trace**
- **Qualified tracing**
- **Pre-, post-, or center-triggering on trace**
- **Two 32-bit execution counters**
- **Supports Memory Management Unit functions**
- **Supported under various host systems and operating systems**
- **Hierarchical on-line help facility**
- **Self-diagnostic**

*Refer to ISE speed consideration section.

Description

The NS32032 In-System Emulator (ISE32) is a powerful tool for both hardware and software development of NS32032 microprocessor-based products.

The ISE32 emulates the NS32032 Central Processing Unit (CPU), the NS32201 Timing Control Unit (TCU) and NS32082 Memory Management Unit (MMU). NS32082 MMU emulation can be disabled by a switch setting. The ISE32 allows users to test and debug both hardware and software in their own hardware environment.

The ISE32 is a complete unit, including an internal clock oscillator that generates a choice of three clock signals: 10 MHz, 5 MHz, and 2.5 MHz; and 128 kbytes of dedicated user's ISE™ memory. With the ISE32, users can easily stop emulation and examine the contents of CPU registers, slave processor registers, and memory.

The ISE32 consists of the ISE hardware, the ISE firmware monitor, and RS232 cables. A host-dependent debugger software program is available as part of the appropriate Series 32000® software support package.

Each of the Series 32000 software support packages include software tools to produce code compatible with the debugger software. Refer to the section "Required User-Supplied Equipment".

Hardware Description

The ISE32 hardware is housed in three enclosures: the ISE Support Box; the Emulator Pod; and the TTL Status Pod. *Figure 1* is a block diagram of ISE32 hardware.

The ISE Support Box is the largest enclosure. It contains the emulation support circuits for trace, breakpoints, and mapped memory; as well as the hardware for the RS232 serial ports, which are used to communicate with the host and the user's terminal. It also houses the power supplies and the ISE32 control switches and indicators. *Figure 2* shows the location of the ISE32 control switches and indicators. Table I lists the functions of each switch and LED.

The Emulator Pod contains the NS32032 CPU, NS32082 MMU, and NS32201 TCU required for target system emulation. It also contains the ISE Monitor firmware.

The Emulator Pod connects to the ISE Support Box via a four-foot flat cable assembly. Connections to the target system are made via three one-foot target cables. One target cable is provided for each member of the Series 32000 chip set (CPU, MMU, and TCU).

The Status Pod is the smallest enclosure. It provides TTL-compatible input and output signals for use during ISE operation. The Status Pod has ten leads and three binder posts that can be connected to either the target system or test equipment such as logic analyzers or oscilloscopes. Table II lists the function of each lead and post of the Status Pod. The Status Pod connects to the ISE Support Box via a six-foot cable.

ISE32 Software Overview

The ISE32 software consists of the ISE firmware monitor, which resides in PROMs in the Emulator Pod, and the ISE Debugger, which runs on the host system.

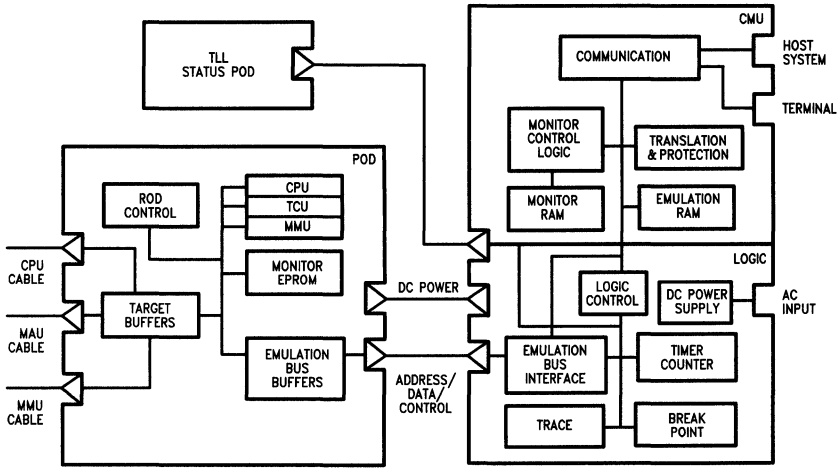
When the ISE32 unit is not running an emulation program, it is running a program called the ISE monitor.

The monitor communicates with the ISE Debugger and provides a command protocol that allows the host complete control of the ISE32 hardware.

The ISE Debugger translates commands entered on the host system from a terminal, into low-level instructions that the ISE monitor uses to drive the hardware. The ISE Debugger also translates and sends ISE responses to the user via the terminal. All ISE monitor operation is transparent to the user.

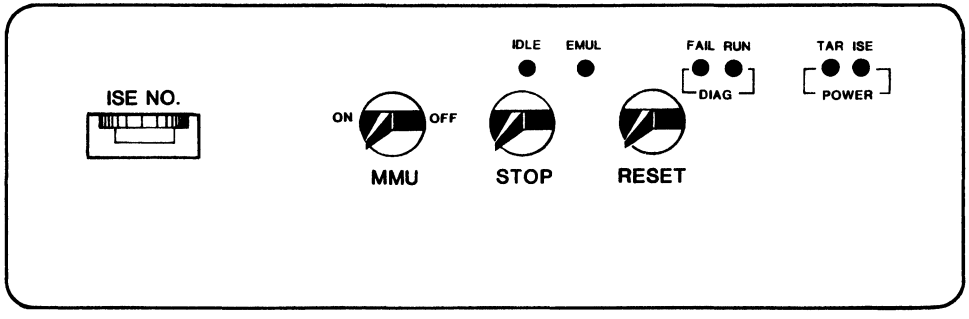
The ISE32 Debugger

The ISE32 Debugger is user compatible with the standard non-ISE Series 32000 Debugger. Compatibility



TL/R/8522–4

FIGURE 1. ISE32 Block Diagram



TL/R/8522–3

FIGURE 2. ISE32 Controls and Indicators

minimizes the user's learning time of the various development tools. The ISE32 Debugger fully supports all the powerful debugging and emulation facilities provided by the ISE32 hardware, and supplements these features with a very powerful software-based program debugging environment.

The basic debugging features of the ISE32 are as follows:

(1) Supports both high-level and assembly languages*.

(2) Breakpoints can be set at the source code level, even when using high-level languages.*

(3) Supports symbolic debugging; variables can be referenced by their source code names.*

(4) Certain procedure parameters and variables are easily displayed.

(5) Structured data types and pointers are easily displayed.

(6) Supports both command and history files.

(7) Memory can be displayed in many different ways, including a disassembly mode displaying memory as NS32032 instructions.

(8) Supports all the emulation and debug facilities provided by the ISE32 hardware.

*Depends on host environment and language.

Modes of Operation

ISE32 can be set-up to operate in either stand-aside mode or transparent mode.

In stand-aside mode, one serial RS232 link from the host system is connected to the ISE32 while another serial RS232 from the host system is connected to the user's terminal. In this configuration, any of the host's users can access the ISE32.

In transparent mode, one serial RS232 link from the host system is connected to one serial port on the ISE32 while the user terminal is connected to a second serial port on the ISE32. In this configuration, only one serial port is required from the host system. In non-emulation mode, the ISE32 is transparent to the user, allowing normal communication between the user and the host system.

ISE32 Operation

Human Interface

ISE32 is easy to learn and easy to use. The software includes a complete on-line help facility. Invoking the

TABLE I. ISE32 Control and Indicator Functions

Control/Indicator	Function
ISE NO. Switch	Set to 0; other positions reserved.
MMU Switch	When ON, ISE32 enables MMU operation.
STOP Switch	Interrupts emulations, restores control to the ISE32 monitor.
RESET Switch	Resets the ISE32 hardware.
IDLE	Warning that POD CPU is in a wait state. (Time out)
EMUL	Indicates that ISE32 is executing the user's program.
FAIL	Warning that diagnostics have failed.
RUN	Indicates that ISE32 diagnostics are running.
TAR	Indicates that target power is on.
ISE	Indicates that ISE32 is on.

TABLE II. Status Pod Signal Description

Status Pod Label	ISE Function
Leads	
1-WHT-USRCLK-U	Not Used
2-BLK-GND	Common Ground
3-BRN-EXT0-U	EXT0 (external input 0)
4-RED-EXT1	EXT1 (external input 1)
5-ORN-EXT2	EXT2 (external input 2)
6-YEL-EXT3	EXT3 (external input 3)
7-GRN-EXT4	EXT4 (external input 4)
8-BLU-EXT5	EXT5 (external input 5)
9-VIO-EXT6	EXT6 (external input 6)
10-GRY-EXT7	EXT7 (external input 7)
11-WHT-USEBRK/U	IS (input sync)
Posts	
TBRUN	Not Used
BK SYNCH/-U	Output Sync
TR SYNCH/-U	Not Used
GND	Common Ground
TSYNC31/	Not Used
TSYNC21	Not Used
GND	Common Ground

"HELP" command gives a summary of all ISE32 commands, an individual command, or an individual commands parameters. This feature helps the user get his work done quickly with less frustration.

Emulation

The ISE32 unit has its own CPU, MMU, and TCU components. These components are connected to the target system via cables. These components perform the same functions, with close to the same timing characteristics as they would if mounted in the target system.* The ISE32 does not require wait states for operation.

Emulation memory, resident in the ISE32, can be used instead of target system memory. This feature is implemented by the mapping capabilities. With this feature, the ISE32 can run and debug programs without a working target system. User target memory from the entire address space of the CPU or MMU (whether it exists or not) can be mapped onto the ISE32 emulation memory in 4 kbyte blocks. The total amount of mapped memory cannot exceed 32 4 kbyte blocks (128 kbytes).

Associated with the emulation memory mapping scheme is a capability for read/write protection. Any 4 kbyte block within the address space of the CPU or MMU can be protected.

Generalized Events

To provide a versatile way of observing and controlling the significant state changes on the microprocessor, ISE32 allows the use and definition of "events". In general, a simple event is a breakpoint, a bus change, or a significant observation. An event can also be a logical combination of simple events (an Event-Expression).

Simple Event Definition

The simple events are:

— Breakpoints
— Latched Events
— Counter Done
— Status Pod Inputs
— Trace Done

Breakpoint Events

ISE32 provides four common breakpoint events, named A, B, C, and D. The breakpoint event can be used in two ways:

(1) Execution Breakpoint—occurs just prior to execution of an instruction at a specified address.
(2) Reference Breakpoint—occurs on a match when sampling:

— Address Bits
— Data Bits
— External Status Bits
— User/Supervisor Pin
— Byte Enable Pins
— Data Direction Pin
— Status Bits
— Interlock Bit
— Masked combinations of any of the above options.

Either virtual or physical addresses can be sampled. ISE32 also provides a range breakpoint event, R. The range breakpoint can be qualified by any of the above options within a specified address range.

Any breakpoint can cause emulation to stop immediately. Also, if used with the No Stop option, breakpoints can be combined with other events to cause a variety of action.

*Refer to ISE speed consideration section.

Event-Expressions

An event-expression is a Boolean expression made up of simple events, i.e., a logical combination of simple events. This allows the user to generate many different event combinations, tailored to system activity of particular interest to the user. These generalized events are used by many ISE32 commands such as stop, trace, event counting, etc. Event-expressions provide creative and flexible debugging procedures.

Event-expressions can be evaluated as either logically true or logically false. Valid logic operations for event expressions are: Negation (NOT), AND, and OR.

Stopping Execution on Events

A common debugging activity is to stop emulation on the occurrence of an event of interest. Stopping emulation puts ISE32 in the monitor mode so the user can examine and alter the state of the CPU, memory, and ISE32 functions. Emulation can be stopped on either simple events or event-expressions.

Flexible Tracing

ISE32 maintains a 1023-entry trace memory. Trace memory captures bus activity in one of two trace modes:

— Program Flow Trace
— Memory Bus Trace

Any combination of events can be used to qualify tracing. When enabled, tracing in either mode continues until a specified terminating event occurs. The actual end of tracing can be delayed after the terminating event by a count of 1 to 1023. This allows trace data to be captured before, after, or around the terminating event.

Program Flow Trace

The Program Flow Trace mode captures the CPU Program Counter address of 1023 non-sequential instructions. This mode also maintains a count of sequential instructions executed between each non-sequential instruction stored in the trace memory.

Memory Bus Trace

The Memory Bus Trace mode captures a summary of the following system parameters:

— Address bus contents

— Data bus contents
— CPU status (data transfer, non-sequential fetch, interrupt acknowledge, etc.)
— Time base counter contents
— PFS counter contents
— Status Pod external inputs
— States of the following CPU pins:
UNS—User/Not Supervisor
BE0–BE3—Byte Enable
DDIN—Data Direction In
NMI—Non-Maskable Interrupt
ILO—Interlock

Counters

The ISE32 contains two 32-bit counters with an overflow flag that may be used to count events, instruction cycles, memory cycles, or clock cycles. The counters may be programmed to start and stop counting on specific events. This permits counters to be used as timers to determine relative timing differences between various events. One use of this feature is to measure software or hardware performance. The counters may also be used to generate other ISE32 events upon completion of a count.

Event Trigger for External Test Equipment

ISE32 events can trigger external test equipment, such as oscilloscopes and logic analyzers. This test equipment can be used in conjunction with the ISE32's debugging features to solve system timing problems. The external trigger signal is available at the status pod output:

— BKSYNCH/-U (Output Sync)

Self-Test Diagnostics

At power-up, ISE32 runs a diagnostic program to verify ISE firmware integrity and proper hardware function.

ISE32 Timing Options

ISE32 includes the following timing options:

— Sampling time can be set to sample either virtual or physical addresses
— Status Pod external lines can be sampled at either data valid or address valid times
— The emulation clock frequency can be set to one of the following frequencies:
2.5 MHz
5.0 MHz
10.0 MHz
Target Board Frequency

ISE Speed Considerations

ISE32 utilizes standard, 10 MHz NS32032, NS32082 and NS32201 devices to perform control and emulation functions. When emulating, each device is connected to customer hardware via a target cable and associated cable transceivers. This arrangement delays the signal propagation between the Series 32000 components in the ISE32 POD and Series 32000 sockets in the customer hardware. These delays reduce timing margins in that hardware; i.e., combined propagation paths are lengthened by the ISE32 target cable and transceiver delays.

If sufficient timing margins are not restored, emulation may not be successful. In many cases, margin can be restored by reducing the emulation speed, lengthening the available time for signals to propagate. However, the exact speed reduction necessary to regain margins will depend on how Series 32000 components are used in the customer hardware. Tables III, IV and V list the combined cable and transceiver maximum propagation delay for each signal. It is the customer's responsibility to factor these delays with those of his own circuitry. In doing so, it can be determined whether ISE32 can reliably emulate with that circuitry.

Supported Configurations

This product is designed to work in most target systems configurations. However, certain design restrictions may apply. Refer to the ISE32 User's Manual for further information. (See "Documentation section")

Required User-Supplied Equipment

For use with SYS32™/GENIX™ Systems:

— Included with the GENIX Operating System Software Package.

For use with VR32/System V/Series 32000

— Included with the System V/32000 Operating System Software Package.

For use with VAX™/UNIX™ Systems:

— Valid DEC VAX-11™ configuration with available RS232 port.
— Berkeley UNIX 4.2 bsd Operating System.
— NSW-C-4VXR Series 32000 Cross Software Package.

For use with VAX/VMS™ Systems:

- Valid DEC VAX/11 configuration with available RS232 port.
- VMS™ Operating System, Version 4.2 or later.
- NSW-ASSEMB-9VMR or NSW-PASCAL-9VMR Series 32000 Cross Software Package.

Specifications

Environmental	Operating Temperature +10°C to +40°C Storage Temperature −20°C to +65°C	
Power	2.5A @ 115 VAC, 50/60 Hz, single phase 1.5A @ 220 VAC, 50/60 Hz, single phase. Approximately 1170 BTU.	
Physical		
ISE Support Box	Height:	5.8 in. (14.7 cm)
	Width:	18.5 in. (47.1 cm)
	Depth:	12.3 in. (31.2 cm)
Emulation Pod	Height:	2.1 in. (5.3 cm)
	Width:	9.3 in. (23.6 cm)
	Depth:	10.0 in. (25.4 cm)

Specifications (Continued)

TTL Status Pod
Height: 1.0 in. (2.5 cm)
Width: 3.125 in. (7.9 cm)
Depth: 6.125 in. (15.6 cm)

Cable Lengths
ISE Support Box to Emulation Pod: 4.0 ft. (1.22M)
ISE Support Box to TTL Status Pod: 6.0 ft. (1.83M)
Emulation Pod to Target Board: 1.0 ft. (0.30M)

Target Interface Electrical Characteristics— See Tables III through V.

Order Information

Complete ISE32 Units

NSS-ISE32	ISE32 (NS32032), 115 VAC
NSS-ISE32E	ISE32 (NS32032), 220 VAC

TABLE III. Electrical Characteristics for TCU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T_{pd}*
Outgoing Signals		I_{OH}	I_{OL}	
NTSO	74ALS244	15 mA	†48 mA	12.4 ns
CTTL	74ALS244	15 mA	†48 mA	12.4 ns
FCLK	74ALS244	15 mA	†48 mA	12.4 ns
NDBE	74ALS244	15 mA	†48 mA	12.4 ns
NRD	74ALS244	15 mA	†48 mA	12.4 ns
NWR	74ALS244	15 mA	†48 mA	12.4 ns
NRSTO	74ALS244	15 mA	†48 mA	12.4 ns
RDY	74ALS244	15 mA	†48 mA	12.4 ns
Incoming Signals		I_{IH}	I_{IL}	
NPER	74F244	20 μA	1.6 mA	7.9 ns
NCWAIT	74F244	20 μA	1.6 mA	7.9 ns
NWAIT1	74ALS244	20 μA	0.1 mA	12.4 ns
NWAIT2	74ALS244	20 μA	0.1 mA	12.4 ns
NWAIT4	74F244	20 μA	1.6 mA	14.5 ns
NWAIT8	74ALS244	20 μA	0.1 mA	12.4 ns
XCTL1	74F244	20 μA	1.6 mA	7.9 ns
NRWEN	74F244	20 μA	1.6 mA	7.9 ns
NRST1	74ALS244	20 μA	0.1 mA	12.4 ns

*Interface device, plus cable.
†For V_{CC} maintained between 4.75V and 5.25V.

TABLE IV. Electrical Characteristics for MMU Interface

Signal Name	Interface Device	Input And/Or Output Current				Propagation Delay Time T_{pd}*
		I_{OH}	I_{OL}	I_{IH}	I_{IL}	
BIDIRECTIONAL SIGNAL						
NPAV	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	11.4 ns
OUTGOING SIGNALS						
A24	74ALS244	15 mA	†48 mA	—	—	12.4 ns
MMUINT	74ALS244	15 mA	†48 mA	—	—	12.4 ns
NABT	74ALS244	15 mA	†48 mA	—	—	12.4 ns
NFLT	74ALS244	15 mA	†48 mA	—	—	12.4 ns
NHLDAO	74ALS244	15 mA	†48 mA	—	—	12.4 ns
INCOMING SIGNALS						
NHOLD	74LS126	—	—	20 μA	0.4 mA	19.4 ns

*Interface device, plus cable.
†For V_{CC} maintained between 4.75V and 5.25V.

TABLE V. Electrical Characteristics for CPU Interface

Signal Name	Interface Device	Input And/Or Output Current				Propagation Delay Time
		I_{OH}	I_{OL}	I_{IH}	I_{IL}	T_{pd}*
BIDIRECTIONAL SIGNAL						
NSPC	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD00	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD01	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD02	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD03	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD04	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD05	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD06	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD07	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD08	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD09	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD10	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD11	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD12	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD13	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD14	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD15	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD16	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD17	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD18	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD19	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD20	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD21	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD22	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
AD23	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D24	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D25	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D26	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D27	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D28	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D29	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D30	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
D31	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
NNDIN	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
NADS	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
NBE0	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
NBE1	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
NBE2	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns
NBE3	74ALS245	15 mA	†48 mA	20 μA	0.1 mA	12 ns

*Interface device, plus cable.
†For V_{CC} maintained between 4.75V and 5.25V.

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time
Outgoing signals		I_{OH}	I_{OL}	T_{pd}*
NILO	74ALS244	15 mA	†48 mA	13.0 ns
ST0	74ALS244	15 mA	†48 mA	13.0 ns
ST1	74ALS244	15 mA	†48 mA	13.0 ns
ST2	74ALS244	15 mA	†48 mA	13.0 ns
ST3	74ALS244	15 mA	†48 mA	13.0 ns
NPFS	74ALS244	15 mA	†48 mA	13.0 ns
UNS	74ALS244	15 mA	†48 mA	13.0 ns
BB	74ALS244	15 mA	†48 mA	13.0 ns
NDS	74ALS244	15 mA	†48 mA	13.0 ns
NBRO	74F244	15 mA	†48 mA	8.5 ns
NHLDA	74ALS244	15 mA	†48 mA	13.0 ns
Incoming signals		I_{IH}	I_{IL}	
TGTPC	—	—	—	—
NINTC	74ALS244	20 mA	0.1 mA	13.0 ns
NMI	74ALS244	20 mA	0.1 mA	19.6 ns
NBRI	74F244	20 mA	1.6 mA	8.5 ns
NHOLDC	74ALS244	20 mA	0.1 mA	19.6 ns

*Including internal logic, interface device, and cable.
†For V_{CC} maintained between 4.75V and 5.25V.

Documentation

NSP-ISE32GNX-M ISE32 User's Manual for SYS32/GENIX and VAX/UNIX operation. (Included with the appropriate Series 32000 support/cross-support software package.)

NSP-ISE32COF-M ISE32 User's Manual for VR32/System V/Series 32000 operation. (Included with the appropriate Series 32000 support software package.)

NSP-ISE32VMS-M ISE32 User's Manual for VAX/VMS operation. (Included with the appropriate Series 32000 cross-support software package.)

ISE32 Debugger Command Summary

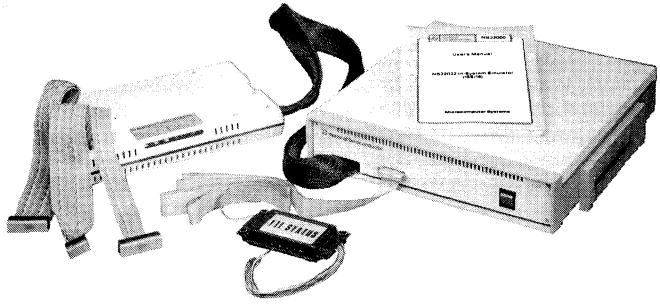
The following comprehensive list of ISE32 Debugger commands is in alphabetical order. Refer to the ISE32 User's Manual for a detailed description of each command.

Command	Function
Begin	Load the program into memory and initializes registers.
Breakpoint Create	Creates breakpoint A, B, C, D at specified address or, creates RANGE breakpoint at specified address range.
Breakpoint Delete	Deletes specified breakpoint.
Breakpoint Print	Print address and conditions of specified breakpoints.
Breakpoint Revive	Revives specified breakpoint.
Indirect File	Executes command file or debugger string.
Debugger String	Sets debugger string.
Define Counter	Defines set up for ISE counter 1 or 2.
Define Latch	Defines the latch 0 or latch 1 event.
Define Output Sync	Defines output sync event.
Define Stop	Defines stop event.
Define Trace	Defines the end, delay, and trace mode parameter for trace.
Disassemble	Disassembles instructions.
Go	Starts execution of the program.
Help	Displays general help.
In	Checks that the contents of address or register are within a specified range.
List Calls	List entries in a call.
List Definition	Lists current definitions.
List Files	Lists nine entries of a selected file.
List Information	Lists current ISE status.
List Modules	Lists modules in current program.
List Strings	Lists current debugger string values.
List Trace	Lists nine trace entries.
Map Create	Maps and/or protects 4 kbyte blocks in a specified address range.
Map Print	Prints current mapping.
Memory Fill	Fills specified address range with value.

Command	Function
Memory Move	Moves memory content from address range to address range.
Memory Search	Searches for value.
On	Sets idbg32 response on condition.
Print	Prints content of address range or registers.
Print Address	Prints absolute address and module area associated with address.
Protection Create	Creates protection/ translation for pages specified by address range.
Protection Print	Prints protection level status.
Quit	Terminates session.
Repeat	Repeats previous command.
Replace	Replaces content of address or register.
Select Echo	Selects echo mode.
Select Full	Selects full symbolic PC.
Select History	Selects history file.
Select Link	Selects communication channel.
Select Module	Selects module.
Select Options	Select current ISE operation option.
Select Radix	Select global radix.
Step	Execute specified number of machine instructions.
Step Call	Executes until a call or return.
Step Down	Executes one instruction inside a procedure, skips over call instructions.
Step Instruction	Executes specified number of instructions inside a module.
Step Until	Executes instructions until contents of address or register are within specified value.
Step While	Executes instructions while contents of address or register are within specified value.

National Semiconductor Corp.

ISE16™ NS32016 In-System Emulator



TL/R/5127–1

- **Operation up to 10 MHz***
- **Emulation of NS32016 Central Processing Unit, NS32082 Memory Management Unit, NS32201 Timing Control Unit**
- **Host resident high-level language and assembly language symbolic debugger**
- **Generalized event driven system**
- **Memory mapping, up to 30 kbytes**
- **Write protection/detection of 2 kbyte memory blocks**
- **Program flow tracing, up to 255 non-sequential fetches**
- **Complete bus activity trace**
- **Qualified tracing**
- **Pre-, post-, or center-triggering on trace**
- **Count-down event counter**
- **Count-up execution timer/counter**
- **Supports Memory Management Unit functions**
- **Supported under various host systems and operating systems**
- **Hierarchical help facility (on-line)**
- **Self-diagnostic**

Description

The ISE16, NS32016 In-System, Emulator is a powerful tool for both hardware and software development of NS32016 microprocessor-based products.

The ISE16 emulates a specific Series 32000® chip set. This chip set includes the NS32016 Central Processing Unit (CPU), the NS32082 Memory Management Unit (MMU), and the NS32201 Timing Control Unit (TCU). NS32032 MMU emulation can be disabled by a switch setting. ISE16 allows users to test and debug both hardware and software in their own hardware environment. ISE16 operates in either of two modes: emulation mode, when ISE16 is actually running the user's program, or monitor mode, when ISE16 is communicating with the user via the host system.

*Refer to speed considerations section.

ISE16 is a complete unit, including an internal clock oscillator and 30 kbytes of dedicated user's ISE™ memory. With ISE16, users can easily stop emulation and examine the contents of CPU registers, slave processor registers, and memory. ISE16 consists of the ISE hardware, the ISE firmware monitor and RS232 cables. A host-dependent debugger software program is available as part of the appropriate Series 32000 cross software support package.

Each of the Series 32000 software support packages include software tools to produce code compatible with the debugger software. Refer to the section "Required User-Supplied Equipment".

Hardware Description

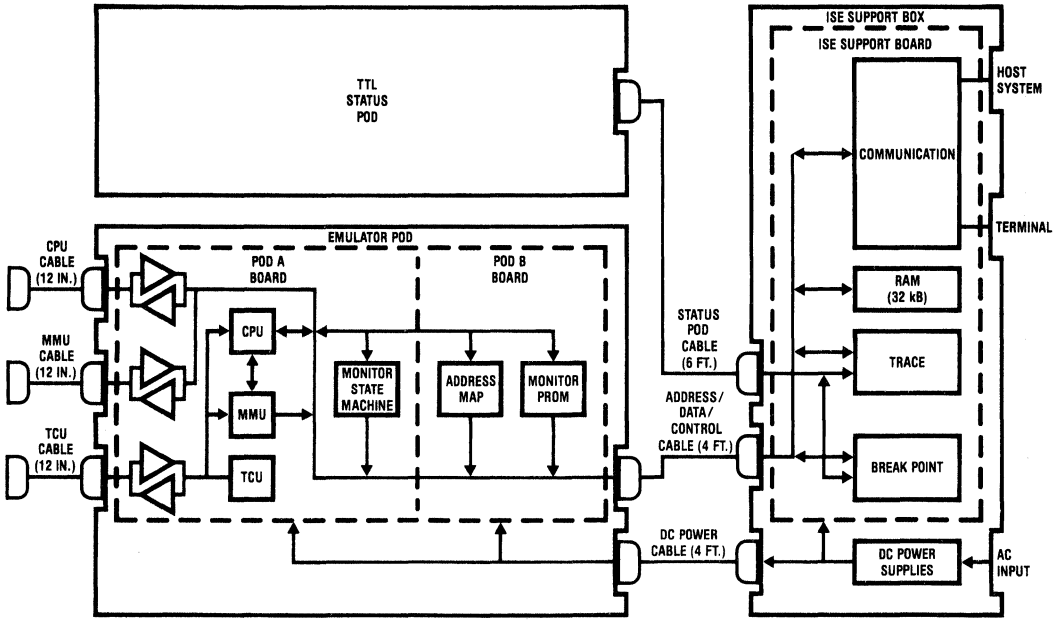
The ISE16 hardware is housed in three enclosures: the ISE Support Box, the Emulator Pod, and the TTL Status Pod. *Figure 1* is a block diagram of ISE16 hardware. The ISE16 enclosures are described in the following paragraphs.

The ISE Support Box is the largest enclosure. It contains the emulation support circuits for trace, breakpoints, and mapped memory. It also contains power supplies and the hardware for the RS232 serial ports.

The Emulator Pod contains the NS32016 CPU, NS32082 MMU, and NS32201 TCU required for target system emulation. It also contains the ISE Monitor firmware and houses the ISE16 controls and indicators. *Figure 2* shows the location of the ISE16 controls and indicators. Table I lists the functions of each switch and LED.

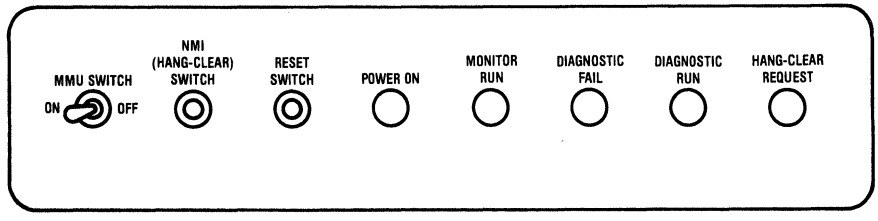
The Emulator Pod connects to the ISE Support Box through a 4-foot flat cable assembly. Connections to the target system are made with 12-inch target cables. One target cable is provided for each member of the Series 32000 chip set (CPU, MMU, and TCU).

The Status Pod is the smallest enclosure. It provides TTL-compatible input and output signals for use during ISE operation. The Status Pod has eleven leads and five binder posts that can be connected to



TL/R/5127-2

FIGURE 1. ISE16 Block Diagram



TL/R/5127-3

FIGURE 2. ISE16 Controls and Indicators

either the target system or test equipment such as logic analyzers or oscilloscopes. Table II lists the function of each lead and post of the Status Pod. The Status Pod connects to the ISE Box front panel status connector via a 6-foot cable.

ISE16 Software Overview

The ISE16 software consists of the ISE firmware monitor, which resides in PROMs in the Emulator Pod, and the ISE Debugger, which runs on the host system.

The monitor controls the ISE hardware. The Debugger is a high-level user-friendly debugger program. It translates commands entered by the user on the host system from a terminal, into low-level instructions the ISE monitor uses to drive the hardware with. The Debugger also translates and sends ISE responses to the user via the terminal. All ISE monitor operation is transparent to the user.

Debugger software is available for the following host systems: SYS32/GENIX, VAX11/VMS, VAX11/UNIX™ (Berkeley), and VR32/UNIX (System V). Please contact a sales representative for a complete list of host software.

The ISE16 Debugger

The ISE16 Debugger is user compatible with the standard non-ISE Series 32000 Debugger. Compatibility minimizes learning time for users of the various development tools. The ISE16 Debugger fully supports all the power debugging and emulation facilities provided by the ISE16 hardware, and supplements these features with a very powerful software-based program debugging environment.

TABLE I. ISE16 Control and Indicator Functions

Control/Indicator	Function
MMU Switch	When on, it enables MMU operation (Mbit in CPU Configuration Register set to 1). When off, disables MMU (Mbit set to 0).
NMI Switch	When pressed, <HANG-CLEAR> occurs. <HANG-CLEAR> restores control to ISE monitor.
RESET Switch	When pressed, resets the ISE hardware.
POWER ON	Indicates power to ISE.
MONITOR RUN	Indicates ISE monitor is running.
DIAGNOSTIC RUN	Indicates ISE diagnostics are running.
DIAGNOSTIC FAIL	Indicates failure during diagnostic tests.
HANG-CLEAR REQUEST	Indicates CPU has stopped executing instructions.

TABLE II. Status Pod Signal Description

Status Pod Label	ISE Function
1-WHT-USRCLK-U	IS0 (input sync 0)
2-BLK-GND	Common Ground
3-BRN-EXTO-U	EXT0 (external input 0)
4-RED-EXT1	EXT1 (external input 1)
5-ORN-EXT2	EXT2 (external input 2)
6-YEL-EXT3	EXT3 (external input 3)
7-GRN-EXT4	EXT4 (external input 4)
8-BLU-EXT5	EXT5 (external input 5)
9-VIO-EXT6	EXT6 (external input 6)
10-GRY-EXT7	EXT7 (external input 7)
11-WHT-USEBRK/U	IS1 (input sync 1)
TBRUN	Multi-Processor Sync
BK SYNCH/-U	DO (output sync)
TR SYNCH/-U	TO (trace sync)
GND	Common Ground
GND	Common Ground

The basic debugging features of the ISE16 are as follows:

(1) Supports both high-level and assembly languages.*

(2) Breakpoints can be set at the source code level, even when using high-level languages.*

(3) Supports symbolic debugging; variables can be referenced by their source code names.*

(4) Certain procedure parameters and variables are easily displayed.

(5) Structured data types and pointers are easily displayed.

(6) Supports both command and history files.

(7) Memory can be displayed in many different ways, including a disassembly mode displaying memory as NS32016 instructions.

(8) Supports all the emulation and debug facilities provided by the ISE16 hardware.

*Depends on host environment and language.

ISE Speed Considerations

ISE16 utilizes standard, 10 MHz NS32016, NS32082, and NS32201 devices to perform control and emulation functions. When emulating, each device is connected to customer hardware via a target cable and associated cable transceivers. This arrangement delays the signal propagation between the Series 32000 components in the ISE16 POD and Series 32000 sockets in the customer hardware. Those delays reduce timing margins in that hardware; i.e., combined propagation paths are lengthened by the ISE16 target cable and transceiver delays.

If sufficient timing margins are not restored, emulation may not be successful. In many cases, margin can be restored by reducing the emulation speed, lengthening the available time for signals to propagate. However, the exact speed reduction necessary to regain margins will depend on how the Series 32000 components are used in the customer hardware. Tables III, IV and V list the combined cable and transceiver maximum propagation delay for each signal. It is the customer's responsibility to factor these delays with those of his own circuitry. In doing so, it can be determined whether ISE16 can reliably emulate with that circuitry.

Optional Terminal Feature

ISE16 can be set-up to operate in either transparent mode or stand-aside mode.

In stand-aside mode, one serial RS232 link from the host system is connected to the ISE16 while another serial RS232 link from the host system is connected to the user terminal. In this configuration, several users can access and use the ISE16.

In transparent mode, one serial RS232 link from the host system is connected to one serial port on the ISE16 while the user terminal is connected to a second serial port on the ISE16. Thus, only one serial port is required from the host system. In non-emulation mode, the ISE16 is transparent to the user allowing normal communication between the user and the host system. In this configuration, a user can do off-site remote development work.

ISE16 Operation

Human Interface

ISE16 is easy to learn and easy to use. The software includes a complete on-line help facility. Invoking the "HELP" command gives a summary of all ISE16 commands, an individual command, or an individual command's parameter. This feature helps the user get his work done quickly with less frustration.

Operational States

The ISE16 can either be in monitor mode or emulation mode. In monitor mode, the ISE firmware program residing in the Emulator Pod is controlling the ISE16 hardware. In emulation mode, the firmware gives control to the user program which, in turn, begins executing and controlling the ISE.

Real-Time Emulation

The ISE16 unit has its own CPU, MMU, and TCU components. These components are connected to the target system via cables, and they perform the same functions, with close to the same timing characteristics as they would if mounted in the target system. ISE16 does not add wait states in its operation. (See speed consideration.)

Emulation memory, resident in ISE16, can be used in lieu of target system memory. This feature is implemented by the mapping capabilities. ISE16 can run and debug programs, without a working target system.

User target memory address space (whether it exists or not) can be mapped onto the ISE16 emulation memory. The control signals (NRD, NWR, NADS . . .) and address lines are still active when using mapped memory. During a memory read from mapped memory, the data from the target is ignored. During a memory write to mapped memory, the data is written to both the map memory and the target.

Memory from the entire 24-bit physical address space of the CPU or MMU can be mapped onto emulation memory if the following restrictions are observed:

(1) Up to four, non-contiguous segments can be defined.

(2) The address range mapped by a segment must lie within an integral 128 kbyte division of the address space, e.g. 00000 to h'1FFFF, or h'20000 to h'3FFFF.

(3) The address range mapped by a segment must start at the beginning of an integral 2 kbyte division of the address space, and end at the end of one such division e.g. h'00 to h'FFF, or h'2800 to h'37FF.

(4) The total memory space mapped by all segments must not exceed 30 kbytes.

Associated with the emulation memory mapping scheme is a capability for write protection/detection. Any 2 kbyte block within any of the four 128 kbyte segments selected can be protected. A write operation to a protected memory segment causes an IM (Illegal Map) event to occur. Write operations to protected memory are inhibited only if they occur on emulation memory. They are not inhibited if they occur in target system memory.

Generalized Events

To provide a versatile way of observing and controlling the significant state changes on the microprocessor, ISE16 allows the use and definition of "events". In general, a simple event is a breakpoint, a bus change, or a significant observation. An event can also be a logical combination of simple events (an Event-Expression).

Simple Event Definition

The simple events are:

- Breakpoints
- Latched Breakpoints
- Counter Done
- Status Pod Inputs
- Illegal Map
- Trace Done

Breakpoint Events

ISE16 provides three common breakpoint events, named A, B, and C. The breakpoint event can be used in two ways:

(1) Execution Breakpoint—occurs just prior to execution of an instruction fetched from a specified address.

(2) Memory Reference Breakpoint—occurs on a match when sampling:

- Address Bits
- Data Bits
- External Status Bits
- User/Supervisor Pin
- High Byte Enable Pin
- Data Direction Pin
- And where any of the above options or bits can be masked.

Either virtual or physical addresses can be sampled.

ISE16 also provides a range breakpoint event, R. The range breakpoint occurs on any read or write operation to an address in a specified address range.

All breakpoints can cause emulation to stop immediately. Also, if used with the No Stop (/NS) option, breakpoints can be combined with other events to cause a variety of action.

Events, instructions, memory cycles, and clock cycles can be counted with the breakpoint counter (up to 12 bits). Upon reaching a certain count provided by the Define Counter (DC) command, the Counter Done (CD) event takes place.

Other Simple Events

The other simple events available are:

(1) IS0, IS1—Status Pod Input Sync 0 and Input Sync 1.

(2) IM—Write operation to write-protected address.

(3) TD—End of trace.

Related commands:

BC—Breakpoint Create

BD—Breakpoint Delete

DP—Breakpoint Print

Event-Expressions

An event-expression is a Boolean expression made up of simple events, i.e., a logical combination of simple events. This allows the user to generate many different event combinations, tailored to system activity of particular interest to the user. These generalized events are used by many ISE16 commands such as stop, trace, event counting, etc. Event-expressions provide creative and flexible debugging procedures.

Event-expressions can be evaluated as either logically true or logically false. Valid logic operations for event-expressions are: Negation (NOT), AND, and OR.

Stopping Execution on Events

A common debugging activity is to stop emulation on the occurrence of an event of interest. Stopping emulation puts ISE16 in the monitor mode so the user can examine and alter the state of the CPU, memory, and ISE16 functions. Emulation can be stopped on either simple events or event-expressions.

Related commands:

DS—Define Stop

BS—Breakpoint Create

Flexible Tracing

ISE16 maintains a 255-entry trace memory. Trace memory captures bus activity in one of two trace modes. The trace modes are:

- Program Flow Trace
- Memory Bus Trace

When enabled, tracing in either mode continues until a specified terminating event occurs. The actual end of tracing can be delayed after the terminating event by a count of 1 to 255. This allows trace data to be captured before, after, or around the terminating event.

Program Flow Trace

The Program Flow Trace mode captures the CPU Program Counter address of 255 non-sequential instructions. This mode also maintains a count of sequential instructions executed between each non-sequential instruction stored in the trace memory.

Memory Bus Trace

The Memory Bus Trace mode captures a summary of system parameters during 255 memory bus access cycles. The following parameters are captured:

- Address bus contents
- Data bus contents
- CPU Status (data transfer, non-sequential fetch, interrupt acknowledge, etc.)
- Status Pod External Inputs
- States of the Following CPU Pins:

  PFSC—Program Flow Status (start of instruction)

  UNS—User/Not Supervisor

  NHBE—Not High Byte Enable

  NDDIN—Not Data Direction In

A tracing event can qualify the memory bus tracing mode. This event allows the user to reduce the number of events captured.

Execution Timer

The execution timer is a 24-bit counter with an overflow flag that may be used to count events, instruction cycles, memory cycles, or clock cycles. The timer may be programmed to start and stop counting on specific events. This permits using the execution timer to determine relative timing differences between various events. One use of this feature is to measure software or hardware performance.

Event Trigger for External Test Equipment

ISE16 events can trigger external test equipment, such as oscilloscopes and logic analyzers. This test equipment can be used in conjunction with ISE16 debugging features to solve system timing problems. Two external trigger sources are provided:

- General Event (or Event-Expression)
- Trace Trigger Event (i.e., an event that causes an entry into trace memory.)

The external trigger signals are available at two status pod outputs:

- BKSYNCH/–U (General Event)
- TRSYNCH/–U (Trace Trigger Event)

ISE16 Timing Options

ISE16 includes the following timing options:

- Sampling time can be set to sample either virtual or physical addresses
- Status Pod external lines can be sampled at either data valid or address valid times
- The emulation clock frequency can be set to one of the following frequencies:

  2.5 MHz

  5.0 MHz

  10 MHz

  Target Board Frequency

Self-Test Diagnostics

At power-up or reset, ISE16 runs a diagnostic program to verify ISE software integrity and proper hardware function.

Supported Configurations

This product is designed to work in most target systems configurations. However, certain design restrictions may apply. Refer to the ISE16 User's Manual (See "Documentation") for further information.

Required User-Supplied Equipment

For use with SYS32/GENIX Systems:

- Included with GENIX Operating System Software Package.

For use with VAX/VMS Systems:

- Valid DEC™ VAX/11 configuration, with available RS232 port.
- VMS Operating System, Version 3.0 or later.
- NSW-ASSEMB-9VMR or NSW-PASCAL-9VMR Series 32000 Cross Software Package.

For use with VAX/UNIX Systems:

- Valid DEC VAX/11 configuration, with available RS232 port.
- Berkeley UNIX Operating System.
- NSW-C-4VXR Series 32000 Cross Software Package.

For use with VR32/UNIX (System V) systems:

- Included with Operating System Software Package.

For other applications contact National Semiconductor sales representative.

Specifications

Environmental

Operating Temperature
+10°C to +40°C
Storage Temperature
−20°C to +65°C

Power

3A @ 115 V_{AC}, 50/60 Hz, single phase
1.5A @ 220 V_{AC}, 50/60 Hz, single phase
Approximately 1170 BTU.

Physical

ISE Support Box—
Height: 4.125 in. (10.5 cm)
Width: 19.0 in. (48.3 cm)
Depth: 17.5 in. (44.5 cm)

Emulation Pod—
Height: 2.25 in. (6.4 cm)
Width: 9.25 in. (23.5 cm)
Depth: 14.0 in. (35.6 cm)

TTL Status Pod—
Height: 1.0 in. (2.5 cm)
Width: 3.125 in. (7.9 cm)
Depth: 6.125 in. (15.6 cm)

Cable Lengths—
ISE Support Box to Emulation Pod: 4.0 ft. (1.22M)
ISE Support Box to TTL Status Pod: 6.0 ft. (1.83M)
Emulation Pod to Target Board: 1.0 ft. (0.30M)

Electrical

Operating Frequency—
User selectable to one of the following:
2.5 MHz
5.0 MHz
10.0 MHz
Target Board Frequency

Target Interface Electrical Characteristics—
See: Speed Considerations

See Tables III through V.

Order Information

Complete ISE16 Units

NSS-ISE16	ISE16 (NS32016), 115 V_{AC}
NSS-ISE16	ISE16 (NS32016), 220 V_{AC}

Documentation

420306675-002	ISE16 User's Manual for VAX/VMS operation. (Included with appropriate cross-support software package.)
420308165-001	ISE16 User's Manual for VAX/UNIX and SYS32/GENIX operation. (Included with appropriate cross-support software package.)
420010496-001	COFF IDBG16 User's Manual (Included with appropriate cross-support software package.)

TABLE III. Electrical Characteristics for TCU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T_{pd}*
Outgoing Signals:		I_{OH}	I_{OL}	
NTSO	74S244	15 mA	64 mA	14.6 ns
CTTL	74S244	15 mA	64 mA	14.6 ns
FCLK	74S244	15 mA	64 mA	14.6 ns
NDBE	74S244	15 mA	64 mA	14.6 ns
NRD	74S244	15 mA	64 mA	14.6 ns
NWR	74S244	15 mA	64 mA	14.6 ns
NRST	74S244	15 mA	64 mA	14.6 ns
RDY	74S244	15 mA	64 mA	14.6 ns
Incoming Signals:		I_{IH}	I_{IL}	
NPER	74S244	50 μA	400 μA	14.6 ns
NCWAIT	74S244	50 μA	400 μA	14.6 ns
NWAIT1	74S244	50 μA	400 μA	14.6 ns
NWAIT2	74S244	50 μA	400 μA	14.6 ns
NWAIT3	74S244	50 μA	400 μA	14.6 ns
NWAIT4	74S244	50 μA	400 μA	14.6 ns
XCTL1	74S244	50 μA	400 μA	14.6 ns
NCEN	74S244	50 μA	400 μA	14.6 ns
NRST1	74S244	50 μA	400 μA	14.6 ns

*Interfaced device, plus cable.

TABLE IV. Electrical Characteristics for MMU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T_{pd}*
Outgoing Signals:		I_{OH}	I_{OL}	
A24	74S244	15 mA	64 mA	14.6 ns
MMUMINT	74S244	15 mA	64 mA	14.6 ns
NPAV	74S244	15 mA	64 mA	14.6 ns
NABT	74S244	15 mA	64 mA	14.6 ns
NFLT	74S244	15 mA	64 mA	14.6 ns
NHLDA0	74S244	15 mA	64 mA	14.6 ns
Incoming Signals:		I_{IH}	I_{IL}	
NHOLD	74S244	50 μA	400 μA	14.6 ns

*Interface device, plus cable.

TABLE V. Electrical Characteristics for CPU Interface

Signal Name	Interface Device	Input And/Or Output Current				Propagation Delay Time T_{pd}*
Bidirectional Signals		I_{OH}	I_{OL}	I_{IH}	I_{IL}	
NSPC	none	—	—	—	—	1.4 ns
AD15	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD14	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD13	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD12	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD11	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD10	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD09	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD08	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD07	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD06	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD05	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD04	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD03	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD02	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD01	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
AD00	8T28	10 mA	48 mA	25 μA	200 μA	18.4 ns
Outgoing Signals						
A23	74S244	15 mA	64 mA	—	—	14.6 ns
NIL0	74S244	15 mA	64 mA	—	—	14.6 ns
ST0	74S244	15 mA	64 mA	—	—	14.6 ns
ST1	74S244	15 mA	64 mA	—	—	14.6 ns
ST2	74S244	15 mA	64 mA	—	—	14.6 ns
ST3	74S244	15 mA	64 mA	—	—	14.6 ns
NPFS	74S244	15 mA	64 mA	—	—	14.6 ns
NDDIN	74S244	15 mA	64 mA	—	—	14.6 ns
NADS	74S244	15 mA	64 mA	—	—	14.6 ns
UNS	74S244	15 mA	64 mA	—	—	14.6 ns
NHBE	74S244	15 mA	64 mA	—	—	14.6 ns
HHLDA	74S244	15 mA	64 mA	—	—	14.6 ns
A22	74S244	15 mA	64 mA	—	—	14.6 ns
A21	74S244	15 mA	64 mA	—	—	14.6 ns
A20	74S244	15 mA	64 mA	—	—	14.6 ns
A19	74S244	15 mA	64 mA	—	—	14.6 ns
A18	74S244	15 mA	64 mA	—	—	14.6 ns
A17	74S244	15 mA	64 mA	—	—	14.6 ns
A16	74S244	15 mA	64 mA	—	—	14.6 ns
Incoming Signals						
TSYSPWR	1N4002	—	—	—	—	—
NINT	74S244	—	—	50 μA	400 μA	14.6 ns
NNMI	74S244	—	—	50 μA	400 μA	14.6 ns
NHOLD	74S244	—	—	50 μA	400 μA	14.6 ns

*Interface device, plus cable.

TABLE VI. ISE16 Debugger Command Summary

The following comprehensive list of ISE16 debugger commands is in alphabetical order. Refer to the ISE16 User's Manual for a detailed description of each command.

Command	Function
Begin	Load the program into memory and initializes registers.
Breakpoint Create	Creates breakpoint A, B, C at specified address or, creates RANGE breakpoint at specified address range.
Breakpoint Delete	Deletes specified breakpoint.
Breakpoint Print	Print address and conditions of specified breakpoints.
Breakpoint Revive	Revives specified breakpoint.
Indirect File	Executes command file or debugger string.
Debugger String	Sets debugger string.
Define Counter	Defines setup for ISE counter.
Define Output Sync	Defines output sync event.
Define Stop	Defines stop event.
Define Timer	Defines execution timer.
Define Trace	Defines the end, delay, and trace mode parameter for trace.
Disassemble	Disassembles instructions.
Go	Starts execution of the program.
Help	Displays general help.
In	Checks that the contents of address or register are within a specified range.
List Calls	List entries in a call.
List Definition	Lists current definitions.
List Files	Lists nine entries of a selected file.
List Information	Lists current ISE status.
List Modules	Lists modules in current program.
List Procedure	Lists current procedure.
List Strings	Lists current debugger string values.
List Trace	Lists nine trace entries.
Map Create	Maps and/or protects 4 kbyte blocks in a specified address range.
Map Print	Prints current mapping.
Memory Fill	Fills specified address range with value.

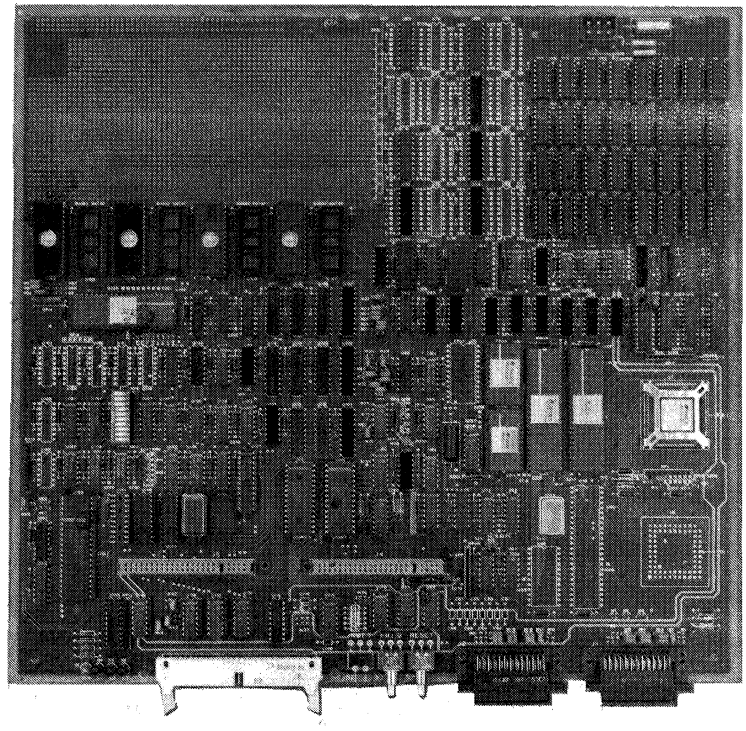
Command	Function
Memory Move	Moves memory content from address range to address range.
Memory Search	Searches for value.
On	Sets debugger response on condition.
Print	Prints content of address range or registers.
Print Address	Prints absolute address and module area associated with address.
Protection Create	Creates protection/translation for pages specified by address range.
Protection Print	Prints protection level status.
Quit	Terminates session.
Repeat	Repeats previous command.
Replace	Replaces content of address or register.
Select Echo	Selects echo mode.
Select Full	Selects full symbolic PC.
Select History	Selects history file.
Select Link	Selects communication channel.
Select Module	Selects module.
Select Options	Select current ISE operation option.
Select Procedure	Selects procedure.
Select Radix	Select global radix.
Step	Execute specified number of machine instructions.
Step Call	Executes until a call or return.
Step Down	Executes one instruction inside a procedure, skips over call instructions.
Step Instruction	Executes specified number of instructions inside a module.
Step Until	Executes instructions until contents of address or register are within specified value.
Step While	Executes instructions while contents of address or register are within specified value.

TABLE VII. Register Names

Name	Description	Name	Description
CPU Registers			
.R0	General Register 0	.PC	Program Counter
.R1	General Register 1	.SB	Static Base Register
.R2	General Register 2	.FP	Frame Pointer Register
.R3	General Register 3	.SP	Current Stack Pointer
.R4	General Register 4	.IS	Interrupt Stack Pointer
.R5	General Register 5	.US	User Stack Pointer
.R6	General Register 6	.MOD	Module Register
.R7	General Register 7	.INTBASE	Interrupt Base Register
.PSR	Processor Status Register		
.UPSR	Processor Status (Low Byte)		
.CFG	Configuration Register		
1. If the .SB register is changed, IDBG16 changes the relevant entry in the module table.			
2. If the .CFG register is changed, IDBG16 executes a SETCFG instruction.			
PSR Fields			
.PSR_C	Carry Field	.PSR_N	Negative Field
.PSR_T	Trace Trap Field	.PSR_U	User/Supervisor Mode Field
.PSR_L	Low Field	.PSR_S	User/Interrupt Stack Field
.PSR_F	Flag Field	.PSR_P	Trace Pending Field
.PSR_Z	Zero Field	.PSR_I	Interrupt Enable/Disable
FPU Registers (available only when FPU device is present)			
.F0	Floating Point Register 0	.F4	Floating Point Register 4
.F1	Floating Point Register 1	.F5	Floating Point Register 5
.F2	Floating Point Register 2	.F6	Floating Point Register 6
.F3	Floating Point Register 3	.F7	Floating Point Register 7
.FSR	Floating Point Status		
FSR Fields (available only when FPU device is present)			
.FSR_TT	Trap Type Field (3-bits)	.FSR_IEN	Inexact Result Enable Field
.FSR_UEN	Underflow Enable Field	.FSR_IFL	Inexact Result Trap Field
.FSR_UFL	Underflow Trap Field	.FSR_RM	Rounding Mode Field (2-bits)
MMU Registers (available only when MMU device present)			
.PTB0	Page Table 0	.BCNT	Breakpoint Count Register
.PTB1	Page Table 1	.PF0	Program Flow 0
.EIA	Error/Invalidate Address	.PF1	Program Flow 1
.BPR0	Breakpoint Register 0	.SC	Sequential Count Register
.BPR1	Breakpoint Register 1	.MSR	MMU Status Register
MSR Fields (available only when MMU device is present)			
.MSR_ERC	Error Class Field	.MSR_DS	Dual Space Field
.MSR_TET	Translation Error Trace	.MSR_AO	Access Override Field
.MSR_BN	Breakpoint Number Field	.MSR_BEN	Breakpoint Enable Field
.MSR_ED	Error Direction Field	.MSR_UB	User Break Field
.MSR_BD	Breakpoint Direction Field	.MSR_AI	Abort/Interrupt Field
.MSR_EST	Error Status Field	.MSR_FT	Flow Trace Field
.MSR_BST	Breakpoint Status Field	.MSR_UT	User Trace Field
.MSR_TU	Translate User Field	.MSR_NT	Non-Sequential Trace Field
.MSR_TS	Translate Supervisor Field		

National Semiconductor Corp.

DB32000 Development Board



TL/EE/8523–1

- **Series 32000® Microprocessor Family**
  - **— NS32032 Central Processing Unit (CPU) (can be replaced by NS32016 CPU, or NS32008 CPU, for evaluation)**
  - **— NS32082 Memory Management Unit (MMU)**
  - **— NS32081 Floating Point Unit (FPU)**
  - **— NS32202 Interrupt Controller Unit (ICU)**
  - **— NS32201 Timing Control Unit (TCU)**
- **256K bytes DRAM expandable to 1 Mbyte**
- **Up to 256K bytes of EPROM in two banks**
- **Two RS-232 Serial Communication Ports**
- **24 Programmable Parallel I/O Lines**
- **Two BLX™ Connectors**
- **Wire-wrap area for user expansion**
  - **— Bus interface**
  - **— Dual port RAM**
  - **— ROM expansion**
  - **— I/O expansion**
  - **— RAM expansion**
- **TDS™ firmware provides edit, assembly, and debug capabilities**

Product Overview

National Semiconductor's DB32000 Development Board is a complete microcomputer system. It is specifically designed to assist the user in evaluating and developing hardware and software for the NS32032 CPU, related slave processors (NS32081 FPU and NS32082 MMU) and support devices. With the DB32000, the user may evaluate other CPUs such as the NS32016 and NS32008. The DB32000 enables

Product Overview (Continued)

the user to examine the architecture, instruction set, cycle timing, and the bus interfaces for the Series 32000® family of microprocessors. Small programs can be written, debugged, assembled, and executed with EPROM-based TDS (Tiny Development System) software.

Optionally, the DB32000 can provide a native debug and execution environment for programs developed on a larger host computer system. In this case, the board complements capabilities provided by National's Pascal and C cross-software packages.

The DB32000 includes the NS32032 CPU, NS32082 MMU, NS32081 FPU, NS32202 ICU, support circuitry, dynamic RAM, extensive ROM/EPROM capacity, and serial and parallel I/O. I/O capability can also be expanded via BLX interfaces.

Central and Slave Processors

The DB32000 is equipped with an NS32032 CPU, featuring 32-bit internal structure and 32-bit data bus. Optionally, an NS32016 or NS32008 CPU can be installed, with 32-bit internal structure and 16-bit or 8-bit data path. Each CPU provides a very powerful instruction set designed for high level language support.

The DB32000 also includes the NS32082 MMU and the NS32081 FPU. The NS32082 Memory Management Unit provides hardware support for demand-paged virtual memory management. The NS32081 provides high-speed floating-point instruction execution.

Interrupts

As part of factory configuration, the DB32000 comes with the NS32202 ICU installed. The NS32202 Interrupt Control Unit manages up to 16 maskable interrupt sources, resolves interrupt priorities, and supplies a single-byte vector to the CPU. In addition, the ICU provides two, 16-bit counters.

Memory

Expandable to 1 Mbyte, 256K bytes of on-board dynamic RAM are provided. The wire-wrap area may be used in conjunction with the DB32000 circuitry to develop dual port capability.

Up to 256K bytes of ROM/EPROM space is provided in eight 28-pin sockets. The sockets are divided into two banks, each bank permitting installation of 24- or 28-pin devices. All factory configurations include TDS firmware installed in the lower bank, with the upper bank vacant.

Parallel I/O

Twenty-four parallel I/O lines are provided via an 8255A Programmable Peripheral Interface. These may be divided into two 8-bit ports and two 4-bit ports.

Serial I/O

Two serial I/O ports are provided via 2651 Universal Synchronous/Asynchronous Receiver/Transmitters. These ports permit the DB32000 to communicate with RS232C compatible terminals or other computers. The baud rate for each port is software programmable.

BLX I/O Expansion

Two connectors are provided for attachment of 8- or 16-bit BLX expansion modules. BLX modules may be used to expand the DB32000's I/O capability; *e.g.,* additional serial on parallel ports.

Switches

Two button switches (S3 and S4), and one 10-position DIP switch (S1) are provided. S3, labeled NMI 0, will introduce a non-maskable interrupt to the DB32000's CPU when pressed. S4, labeled RESET, will reset the board when pressed. Switch S1 is a software readable dip switch that may be used to indicate defined options, *e.g.,* baud rate, MMU present, etc. Each switch position function is defined by the on-board PROM-based software.

Indicators

Four LED indicators (D2–D5) are mounted near the lower left corner of the DB32000. D2–D4 are controlled by the contents of a program-addressed register. They are used by the TDS power-on confidence test program to indicate test status. They may also be used to indicate any other information the user desires. D5 is driven directly by a 15-millisecond 1-shot timer. D5 will be extinguished whenever there is no CPU memory or I/O access within this time. D5 is illuminated when the CPU is executing instructions. This LED indicates whether or not the CPU is active.

Wire-Wrap Expansion Area

The wire-wrap expansion area provides the user with space that is drilled to accept integrated circuits. Signal pad terminators (stubs) are located at different locations on the board enabling the user to construct the following functions in the wire-wrap area:

— External Bus Interface
— Dual Port Memory Interface
— ROM Expansion
— I/O Expansion
— DRAM Capacity Expansion Using the On-board DRAM Controller

Tiny Development Systems (TDS) Functional Description

The TDS firmware allows the user to create programs by entering source via the editor. This source is then assembled to produce executable code suitable for debugging. These functions have the following features:

Assembler:

— Subset of existing Series 32000 assembler
— Supports FPU by providing long and short format real number data initialization
— Generates listings to either a printer at the parallel port, or any RS232 device connected via serial port
— Symbolic definition of static base or PC segment

Debugger:

— Numerical arguments to commands can be in four bases: decimal, hex, long real and short real
— Program flow visually traced by displaying source line at all breakpoints or step stops
— Memory/register print or change commands
— Step-through program commands: step "n" instructions, step while variable in range, step until variable reached

Editor:

— Commands to insert, replace, delete, type lines
— Automatic line number maintenance
— Save and retrieve source from audio cassette recorder
— Upload/download to/from any RS232-equipped PC
— Debug data displayed by type command after assembly

User Program Run Time Support:

— Accessed via a supervisor call instruction
— Routines to do terminal I/O
— Printer driver access to parallel port
— Routine to convert binary value to ASCII string
— Routine to convert ASCII string to binary value
— Conversion in four bases: decimal, hex, long real and short real

As shipped with the DB32000, TDS provides on-board hardware confidence test routines. These are invoked by power-on or manual reset.

User Modes

The DB32000 can operate stand-alone, with no assistance from a host computer system. Optionally, the board can be operated in conjunction with a host, taking advantage of more powerful software development tools and I/O capabilities.

Stand-Alone Mode (Factory Configuration)

The stand-alone user mode (see *Figure 1*) requires only an RS232C-compatible terminal and power supplies for the DB32000.

TDS (Tiny Development System) software is supplied in on-board PROMs to support this user mode. TDS is used to edit, assemble, and execute small Assembly language programs. In addition, TDS can control the DB32000's on-board I/O to provide cassette and printer interfaces, making the DB32000 a light duty development vehicle.

Host-Assisted Modes

The DB32000 can be connected to another computer system or host (refer to *Figure 1*). In this case, the user first develops Series 32000 software on the host system, then uses the RS232 communication link to download the software to the DB32000, which executes and debugs the software in a native environment.

Several development software packages are available for use in generating Series 32000 user programs. Among them are:

— Pascal and C, operating under VAX/VMS
— Pascal and C, operating under VAX/UNIX®

In each case, the DB32000's factory-supplied, on-board TDS software must be replaced. A suitable PROM-based monitor software package is supplied with the host development software.

The basic modes of host-assisted DB32000 operation are "stand-aside" and "transparent". The terms "stand-aside" and "transparent" may be visualized by observing the communication configuration for each mode. Refer to *Figure 1*.

The monitor software will provide:

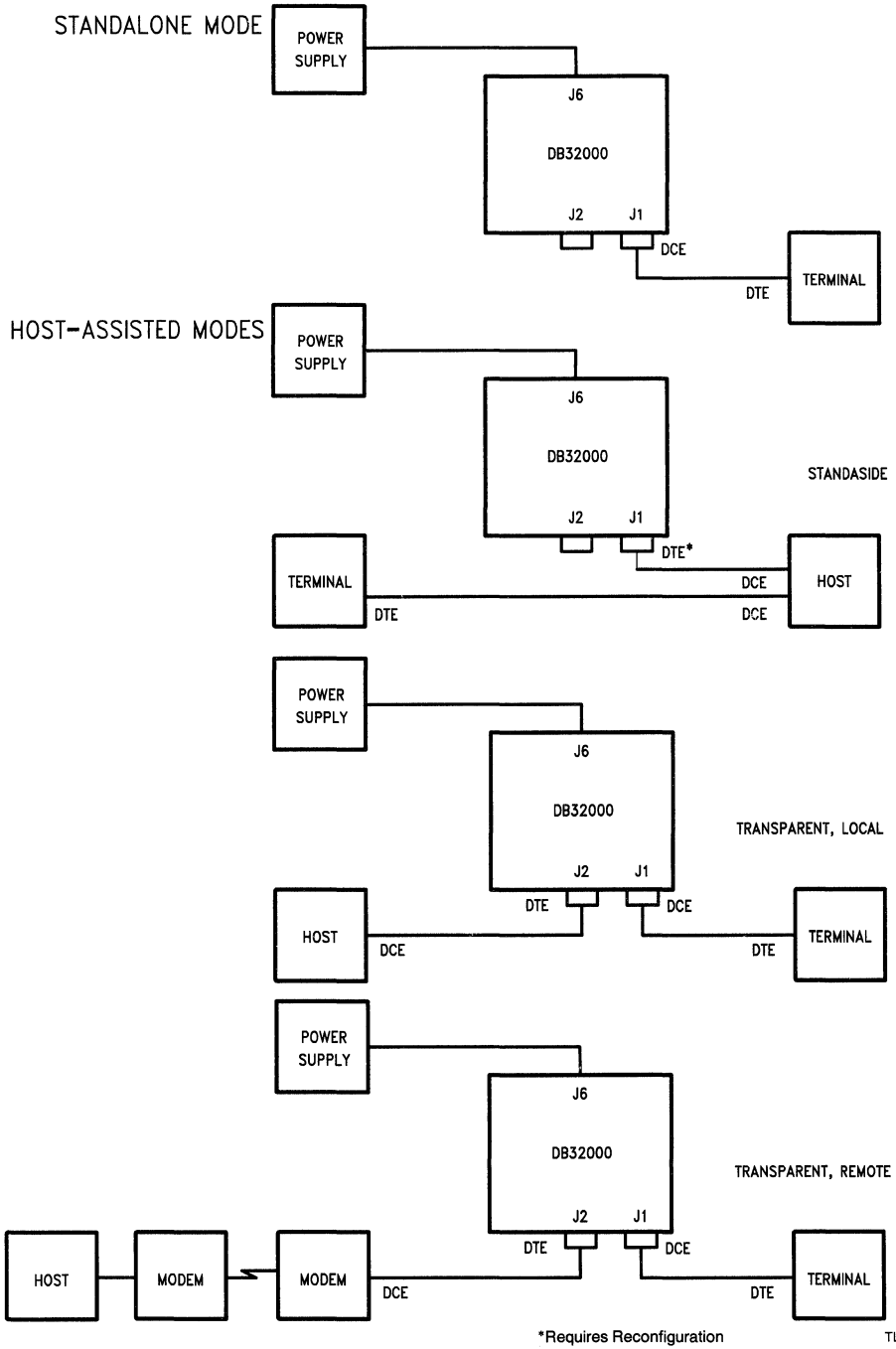
— Terminal Handler (for use in transparent mode)
— Run-Time Environment (to permit execution of downloaded programs)
— Debugger Execute Module (to permit operation with the host's debugger)

Consult the Development Board Monitor Reference Manual for a complete description of capabilities.

In transparent mode, the user's communication with the host is conducted through the DB32000, which is transparent to the user. One advantage is that a single RS232 port on the host computer will support both the user's terminal and the DB32000.

In stand-aside mode, the user communicates directly with the host while the DB32000 "stands aside". This mode is useful when the DB32000 is connected to single-user hosts, notably those where the terminal and keyboard are integral to the host. Optionally, stand-aside operation is possible with multi-user hosts where two RS232 ports are available.

User Modes (Continued)



*Requires Reconfiguration

TL/EE/8523-2

FIGURE 1. DB32000 Configurations

Specifications

Environment

The DB32000 is designed for operation in an office or laboratory environment. Avoid confining the DB32000 in a closed space, unless sufficient air flow is provided to ensure all components are operated within their specified temperature range.

Temperature:	Operating	0°C to +55°C
	Nonoperating	−40°C to +75°C
Humidity:	5% to 95% relative, noncondensing	
Altitude:	Operating	up to 15,000 ft.
	Nonoperating	up to 25,000 ft.

Power Requirements

The DB32000 requires three regulated DC voltages for operation:

— +12 volts DC, ±10%, 40 mA typical (50 mA max)
— −12 volts DC, ±10%, 40 mA typical (50 mA max)
— +5 volts DC, ±5%, 5A typical (10A max)

Ordering Information:

All models are shipped with:

Two RS232 cable sets

Model DB32000-110 Includes NS32032-10 CPU
NS32201-10 TCU
NS32202-10 ICU
NS32081-10 FPU
NS32082-10 MMU

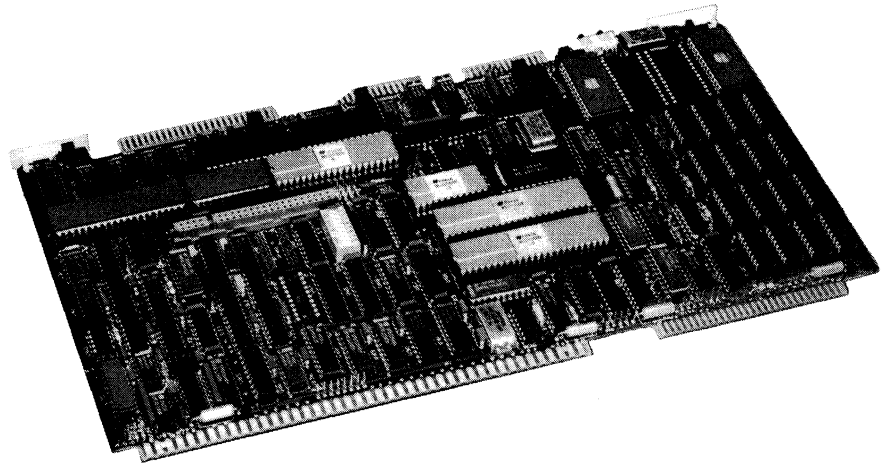
NSP-TDS-M Series 32000 TDS: Tiny Development System User's Manual

NSP-DB32000-M Series 32000 DB32000 Development Board User's Manual

NSP-INST-REF-M Series 32000 Instruction Set Reference Manual

National Semiconductor Corp.

DB32016 Development Board



TL/R/7083–1

- **Series 32000® Microprocessor Family**
  - **NS32016 CPU (can be replaced by NS32008 CPU, for evaluation)**
  - **NS32082 MMU**
  - **NS32081 FPU**
  - **NS32202 ICU**
  - **NS32201 TCU**
- **MULTIBUS® multi-master bus interface**
- **128 Kbytes dual ported RAM**
- **Up to 96 Kbytes PROM capacity**
- **Two RS232 serial communication ports**
- **24-programmable parallel I/O lines**
- **Three 16-bit programmable timer/ counters**
- **One BLX™ expansion module connector for additional I/O capability**
- **TDS™ firmware provides edit, assembly, and debug capabilities**

Product Overview

The DB32016 Development Board is a complete microcomputer using the National Semiconductor Series 32000 family of advanced microprocessors. It is specifically designed to assist evaluation and development of Series 32000 applications in a variety of environments.

By itself, the DB32016 can be used to examine the Series 32000 architecture and instruction set. Small programs can be written, debugged, and executed with EPROM-based TDS (Tiny Development System) software.

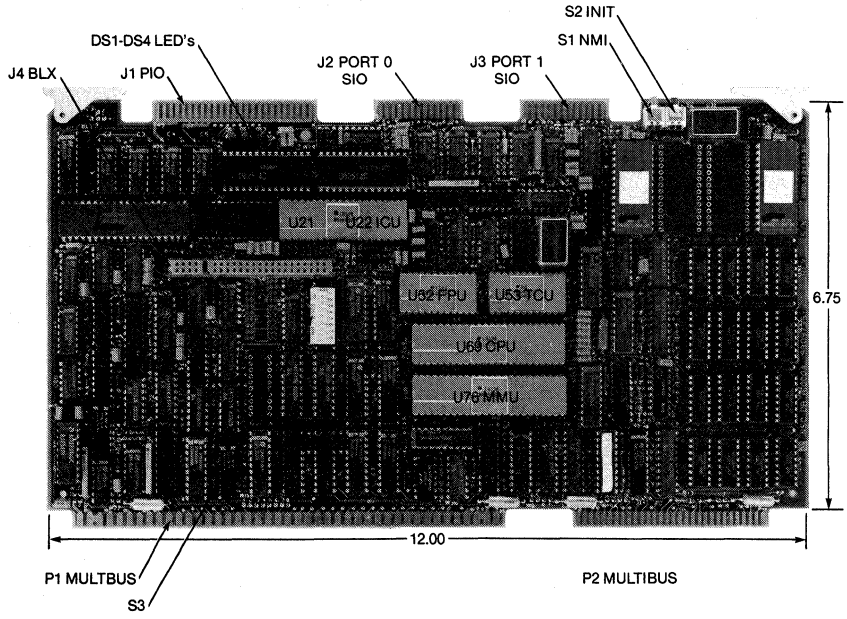
Optionally, the DB32016 can provide a native debug and execution environment for programs developed on a larger host computer. In this case, the board complements capabilities provided by National's C and Pascal cross software packages.

Flexibility is further enhanced by the board's MULTIBUS interface. This permits expansion of the DB32016 microcomputer system to include functions provided by other MULTIBUS compatible boards; e.g. disk/tape controllers, bulk RAM, etc.

All models of the DB32016 include, as a minimum, the NS32016 CPU, support circuitry, serial and parallel I/O, dynamic RAM, and extensive ROM/EPROM capacity. Optionally, the board can be populated with NS32082 Memory Management Unit, NS32081 Floating-Point Unit, and NS32202 Interrupt Control Unit. In all cases, I/O capability can be expanded via BLX and MULTIBUS interfaces.

Hardware Function Description

(Refer to *Figure 1-1*)



TL/R/7083-2

FIGURE 1-1. DB32016 Topography

Central and Slave Processors

The DB32016 is equipped with a NS32016 CPU, featuring 32-bit internal structure and 16-bit data bus. Optionally, a NS32008 CPU can be installed, with 32-bit internal structure and 8-bit data path. Each CPU provides a very powerful instruction set designed for high level-language support.

Included with each DB32016 is the NS32082 MMU and the NS32081 FPU slave processors. The NS32082 Memory Management Unit provides hardware support for demand-paged virtual memory management. The NS32081 provides high-speed floating-point instruction execution.

If the DB32016 is purchased without slave processors, they may be installed by the customer, as required.

Interrupts

Also included with the DB32016 is the NS32202 ICU. The NS32202 Interrupt Control Unit manages up to 16 maskable interrupt sources, resolves interrupt priorities, and supplies a single-byte vector to the CPU. In addition, the ICU provides two, 16-bit counters, one of which can provide programmable baud rate capability for the DB32016's serial I/O ports.

If the DB32016 is purchased without the ICU, it may be installed by the customer, as required.

Memory

128 Kbytes of on-board, dual-ported dynamic RAM are provided. The MULTIBUS starting address of RAM is mappable in 32K byte increments, across the entire 16M byte address space.

Up to 96 Kbytes of ROM/EPROM space is provided in four 28-pin sockets. The sockets are divided into two banks, each bank permitting installation of 24 or 28-pin devices. All factory configurations include TDS firmware installed in the lower bank, with the upper bank vacant.

MULTIBUS Interface

The DB32016 incorporates a MULTIBUS interface, allowing the user to configure larger systems. Most often, the DB32016 would be used in conjunction with MULTIBUS compatible expansion RAM, disk controller, or serial controller boards. However there is no restriction, beyond MULTIBUS compliance.

The DB32016's MULTIBUS compliance levels are:

Master	D16 M24 I16 VOEL; indicating 8/16-bit data path, 24-bit memory address path, 8- or 16-bit I/O address path, and level or edge triggered non-bus vectored interrupts (if NS32202 is installed).
Slave	D16 M24; indicating 8/16-bit data path, and 24-bit memory address path.

Parallel I/O

24 parallel I/O lines are provided via an 8255A Programmable Peripheral Interface. These may be divided into two 8-bit ports and two 4-bit ports.

Serial I/O

Two serial I/O ports are provided via 8251A Universal Synchronous/Asynchronous Receiver/Transmitters. These ports permit the DB32016 to communicate with RS232 compatible terminals or other computers.

Port baud rates may be derived from a variety of sources:

- a fixed, 9600 baud operation of both ports, if the NS32202 ICU is not installed.
- single programmable baud rate for both ports, if the NS32202 ICU is installed.
- Individually programmable baud rates for each port, via the DB32016's 8253-5 PIT.

Timer/Counters

As mentioned above, the NS32202 ICU provides two 16-bit timer/counters, when installed. In addition, three 16-bit counters are provided by the DB32016's 8253-5 Programmable Interval Timer. Each counter output is available for connection as an interrupt source for the ICU, or baud rate generation for the serial ports.

BLX I/O Expansion

A connector is provided for attachment of 8- or 16-bit BLX expansion modules. BLX modules may be used to expand the DB32016's I/O capability; e.g. additional serial or parallel ports.

Switches

Two push button switches (S1 and S2), and one eight-position DIP switch (S3) are provided.

S1, labeled NMI, will introduce a non-maskable interrupt to the DB32016's CPU when pressed. S2, labeled INIT, will reset the board when pressed. Both switches are located on the front edge of the board assembly. DIP switch S3 is used to set the Baud rate of the serial ports and other board configurations.

Indicators

Four LED indicators (DS1-DS4) are mounted near the front edge of the board assembly.

DS1-3 are controlled by the contents of a program addressed register. They are used by the TDS power-on confidence test program to indicate test status. They may also be used to indicate any other information the user desires.

DS4 is driven directly by a one-shot timer, whose period is approximately 15 milliseconds. DS4 will be illuminated whenever there is no memory or I/O access completed by the CPU within this period. This is useful to indicate a MULTIBUS timeout.

Tiny Development Systems (TDS) Functional Description

The TDS firmware allows the user to create programs by entering source via the editor. This source is then assembled to produce executable code suitable for debugging. These functions have the following features:

Assembler:

— Subset of Series 32000 assembler
— Supports FPU by providing long and short format real number data initialization
— Generates listings to either a printer at the parallel port, or any RS232 device connected via serial port
— Symbolic definition of static base or PC segment

Debugger:

— Numerical arguments to commands can be in four bases: decimal, hex, long real and short real
— Program flow visually traced by displaying source line at all breakpoints or step stops
— Memory/register print or change commands
— Step-thru program commands: step "n" instructions, step while variable in range, step until variable reached

Editor:

— Command to insert, replace, delete, type lines
— Automatic line number maintenance
— Save and retrieve source from audio cassette recorder
— Upload/download to/from any RS232 equipped PC
— Debug data displayed by type command after assembly

User Program Run Time Support:

— Accessed via a supervisor call instruction
— Routines to do terminal I/O
— Printer driver access to parallel port
— Routine to convert binary value to ASCII string
— Routine to convert ASCII string to binary value
— Conversion in four bases: decimal, hex, long real and short real

As shipped with the DB32016, TDS provides on-board hardware confidence test routines. These are invoked following power on.

User Modes

The DB32016 can operate stand-alone, with no assistance from a host computer system. Optionally, the board can be operated in conjunction with a host, taking advantage of more powerful software development tools and I/O capabilities.

Figure 1-2 represents the most common variations in user modes.

1. Standalone Mode

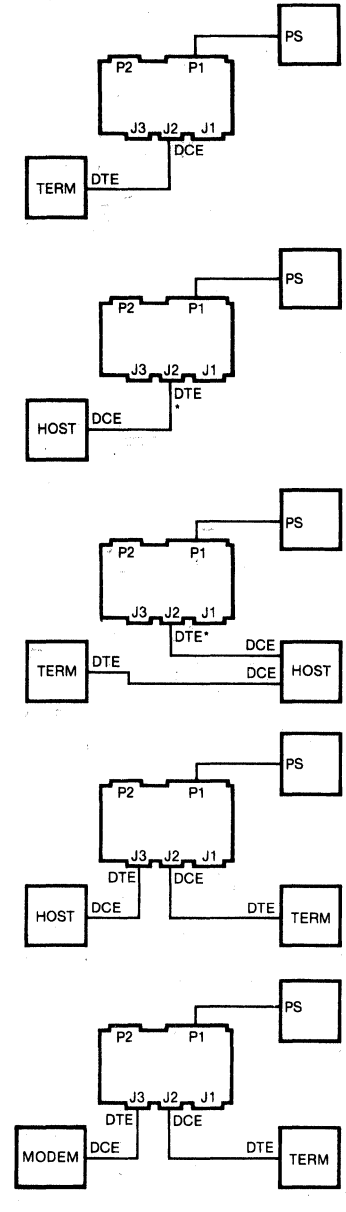
2. Host-assisted Modes

a) Standaside, single-user host

b) Standaside, multiuser host

c) Transparent, Local host

d) Transparent, Remote host



TL/R/7083–3

*requires reconfiguration

FIGURE 1-2. DB32016 User Modes

Stand-Along Mode (Factory Configuration)

From *Figure 1-2* it is clear that the stand-alone user mode is the most simplistic; requiring the least additional equipment. In this case, only an RS232C compatible terminal and power supplies for the DB32016 are required to achieve effective operation.

TDS (Tiny Development System) software is supplied in on-board PROM to support this user mode. TDS is used to edit, assemble, and execute small assembly language programs. In addition, TDS can control the DB32016's on-board I/O to provide cassette and printer interfaces; making the DB32016 a light duty development vehicle.

Host Assisted Modes

Referring to *Figure 1-2,* the DB32016 can be connected to another computer system, or host. In this case, the user will first develop Series 32000 software on the host system, then utilize RS232 communication to download the software to the DB32016. The DB32016 functions as a means of executing and debugging the software in a native environment.

Several development software packages are available for use in generating Series 32000 user programs.

Among them are:

- Pascal and C for VAX/VMS environments
- Pascal and C for VAX/UNIX environments

Host assisted modes require the TDS PROMs to be replaced by a PROM-based monitor program, compatible to the host development software. Monitor software is bundled with National's Series 32000 software packages. The monitor provides:

- a terminal handler, to control RS232 communications
- run-time environment, to permit execution of downloaded programs
- debugger execute module, to facilitate operation with the host's debugger software

The basic host assisted modes are:

- transparent
- standaside

The terms, transparent and standaside, may be visualized by observing the communication configuration in each mode. (Refer to *Figure 1-2*)

In transparent mode, the user's communication with the host is conducted through the DB32016; the DB32016 is transparent to the user. An advantage is that a single RS232 port on the host computer will support both the user's terminal, and the DB32016.

In standaside mode, the user communicates directly with the host; the DB32016 "stands aside". This mode is useful when the DB32016 is connected to single-user hosts. Optionally, standaside operation is possible with multi-user hosts, where two RS232 ports are available.

Specifications

Environment

The DB32016 is designed for operation in an office or laboratory environment. Avoid confining the DB32016 in a closed space, unless sufficient air flow is provided to ensure all components are operated within their specified temperature range.

- Temperature
  Operating 0°C to 55°C
  Non Operating −40°C to 75°C
- Humidity
  5% to 95% relative, non-condensing
- Altitude
  Operating 15,000 ft.
  Non Operating 25,000 ft.

Power Requirements

The DB32016 requires three, regulated DC voltages for operation:

+ 12 VDC, ±10%, 50 ma max
−12 VDC, ±10%, 50 ma max
+5 VDC, ±5%, 7.5A max

All power connections are made via P1. These connections are normally provided by a MULTIBUS compatible backplane. Optionally the user may elect to provide power, using one of the recommended connectors listed for P1.

Connectors

Local bus expansion (P2)-	CDC VPB01B30A00A2
	AMP PES-14559
	TI H311130
Parallel I/O (J1)-	3M 3415-001
	AMP 2-86792-3
Serial I/O (J2)-	3M 3462-0001 flat
	AMP 1-583715-1 round
Bus interface (P1)- and Power	SAE FUPH7212-86MTNE
	Viking 2KH43/9AMK12

Ordering Information

Model DB32016-110 (Order #NSV-32016P8T-10)

Includes NS32016-10 CPU, NS32082-10 MMU, NS32081-10 FPU, NS32202-10 ICU, and NS32201-10 TCU for 10 MHz operation.

All Models are shipped with:

- Two RS232 cable sets
- TDS: Tiny Development System User's Manual (Publication No. 420306440-001)
- DB32016 Development Board User's Manual (Publication No. 420310111-001)

Related Reference Material

Series 32000 Instruction Set Reference Manual (Customer Order No. NSP-INST-REF-M)

National Semiconductor Corporation

PRELIMINARY

ICM-3332 Integrated Computer Module

Features

- Series 32000® Computing Cluster
  - — NS32332 32-bit Central Processing Unit
  - — NS32201 Timing Control Unit
  - — NS32082 Memory Management Unit
  - — NS32081 Floating Point Unit
  - — NS32202 Interrupt Control Unit
- NS32016 I/O Processor
- Four asynchronous RS232C compatible serial ports
- Centronics compatible Printer Port
- High Speed 32/16-bit DMA "Mover" to/from System Memory
- Small Computer System Interface (SCSI) Port
- Time of day clock/calendar with battery backup
- 2–14 Mbytes of Random Access Memory with byte parity and "nibble" mode capability
- EPROM sockets for 16k to 128k bytes of EPROM
- UNIX V.3 available as an option

Overview

Integrated Computer Modules are designed to provide "supermicrocomputer" solutions to the OEM for applications in the office automation, workstation, graphics and process control areas. These Series 32000 based modules are not designed around any "Industry Standard Bus" but are, rather, designed as single board computers which access memory across a very fast private bus designed to enhance the Series 32000 operation. The modular concept employs a generic base CPU board and an IOP processor board used by all applications along with sufficient memory boards and I/O Controller boards (IOCs) to provide the required functionality. This improves Price/Performance: Price decreases because there is no need for backplanes and card cages; Performance increases because there is no need for wait states while the CPU is accessing memory. We do, however, retain the bus-like quality of having configuration options.

The ICM-3332 is a complete computer system contained on two to four 11.02 in. (300 mm) x 9.18 in. (233 mm) printed circuit boards. The CPU board and the IOP board are partitioned so that the ICM form factor can be maintained. The IOP controls the interface to the real world and handles all of the tasks associated with the movement of information. This allows the CPU board to dedicate most of its time to computing functions without the overhead of system management. This Integrated Computer Module will add the power of the Series 32000 processor chip set to National Semiconductor's line of board level microcomputer products. The CPU cluster (CPU, TCU, MMU, FPU, and ICU), 2 megabytes of DRAM and the 32-bit memory bus (MaxiBus) reside on the CPU board. The I/O Processor (IOP) board contains a MaxiBus interface to the CPU board, PROM sockets, serial interfaces, SCSI interface, parallel port and MiniBus to provide I/O capability to the outside world. MaxiBus is the 32-bit system bus which connects the NS32332, the 32-bit system memory and the IOP together. MiniBus is a 16-bit bus connecting the unique application boards (IOCs) to the 16-bit NS32016 bus on the IOP.

Two megabytes of RAM reside on the CPU board. Up to two memory boards can be added. Each of these boards can provide up to 6 megabytes of memory. This yields a maximum capacity for the system of 14 megabytes of physical memory. These boards are the same size as the CPU and IOP boards. They interface to the CPU unit via a 32-bit MaxiBus interface.

This Integrated Computer Module is designed to operate as a single board computer and communicates with memory via a high speed local bus (MaxiBus) which removes the overhead associated with the various industry standard buses. The CPU cluster operates at up to 10 MHz and requires no wait states when using Burst mode memory accessing.

Functional Description

SERIES 32000 FAMILY

The Series 32000 is a complete family of 32-bit CPUs, slave processors and peripherals. All processors, Memory Management Unit and Floating Point Unit are designed with 32-bit internal buses and registers to provide a true, two address architecture. The instruction set supports fast, highly compact, compiler code making the Series 32000 ideal for use with high level languages. The Memory Management Unit offers Demand-Paged Virtual Memory and fast on-chip address translation. This makes the Series 32000 an excellent engine for the UNIX™ operating system. The Floating Point Unit provides a hardware math unit which operates up to 100 times faster than software emulation. Since all CPUs in the Series 32000 line have a 32-bit internal architecture, the same software can be incorporated into all the systems within the family.

Space saving compiler code, hardware floating point support, demand-paged virtual memory and total software compatibility assure the speed, flexibility and cost effective operation needed in a system.

Series 32000 CPUs

The ICM-3332 utilizes the NS32332 as the Central Processing Unit. This CPU has a uniform linear 4 Gbyte addressing range and a full 32-bit internal architecture and implementation. Internal working registers, data paths and ALU are all 32 bits. This provides simpler handling of 32-bit data types and provides total compatibility with the next generation of 32-bit microprocessors. It also provides a full 32-bit data bus. The I/O Processor (IOP) for the ICM-3332 uses the NS32016 as its processor. This processor is identical to the NS32032 with respect to internal architecture, but has a 16-bit external data bus.

Functional Description (Continued)

The instruction set is fashioned after high level language instructions. Displacements can be 32 bits, ensuring that the instruction set will remain upward/downward compatible across the spectrum of Series 32000 CPU devices. With the Series 32000 CPU it is possible to add, subtract, multiply, divide or do any other operation with both operands in memory. This is a speed benefit for assembly code and is particularly needed in a high level language environment, since high level languages normally do memory-to-memory operations.

Restrictions frequently encountered, where only certain data types can be used with certain instructions, with particular addressing modes, with restricted sources and destinations, do not exist on Series 32000. ANY data type can be used with ANY instruction with ANY addressing mode with ANY source or destination. This makes the assembly programmer's task easier because there are no restrictions to track. With high level languages, the improvement becomes even more pronounced. Compilers produce very fast and efficient code when operating with a symmetrical machine.

NS32201 TCU (Timing Control Unit)

The NS32201 Timing Control Unit provides the two phase clock system, system control logic (read, write and data buffer enable) and cycle extension logic for the Series 32000 family.

Clocks provided are a two phase, non-overlapping 10 MHz clock, and 10 MHz and 20 MHz TTL compatible clocks.

NS32082 MMU (Memory Management Unit)

The increased requirement for development and execution of very large programs on cost effective systems has caused the development of virtual memory machines where programs larger than the physical memory space are routinely executed on the system. This is accomplished by maintaining the currently necessary portions of the program in the physical memory space and storing the remainder in mass storage. Portions of the program are switched in or out of physical memory or necessary. Since this switching operation is controlled by the operating system and is transparent to the user, it appears that greater memory is available than is actually the case. Thus, "virtual" memory.

The Series 32000 uses Demand-Paged Virtual Memory, with each page containing 512 bytes. Since the physical memory is a collection of these 512 byte pages, moving from physical memory to disk or from disk to physical memory becomes relatively simple.

The NS32082 Memory Management Unit provides hardware support for demand-paged virtual memory management. Its specific capabilities include fast dynamic address translation, protection on individual 512 byte pages and detailed status to assist an operating system in efficiently managing up to 16 Mbytes of physical memory.

For the purpose of address translation, memory is divided into 512 byte pages. A virtual address for the MMU is composed of two fields: a virtual page frame number and a nine bit offset. The offset is unchanged by the translation algorithm. The MMU translates the virtual page number to a physical page number according to page tables stored in memory.

The operating system and MMU exchange information on the status of the memory pages through a Page Table in physical memory and the protection level on that page. By manipulating the Page Tables, an operating system dynamically controls the mapping of virtual to physical addresses. In particular, the operating system may specify that references to certain pages should generate translation error aborts. This mechanism implements virtual memory management and protection.

The MMU utilizes memory called the Translation Buffer, which contains direct virtual-to-physical address mappings of the 32 most recently used pages. Entries in the Translation Buffer are allocated and replaced by the MMU. The programmer is not involved in the process. The Translation Buffer is a content-addressable memory. The virtual page frame number (the 15 high order bits of the virtual address) and the address space bit are compared to the entries in the buffer. If the virtual page frame number is present in the buffer, the mapped physical address is output immediately. This is the case approximately 98% of the time, so most address translations take only one additional clock cycle. When the virtual page frame number is not present in the buffer, a control line is set, indicating to the control block that the memory page tables should be referenced. When this occurs, the MMU gets the corresponding mapping from memory and replaces the least recently used entry in the Translation Buffer with the new mapping.

NS32081 FPU (Floating Point Unit)

The NS32081 Floating Point Unit implements the most commonly used floating point functions in hardware to yield a great increase in speed over the software routines which it replaces. For example, a multiply routine in software requires approximately 1500 μs to execute. With the FPU, that time is reduced to approximately 5 μs.

The Floating Point Unit functions as a slave processor to the NS32332 (or any other Series 32000 CPU). Its high speed instruction set is consistent with the full two-address architecture and powerful addressing modes of the Series 32000 microprocessor family.

The NS32081 FPU operates on two floating-point data types: single precision (32-bits) and double precision (64-bits). In addition, the FPU performs conversion between integer and floating-point data types. Integers are accepted and generated by the FPU as two's-complement values of byte (8-bit), word (16-bit) or double word (32-bit) in length.

Arithmetic operations include Add, Subtract, Multiply, Divide and Compare. Several Move and Convert instructions are also included.

NS32202 ICU (Interrupt Control Unit)

The NS32202 can operate in either of two data bus modes. In the 16-bit mode, eight hardware and eight software interrupts are possible. The 8-bit mode allows for up to 16 hardware interrupts with programmable priorities. ICM-3332 operates in the 8-bit mode.

The ICU additionally provides two 16-bit counters which can be concatenated under program control to provide a single 32-bit counter.

Functional Description (Continued)

CPU BOARD

The ICM-3332 provides a full 32-bit processing engine capable of operating with as much as 14 megabytes of physical memory with no wait-states when operating at 10 MHz in the Burst mode. The full CPU Cluster, using the NS32332 as the CPU, is implemented on the CPU board along with two megabytes of dynamic RAM with byte parity. An interface to a high performance 32-bit bus (MaxiBus) which allows communication with up to 12 megabytes of expansion memory is also included. The associated I/O processor (IOP) board interfaces with the CPU board and system memory across MaxiBus. Since the IOP controls all of the movement of data to/from any of the I/O devices, the CPU board is left unencumbered to dedicate itself to computing power.

The CPU board has no read-only memory. The processor board will be reset by the MaxiBus RESET signal driven by the IOP. The IOP has the ability to examine the CPU MaxiBus ready lines to determine how the system is configured. The IOP will then initialize memory on the CPU board and execution will begin from location 0.

INPUT/OUTPUT PROCESSOR (IOP)

The ICM-3332 Input/Output Processor (IOP) is the only unit within the ICM-3332 system which contains both a MaxiBus and a MiniBus interface. The MaxiBus interface provides the IOP with access to all ICM-3332 system memory and allows for the exchange of interrupts between the IOP and other members of the MaxiBus community to co-ordinate I/O traffic from the computing complex. The MiniBus interface allows the IOP to exchange data and control information with the I/O controllers (IOCs) on the MiniBus without interfering with the main processor functions. It is necessary to provide the ability for IOCs like graphics boards, local area network boards and serial output port boards to be able to move data and control information without the necessity to wait for the CPU board to grant permission to conduct I/O operations.

The IOP is a dedicated processor with full access to both the memory and the I/O resources of the ICM-3332 system. Its major function is to manage I/O traffic for the main processing complex, relieving the CPU from such functions as scatter-read and gather-write of virtual pages. Other important functions are control of the printer port, RS232 serial ports and the MiniBus interface to the IOCs.

The I/O processor has 512 kBytes of local memory with byte parity. IOCs may access this memory through the MiniBus Interface Controller (MBIC). The IOP processor is a NS32016. Both the IOP and the MBIC operate at 8 MHz. The IOP and MiniBus have a 16-bit architecture which provides low cost interfaces consistent with the requirements of modern peripherals and devices.

The interface between the 16-bit IOP data bus and the 32-bit MaxiBus is through a DMA mover circuit. The "mover" is set up and activated by a single instruction from the NS32016, allowing it to be efficient for blocks as small as one word or as large as 1 kByte. The data rate provided by the mover circuit is approximately 8 MBytes per second.

NS32016 Processor Complex

The IOP processor complex consists of an 8 MHz NS32016 processor, an NS32201 Timing Control Unit (TCU) and an NS32202 Interrupt Control Unit (ICU). The TCU provides an 8 MHz clock which provides the bus clock for the SCSI Controller, the mover, DRAM timing and the MiniBus bus clock.

IOP Memory

The IOP dynamic RAM consists of 512 kbytes with byte parity, which is accessed by the NS32016 with no wait-states. The mover utilizes the DRAM page mode for maximum transfer rate to/from MaxiBus memory.

System EPROM is also contained on the IOP board. Two EPROM sockets are provided on the IOP board. These sockets can be configured for a single pair of 2764, 27128, 27256 or 27512 devices for 16 kBytes to 128 kBytes of EPROM memory space.

Interrupt Control Unit

The Interrupt Control Unit (NS32202) provides the NS32016 with the ability to service 16 vectored interrupts. The ICU latches interrupts from the RS232 channels, the parallel port, the SCSI channel and the MBIC.

Serial Communications

Two 2681 Dual Asynchronous Receiver/Transmitters provide four independent, full duplex, asynchronous receiver/transmitter channels with software selectable baud rates to 19.2 kBaud. Two of these channels have DB-25 connectors to provide full modem support. The remaining two channels are provided with 6-wire modular telephone-type connectors (similar to the RJ-11 connectors on telephones) for use with terminals. These modular phone connectors allow for more efficient use of the available real estate and provide very cost effective cabling between ICM-3232 and the terminals.

Parallel Printer Port

One parallel port is provided. This port is configured as a parallel Centronics printer port designed to operate with the IBM PC type printer cable.

Time of Day Clock with Calendar

The ICM-3332 Time of Day Clock with Calendar uses the MM58274 Real Time Clock chip. Timekeeping from tenths of seconds to tens of years is available in independently accessible registers. The hour counter is programmable in either 12 or 24 hour operation. A lithium battery is provided to maintain clock operation when external power is removed.

Small Computer System Interface (SCSI)

The Small Computer System Interface (SCSI) is an Intelligent Peripheral Standard defined by the American National Standard Institute (ANSI). SCSI is derived from the Shugart Associates Standard Interface (SASI), which is based on the IBM I/O channel. ANSI has defined Command/Status Sets for five types of I/O devices:

Random Access— Rigid and Flexible Disk
Sequential Access— Start/Stop and Streamer Tape
Write Only Devices—Printers and Plotters
Processor Devices—Host to Host Communcations via SCSI
Network Devices— Host to Host Communications via LAN

The use of the ANSI Standard reduces system integration time by providing plug compatibility and command standards for peripherals and by allowing utilization of existing I/O drivers.

Functional Description (Continued)

Drivers have been written for ICM-3332 which allow SCSI to control the interface to rigid disk and to streamer tape. SCSI is implemented using an LSI SCSI controller chip and a state machine driven DMA channel into main memory. This controller supports the full SCSI protocol, including arbitration and recognition. The interface operates only as an initiator on the bus; the peripherals connected to the bus operate as targets.

This controller handles the physical path management between the NS32016 CPU (host) and the target disk or tape device including arbitration, selection, disconnection and reconnection. The operation starts when the host 32016 places a command descriptor block (as defined by SCSI) in memory. The SCSI controller acquires the SCSI bus during the arbitration and selection phase, supplies the command to the selected target, manages the transfer and interrupts the CPU when the target issues the command complete message. The SCSI does no interpretation of the commands or data transmitted between the host and the target; it only manages the transfer. The full power of the SCSI command structure is available to the host without the host suffering the overhead of the SCSI physical path management.

The SCSI can be connected to as many as seven target controllers, each of which can control up to eight subchannels operating concurrently. Each subchannel will control one separate logical device on the SCSI bus. This permits overlapping disk operations for read, write, seek, etc.

The SCSI controller is initialized by I/O commands. Subchannel target commands are controlled by I/O Control Blocks (IOCB) in main memory. Once the host sets up the command in memory, the I/O channel will complete the command and interrupt the host. The host then examines the IOCB(s) to check for errors or for proper completion.

Data transactions between dynamic RAM and a target are accomplished with direct memory access of 16-bit word transfers. This provides 1.5 Mbyte per second (full SCSI bandwidth) transfer in or out of dynamic RAM while still permitting the CPU access to the dynamic RAM an average of 50% of the time during target transfers.

MaxiBus

The IOP uses MaxiBus to move data to and from the 32332 address space. Information exchange between the IOP and the CPU board is by means of messages left in MaxiBus memory. The IOP can also both send and receive interrupts to other MaxiBus units to co-ordinate the message transfers.

To utilize the MaxiBus effectively, the IOP/MaxiBus interface meets the following design requirements:

1. To use the 16-bit IOP memory as a buffer (SCSI, LAN, etc.), the IOP must be able to rapidly move variable length blocks of data between the 16-bit IOP memory and the 32-bit MaxiBus memory.
2. The IOP must be able to send and receive interrupts across MaxiBus.
3. The IOP operation over MaxiBus should not lock up the IOP internal data structure. If the IOP is arbitrating for MaxiBus or in the middle of a MaxiBus transfer, the MBIC accesses to the IOP will continue to be serviced.

DMA Mover Circuit

When the NS32016 processor within the IOP requires a data transfer to/from system memory, the mover circuit is used. The mover is a special circuit which will move data between IOP memory and system memory without processor assistance. The mover circuit implements the interface between the 32-bit system bus and the 16-bit IOP local bus. It moves data to/from system memory at a rate of approximately 8 Mbytes/sec.

The mover will respond to a 32-bit write to the mover address range. The 32-bit address field contains both address and control fields. Direction of the move, type of information and the size of the block to be moved are all included in the 32-bit write. Two types of moves are allowed. Seven bits of the 32-bit field can be used to "count" the number of 16-bit words to be moved. Another bit in the field allows 1 kByte blocks to be moved.

The mover allows arbitration for the IOP bus until it has gained control of the MaxiBus. Then after every 16 cycles of the move operation, it will release the IOP bus to the MBIC if required. This insures that the I/O will not be locked up waiting for the mover. In the IOP priority scheme, the mover will be below the MBIC in access rights for the local bus. The mover maintains MaxiBus mastership during the entire move operation, even though it has momentarily released the local bus to another device.

The NS32016 is the only device that can use the mover. The mover can move only to/from IOP DRAM and MaxiBus or generate a MaxiBus interrupt. Direct movement from a MiniBus bus address to MaxiBus or other memory mapped addresses within the IOP are impossible.

When the move is in process, the IOP local memory and the MaxiBus (system) memory are operating simultaneously in a pipe-lined manner. The IOP memory is operated in page mode with a single MaxiBus cycle for every two IOP accesses to achieve complete overlap of memory timing. Two registers exist between the IOP bus and MaxiBus to allow conversion to/from 16- and 32-bit quantities.

MiniBus

MiniBus is a high performance 16-bit bus, which will support intelligent bus masters, while, at the same time maintaining a simple bus interface for non-intelligent I/O boards. The MiniBus Interface Controller (MBIC), a custom VLSI interface chip, has been developed to provide a single chip bus interface which provides full bus compatibility. This interface chip is available to the user who wishes to design an IOC to interface with the system via MiniBus.

MiniBus provides two distinct transfer disciplines:

Synchronous transfers which occur between units which implement the full bus discipline. Because of the sophistication of the bus, MBIC is used to facilitate this interface.

For asynchronous transfers, MiniBus allows for attachment of units which are directly controlled from a processor through a simple asynchronous bus cycle and dedicated interrupt lines. This allows units with simple interfaces to participate in the family without the need for a MBIC on their board.

Functional Description (Continued)

MiniBus is used to provide an interface between the CPU board and I/O Controllers (IOCs) such as LAN controllers and color graphics boards.

MiniBus Interface

The IOP serves as the gateway from MiniBus to the ICM-3332 system. MiniBus masters may direct interrupts to one another via the "software interrupt" capability provided by the MBIC and access each other's local memory if permitted by the settings of the internal registers of the MBIC.

Data movement to/from the IOP across MiniBus is accomplished by direct access from the active I/O Controller (IOC) into an assigned buffer area within the IOP memory. All processors on MiniBus then move data at their own rate with minimum interference to IOP processing power. Command and status information are also accessed directly by the target IOC.

The MBIC resides on the IOP local bus and manages the entire MiniBus interface. When a reference is made to IOP local memory from another MiniBus master, the MBIC arbitrates for the local IOP bus. Once the arbitration grants the bus, the MBIC will take control of the local IOP bus and access the IOP DRAM as required.

Memory Expansion

The ICM-3332 memory board contains the expansion DRAM for the ICM-3332 Integrated Computer Module. This board is populated by 216 DRAM devices organized with a 36-bit wide data bus which allows for 32 bits of data plus byte parity. The board is designed to accept 256k 150 ns dynamic RAM devices which operate with the Series 32000 CPU cluster at 10 MHz with no wait states for DRAM access when operating in Burst mode. Each board contains 6 Mbytes of memory. It is possible to use up to two boards with each system to provide a total available memory of 14 Mbytes (2 Mbytes are on the CPU board).

OPERATING SYSTEM

AT&T UNIX V.3 is available for use with ICM-3332. This operating system provides optimal use of the Series 32000 architecture. It supports demand-paged virtual memory and offers Job Control, File and Record Locking and Streams. "C" and FORTRAN 77 compilers and the Series 32000 assembler and all 32000 language support tools are also provided. Reconfigurable binary drivers have been written for the following:

SCSI using the EMULEX disk controller model MD01.

SCSI using the EMULEX tape controller model MT02.

4 RS232C serial ports.

Parallel Printer port.

This operating system is available on streamer tape with QIC 24 tape format. It will be delivered as an unbundled binary system. The source for the above drivers is provided for use as examples for those wishing to generate their own drivers. A binary license and distribution agreement is required.

Specifications

Physical:

Width:	11.02 in. (280 mm)
Height:	9.18 in. (233 mm)
Depth:	0.80 in. (20 mm)
Weight:	
CPU Board:	TBS
IOP Board:	TBS
Memory Board:	TBS

Connectors

Board/ Connector	# of Pins	Function	Connector Type	Mating Connector
CPU J1*	96	MiniBus	Direct Connect**	Direct Connect**
CPU J2*	96	MaxiBus	Direct Connect**	Direct Connect**
MEM J1*	96	MiniBus	Direct Connect**	Direct Connect**
MEM J2*	96	MaxiBus	Direct Connect**	Direct Connect**
IOP J1*	96	MiniBus	Direct Connect**	Direct Connect**
IOP J2*	96	MaxiBus	Direct Connect**	Direct Connect**
IOP J3	11	Power/Reset		
IOP J4	50	SCSI	50-pos 0.10 ctr	3M 3435-6000
IOP P1	36	Printer	DB25 female	DB25 male
IOP P2	6	RS232	Telco 6-wire s	Telco 6-wire p
IOP P3	6	RS232	Telco 6-wire s	Telco 6-wire p
IOP P4	36	RS232	DB25 male	DB25 female
IOP P5	36	RS232	DB25 male	DB25 female
V1–V4	1	Power	Banana Plug**	Banana Jack**
G1–G4	1	Ground	Banana Plug**	Banana Jack**

*The Direct Connect Fixtures consist of a 96-pin DIN connector and custom shroud to adjust spacing. The connector is attached to the board with the female side on top. The pins from the bottom of the connector provide a male DIN connector which extends into the shroud. The boards then "stack" together using the MiniBus and MaxiBus Direct Connects and the +5 V_{DC} and Ground banana connectors to electrically connect the boards. This eliminates the need for a backplane and provides the CPU and IOP direct access to the entire memory array.

**See the ICM-DESIGN-KIT under Order Information.

Specifications (Continued)

RAM ADDRESSING

System Memory

Configuration	RAM Size	Address Space
CPU Board	2 MB	0 to 2FFFFF
CPU + 1 Expansion Bd	8 MB	0 to 7FFFFF
CPU + 2 Expansion Bds	14 MB	0 to FFFDFF

IOP Memory

RAM Size: 256 kBytes

Address Space: 0 to 3FFFFF (see IOP Memory Map)

ROM Size: 128 kBytes Max.

ROM Addressing: At power up or reset, the IOP hardware will power up with the ROM address space starting at location zero. The first CPU access with the most significant address bit (A23) set to 1 clears the Shadow EPROM and EPROM then executes starting from address location 800000.

MEMORY MAP

Main Memory

Address Range, HEX	Read or Write	Location
000000-01FFFFF	Read, Write	CPU Local memory
020000-07FFFFF	Read, Write	MaxiBus Expansion Memory
080000-09FFFFF	Read, Write	CPU Local Memory
0A0000-0FFFDFF	Read, Write	MaxiBus Expansion Memory

IOP Memory

Address Range, HEX	Read or Write	Function
000000-3FFFFF	Read, Write	IOP RAM, Bootload EPROM
400000-7FFFFF	Write	Mover Circuit
800000-BFFFFF	Read EPROM	IOP EPROM
	Read, Write	I/O Control Registers
C00000-FFBFFF	Read, Write	MBIC/MiniBus Space

Environmental

Operating Temperature:	0°C to +55°C (+32°F to +131°F)
Storage Temperature:	−30°C to +55°C (−22°F to +131°F)
Thermal Shock:	10 degrees/min.
Relative Humidity:	0 to 90%, noncondensing
Air Flow:	A minimum air velocity of 170 ft. per minute across the board is required.

Power Requirements

	+5 V_{DC} ±5%		+12 V_{DC} ±5%	−12 V_{DC} ±5%
	Typ	Max	Typ	Typ
CPU Board	7.8 A	9.4A		
IOP Board	6.0 A	7.2 A	0.06 A	0.06 A
Memory Bd	1.6 A	1.9 A		

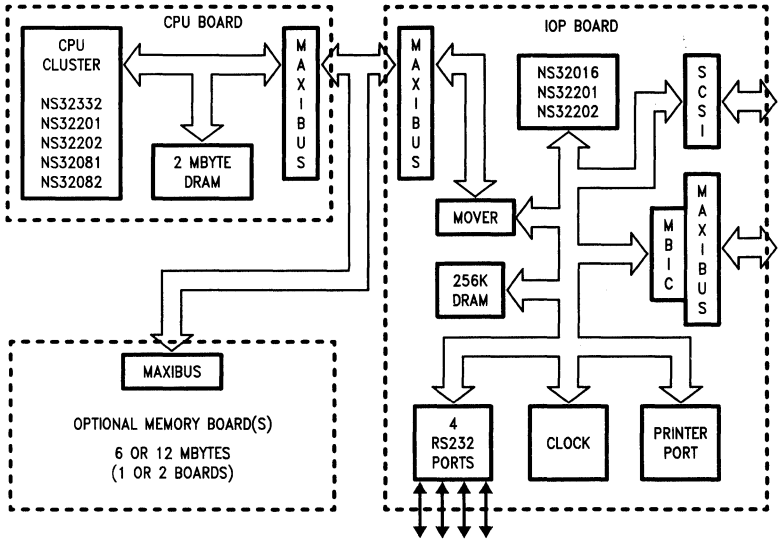
Ordering Information

ICM-3332-CPU	The CPU board contains the full CPU cluster with an NS32332 CPU along with 2 MB of memory.
ICM-3332-IOP	The IOP board contains the NS32016 CPU, NS32201 TCU, NS32202 ICU, 4 RS232 serial ports, a printer port, SCSI interface, MiniBus and MaxiBus.
ICM-3332-6MEM	ICM-3332 Memory Expansion Board with 6 MB of 150 ns DRAM.
ICM-3332-RD	This is the ICM-3332 CPU and IOP boards, UNIX V.2, monitor/bootloader EPROM, a full set of manuals, including AT&T UNIX manuals and NSC manuals, the cables and adapters required for one terminal operation and the binary license. The NSC manuals provided are the Hardware Reference Guide and the ROM Monitor Manual. This is the all inclusive starter kit.
ICM-CBL-TELCO	Cable, Telco, 6-wire, 7 ft.
ICM-CON-MAL	Adapter, Telco to DB25 male connector (configured as DTE).
ICM-CON-FEM	Adapter, Telco to DB25 female connector (configured as DTE).
ICM-3332-SYSV	System V/Series 32000 UNIX V.2 Operating System ported to ICM-3332 hardware, EPROM set and EPROM User's Guide.
ICM-3332-MON	Monitor EPROMs for the ICM-3332. This EPROM set is included in the ICM-3332-SYSV and ICM-3332-RD packages.
ICM-POWER-KIT	Banana Power connectors to allow the user to connect power cables to the ICM boards (10 sets).
ICM-DESIGN-KIT	Male and Female Power Connectors and DIN connector spacers (all of the special design parts) needed for the user to build boards (10 sets).

Documentation

ICM-3332-M	ICM-3332 Hardware Reference Manual
ICM-3332-MON-M	ICM-3332 Monitor User's Guide
ICM-3332-SV-MS	UNIX V.3 Manual Set

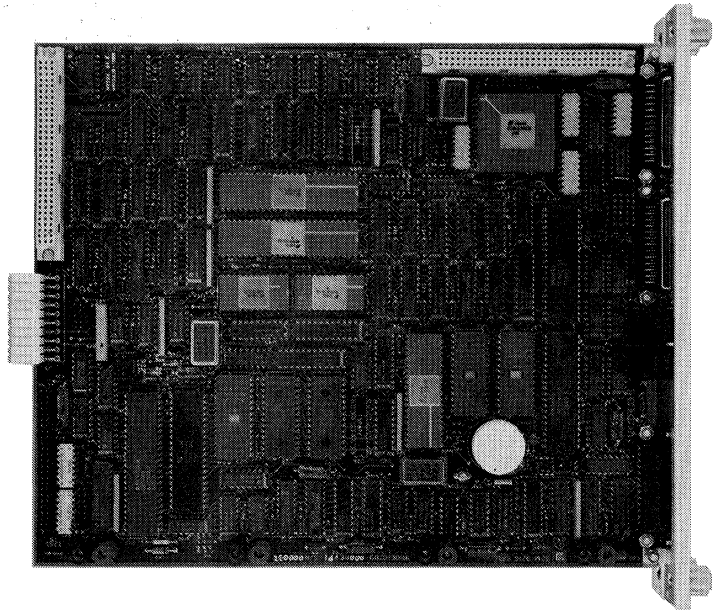
ICM-3332 Block Diagram



TL/T/8778–1

National Semiconductor Corp.

ICM-3216
Integrated Computer Module



TL/T/8653–1

- **Series 32000® Chip Set**
  - **— NS32016 32/16-bit Central Processing Unit**
  - **— NS32201 Timing Control Unit**
  - **— NS32082 Memory Management Unit**
  - **— NS32081 Floating Point Unit**
  - **— NS32202 Interrupt Control Unit**
- **Four asynchronous RS232C compatible serial ports**
- **CENTRONICS compatible Printer Port**
- **Full SCSI Interface**
- **MiniBus I/O Interface**
- **Time of day clock/calendar with battery backup**
- **1–8 Mbytes of Random Access Memory**
- **EPROM sockets for 16k to 128k bytes of EPROM**
- **System V/Series 32000 available as an option**

Overview

Integrated Computer Modules are designed to provide "supermicrocomputer" solutions to the OEM for applications in the office automation, workstation, graphics and process control areas. These Series 32000 based modules are not designed around any Industry Standard Bus but are, rather, designed as single board computers which access memory across a very fast private bus designed to enhance the Series 32000 operation. The modular concept employs a generic base board used by all the units along with a personality board to provide the required functionality. This improves Price/Performance: Price decreases because there is no need for a backplane; Performance increases because there are no wait states caused by the bus. The backplane-like quality of having configuration options is, however, maintained.

Overview (Continued)

The ICM-3216 is a complete computer system contained on two 11.02 in. (300 mm) x 9.18 in. (233 mm) printed circuit boards. This Integrated Computer Module adds the power of the Series 32000 processor chip set to National Semiconductor's line of board level microcomputer products. The CPU cluster (CPU, TCU, MMU, FPU, and ICU), PROM sockets, serial interfaces, address mapping logic, SCSI interface, parallel port, memory interface and MiniBus interface reside on the CPU unit.

Up to two memory boards can be configured with the CPU unit to provide between one and eight million bytes of memory. The memory board(s) interface to the CPU unit via a "Direct Connect" local memory bus. The system operates at 10 MHz with no wait states for normal memory access.

This Integrated Computer Module is designed to operate as a single board computer and communicates with memory via a high speed local bus which circumvents the arbitration and speed problems associated with the various industry standard buses. A Small Computer Systems Interface (SCSI) provides the interface to hard disk and tape units. SCSI is implemented by using an LSI SCSI device and a Z80B® microprocessor. This provides complete control of the interface by the NS32016 CPU while freeing it from controlling the associated details of the transfer. Data transfer between the dynamic RAM and mass storage is accomplished with direct memory access of 16-bit words.

Functional Description

CPU Cluster

The Series 32000 is a complete family of 32-bit CPUs, slave processors and peripherals. All processors, Memory Management Unit and Floating Point Unit are designed with 32-bit internal buses and registers to provide a true, two address architecture. The instruction set supports fast, highly compact, compiler code making the Series 32000 ideal for use with high level languages. The Memory Management Unit offers Demand-Paged Virtual Memory and fast on-chip address translation. This makes the Series 32000 an excellent engine for the UNIX™ operating system. The Floating Point Unit provides a hardware math unit which operates up to 100 times faster than software emulation. Since all CPUs in the series 32000 line have a 32-bit internal architecture, the same software can be incorporated into all the systems within the family.

Space saving compiler code, hardware floating point support, demand-paged virtual memory and total software compatibility assure the speed, flexibility and cost effective operation needed in a system.

NS32016 CPU (Central Processing Unit)

The ICM-3216 utilizes the NS32016 as the Central Processing Unit. This CPU has a uniform linear 16 Mbyte addressing range and a full 32-bit internal architecture and implementation. Internal working registers, data paths and ALU are all 32 bits. This provides simpler handling of 32-bit data types and provides total compatibility with the next generation of 32-bit microprocessors.

The instruction set is fashioned after high level language instructions. Displacements can be 32 bits, ensuring that the instruction set will remain upward/downward compatible across the spectrum of Series 32000 CPU devices. With the Series 32000 CPU it is possible to add, subtract, multiply, divide or do any other operation with both operands in memory. This is a speed benefit for assembly code and is particularly needed in a high level language environment, since high level languages nearly always do memory-to-memory operations.

Restrictions frequently encountered, where only certain data types can be used with certain instructions, with particular addressing modes, with restricted sources and destinations, do not exist on Series 32000. ANY data type can be used with ANY instruction with ANY addressing mode with ANY source or destination. This makes the assembly programmer's task easier because there are no restrictions to track. With high level languages, the improvement becomes even more pronounced. Compilers produce very fast and efficient code when operating with a symmetrical machine.

NS32201 TCU (Timing Control Unit)

The NS32201 Timing Control Unit provides the two phase clock system, system control logic (read, write and data buffer enable) and cycle extension Logic for the Series 32000 family.

Clocks provided are a two phase, non-overlapping 10 MHz clock, and 10 and 20 MHz TTL compatible clocks.

When operating with slow peripherals, Cycle Extension is required. TCU Cycle Extension features include programmable wait state inputs, a "slow" cycle to accommodate slower peripherals and a cycle hold to allow additional time for arbitration before generating control signals.

NS32082 MMU (Memory Management Unit)

The increased requirement for development and execution of very large programs on cost effective systems has caused the development of virtual memory machines where programs larger than the physical memory space are routinely executed on the system. This is accomplished by maintaining the currently necessary portions of the program in the physical memory space and storing the remainder in mass storage. Portions of the program are switched in or out of physical memory as necessary. Since this switching operation is controlled by the operating system and is transpar-

Functional Description (Continued)

ent to the user, it appears that greater memory is available than is actually the case. Thus, "virtual" memory.

The Series 32000 uses demand-paged virtual memory, with each page containing 512 bytes. Since the physical memory is a collection of these 512 byte pages, moving from physical memory to disk or from disk to physical memory becomes relatively simple.

The NS32082 Memory Management Unit provides hardware support for demand-paged virtual memory management. Its specific capabilities include fast dynamic address translation, protection on individual 512 byte pages and detailed status to assist an operating system in efficiently managing up to 16 Mbytes of physical memory.

For the purpose of address translation, memory is divided into 512 byte pages. A virtual address for the MMU is composed of two fields: a virtual page frame number and a nine bit offset. The offset is unchanged by the translation algorithm. The MMU translates the virtual page number to a physical page number according to page tables stored in memory.

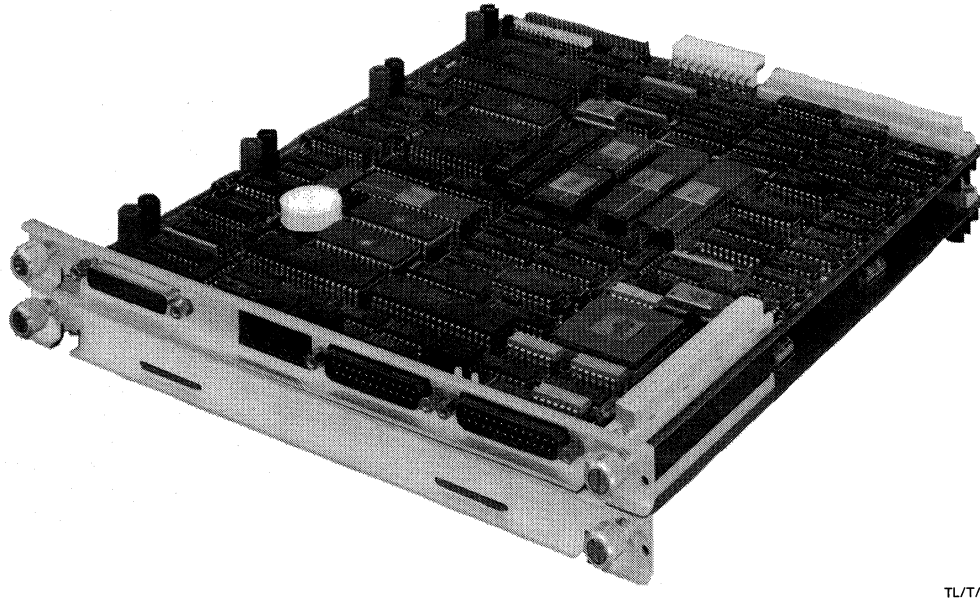
The operating system and MMU exchange information on the status of the memory pages through a Page Table in physical memory and the protection level on that page. By manipulating the Page Tables, an operating system dynamically controls the mapping of virtual to physical addresses. In particular, the operating system may specify that references to certain pages should generate translation error aborts. This mechanism implements virtual memory management and protection.

The MMU has an internal memory called the Translation Buffer, which contains direct virtual-to-physical address mappings of the 32 most recently used pages. Entries in the Translation Buffer are allocated and replaced by the MMU. The programmer is not involved in the process. The Translation Buffer is a content-addressable memory. The virtual page frame number (the 15 high order bits of the virtual address) and the address space bit are compared to the entries in the buffer. If the virtual page frame number is present in the buffer, the mapped physical address is output immediately. This is the case approximately 98% of the time, so most address translations take only one additional clock cycle. When the virtual page frame number is not present in the buffer, a control line is set, indicating to the control block that the memory page tables should be referenced. When this occurs, the MMU gets the corresponding mapping from memory and replaces the least recently used entry in the Translation Buffer with the new mapping.

NS32081 FPU (Floating Point Unit)

The NS32081 Floating Point Unit implements the most commonly used floating point functions in hardware to yield a great increase in speed over the software routines which it replaces. For example, a multiply routine in software requires approximately 1500 microseconds to execute. With the FPU, that time is reduced to approximately five microseconds.

The Floating Point Unit functions as a slave processor to the NS32016 (or any other Series 32000 CPU). Its high speed instruction set is consistent with the full two-address architecture and powerful addressing modes of the Series 32000 microprocessor family.



TL/T/8653–2

Functional Description (Continued)

The NS32081 FPU operates on two floating-point data types: single precision (32-bits) and double precision (64-bits). In addition, the FPU performs conversions between integer and floating-point data types. Integers are accepted and generated by the FPU as two's-complement values of byte (8-bit), word (16-bit) or double word (32-bit) in length.

Arithmetic operations include Add, Subtract, Multiply, Divide and Compare. Several Move and Convert instructions are also included.

NS32202 ICU (Interrupt Control Unit)

The NS32202 ICU is the interrupt controller for the Series 32000 microprocessor family. The ICM-3216 uses the ICU in the eight bit mode which allows for up to 16 hardware interrupts with programmable priorities. Four lines are used with the serial ports, one with the MiniBus controller, one with the SCSI port, one with the printer, and one with the real time clock. The remaining eight lines are available for use by the programmer.

Serial Communications

Two 2681 Dual Asynchronous Receiver/Transmitters provide four independent, full duplex, asynchronous receiver/transmitter channels with software selectable baud rates to 19.2 Kbaud. Two of these channels use the DB25 connector and implement the full set of RS232 signals:

- Tx Data
- Rx Data
- Request to Send
- Clear to Send
- Data Set Ready
- Data Terminal Ready
- Carrier Detect

The remaining RS232 channels use six wire modular telephone jacks, which are suitable for use with most terminals, and implement the following signals:

- I_SEND_DATA
- U_SEND_DATA
- I_MAY_SEND
- U_MAY_SEND

Parallel Printer Port/Input Port

One parallel port is provided. Direction of the port is software configurable. The CPU controls operation by accessing the parallel port command, status and data registers. The system is shipped with the port configured as a parallel Centronics printer port designed to operate with the IBM PC type printer cable.

Time of Day Clock with Calendar

The ICM-3216 Time of Day Clock with Calendar uses the MM58274 Real Time Clock/Calendar chip. It functionally consists of 13 4-bit binary coded decimal (BCD) counters ranging from tenths of seconds to tens of years, plus a day of the week counter. These counters are updated synchronously every tenth of a second. Leap years are automatically registered. Time can be programmed for the 12 hour mode (with AM and PM) or the 24 hour mode. Clock/Calendar Addressing is shown in the following table.

Clock/Calendar Addressing

Address	Read or Write	Function
A00000	READ	Control register status
	WRITE	Control register command
A00002	READ,WRITE	Seconds-tenths
A00004	READ,WRITE	Seconds-units
A00006	READ,WRITE	Seconds-tens
A00008	READ,WRITE	Minutes-units
A0000A	READ,WRITE	Minutes-tens
A0000C	READ,WRITE	Hours-units
A0000E	READ,WRITE	Hours-tens
A00010	READ,WRITE	Days-units
A00012	READ,WRITE	Days-tens
A00014	READ,WRITE	Months-units
A00016	READ,WRITE	Months-tens
A00018	READ,WRITE	Years-units
A0001A	READ,WRITE	Years-tens
A0001C	READ,WRITE	Day of week
A0001E	READ,WRITE	Clock setting register

The clock/calendar circuitry has battery back-up to keep the time current when no external power is supplied.

Small Computer System Interface (SCSI)

The Small Computer System Interface (SCSI) is an Intelligent Peripheral Standard defined by the American National Standard Institute (ANSI). SCSI is derived from the Shugart Associates Standard Interface (SASI), which is based on the IBM I/O channel. ANSI has defined Command/Status Sets for five types of I/O devices:

Random Access—	Rigid and Flexible Disk
Sequential Access—	Start/Stop and Streamer Tape
Write Only Devices—	Printers and Plotters
Processor Devices—	Host to Host Communications via SCSI
Network Devices—	Host to Host Communications via LAN

The use of the ANSI Standard reduces the system integration time by providing plug compatibility and command standards for the peripherals and allowing utilization of existing I/O drivers.

ICM-3216 uses SCSI to control the interface to rigid disk and to streamer tape. SCSI is controlled by an I/O channel controller supporting the full asynchronous SCSI protocol, including arbitration and reconnection. The interface operates only as an initiator on the bus; the peripherals connected to the bus operate

Functional Description (Continued)

as targets. The interface is implemented using a Z80B microprocessor and an NCR 5385E SCSI device.

This controller handles the physical path management between the NS32016 (host) and the target disk or tape device including arbitration, selection, disconnection and reconnection. The operation starts when the host NS32016 places a command descriptor block (as defined by SCSI) in memory. The I/O channel controller acquires the SCSI bus during the arbitration and selection phase, supplies the command to the selected target, manages the transfer and interrupts the CPU when the target issues the command complete message. The I/O channel controller does no interpretation of the commands or data transmitted between the host and the target; it only manages the transfer. The full power of the SCSI command structure is available to the host without the host suffering the overhead of the SCSI physical path management.

The SCSI can be connected to as many as seven target controllers, each of which can control up to eight devices. Up to eight devices may be operated concurrently. This permits overlapping disk operations for read, write, seek, etc.

The I/O channel controller is initialized by I/O commands. Subchannel target commands are controlled by I/O Control Blocks (IOCB) in main memory. Once the host sets up the command in memory, the I/O channel completes the command and interrupts the host. The host then examines the IOCB(s) to check for errors or for proper completion.

Data transaction between dynamic RAM and a target are accomplished with direct memory access of 16-bit word transfers. This provides 1.5 Mbyte per second (full SCSI bandwidth) transfer in or out of dynamic RAM while still permitting the CPU access to the dynamic RAM an average of 50% of the time during target transfers.

MiniBus

MiniBus is a high performance, synchronous 16-bit bus with full support for multiprocessors, complete with parity on address and data lines and physical addressing of all MiniBus masters (8 maximum). The MiniBus interface is provided through the MiniBus Interface Controller (MBIC). This single CMOS LSI compatible component attaches directly to the NS32016 bus and provides the complete bus interface.

MiniBus is not designed to compete with any of the 32-bit buses whose purpose is to satisfy the bandwidth requirements of multiple 32-bit processors. Rather, it is intended to provide a low cost, low power, low real estate bus with similar multiprocessor system characteristics on a smaller scale (16 bits of data, 8 masters).

Virtually all the LSI peripheral chips to support LAN's disks, terminals, floppies, tapes, GPIB and other system requirements are designed in 8- or 16-bit widths and are cheaper and more convenient to implement on a 16-bit bus when bandwidth permits. MiniBus is designed for these types of applications and provides very cost effective solutions in these situations. The MBIC is available to the customer for design of custom boards.

Memory

The ICM-3216 memory board contains the dynamic RAM for the ICM-3216 Integrated Computer Module. This board is populated with 144 RAM devices organized in an 18-bit wide data bus which allows for 16 bits of data plus byte parity. The board is designed to accept 256k 150 ns dynamic RAM devices which operate with the Series 32000 CPU cluster at 10 MHz with no wait states for RAM access. Each board can contain 1 Mbyte to 4 Mbytes of memory with 256k RAMs installed. The system design permits a maximum of two memory boards, each containing the same RAM device type, allowing 1, 2, 4 or 8 Mbyte systems to be configured.

Interface between the CPU board and memory board(s) is provided by two connectors. The MiniBus connector is not used on the memory boards except to provide structural stability for the Integrated Computer Module. The memory port is proprietary and is designed to enhance NS32016 operation with the Integrated Computer Module and contains all necessary address, data, control and timing signals to provide this optimized operation.

Two EPROM sockets are provided on the CPU board. These sockets can be configured for a single pair of 2764, 27128, 27256 or 27512 devices for 16 Kbytes to 128 Kbytes of ROM memory space. When operating with System V/Series 32000, the boot EPROM provided begins at address location 00 on power up or reset and remains in this location until the first time address bit A23 is set to 1. At this time, this EPROM address is changed to start at address 800000H. The boot EPROM contains the initialization program that executes upon power-up or reset and a stand-alone ROM Monitor for ICM-3216. The following functions are provided:

- Display the sign-on message upon power-up or reset
- Perform on-board diagnostics

Initialization

Upon power-up, the following ICM-3216 parameters are initialized:

The console channel baud rate is set to 9600 baud.

The character structure is set to 1 stop bit and 8-bits, no parity.

A unique value is written into all 1 Mbyte increments, then read back again. Addresses wrap around if the system is less than fully configured.

Functional Description (Continued)

The MiniBus Interface Controller is initialized by unlocking the interrupt circuitry so that a power fail or Bus Conflict NMI can get through.

The Memory Management Unit, Floating Point Unit, Interrupt Control Unit, MiniBus Interface Controller registers and the parallel port are initialized.

Power On Confidence Checks

Various power-on confidence checks are performed to insure that certain basic functions of ICM-3216 are working correctly. Memory and the parity circuits are tested as follows:

Parity Circuit: Data with even parity is written out and then read back with odd parity. A check is made for a Non-Maskable Interrupt (NMI) due to parity errors. This test is performed for two different bytes to insure that the correct byte and appropriate RAM is indicated.

Memory: A hexadecimal pattern is written to all memory locations. It is then read back, checking for parity errors. The system cleans up by writing zeros to all memory locations.

- Format Disk

  This allows a user to format a raw disk for use with the system. The disk must previously have been connected to the SCSI channel. The user then follows a menu to supply format parameters.

- Load Disk from Tape Drive

  System V/Series 32000 is supplied to the user on a 30 Mbyte (450 ft) tape cartridge. The user copies from tape to disk using the monitor's copy command. The ICM-3216 system must be configured with a console, a disk drive and a tape drive. At least 28 Mbytes of disk space is required.

- Execute the Monitor Program

  The monitor operates in one of two modes, controlled by a software switch:

  User Mode: the monitor reads command inputs from a terminal and executes the command with a Line Feed (<LF>).

  System Mode: input may come from a terminal as above or from a program generating command strings. This mode is meant to be used by programs sending commands to the monitor without human intervention.

  Monitor commands include Download Memory, Examine Memory, Modify Memory, Print Registers, Change Registers, Write Pattern, Set Breakpoint, Test Memory, Display Memory Size, Single Step and GO.

- Enter Operating System

After the system has completed the power on or reset sequence, it comes up in the monitor mode. After System V/Series 32000 has been installed, the monitor command B <cr> will cause the operating system to boot.

Operating System

ICM-3216 uses System V/Series 32000, a validated version of AT&T's UNIX System V. System V/Series 32000 is a powerful, multi-tasking, multi-user operating system with the following key features:

- Demand Paged Virtual Memory
- Hierarcical file system
- Source Code Control System (SCCS)
- UNIX to UNIX copy (uucp)
- Record and File Locking and High Level Language Programming
- C, FORTRAN 77 and (optional) Pascal compilers

Reconfigurable binary drivers are available for the following:

- SCSI using the EMULEX disk controller model MD01.
- SCSI using the EMULEX tape controller model MT02.
- 4 RS232C serial ports.
- Parallel Printer port.

This operating system is available on streamer tape as an unbundled binary system. The source for the above drivers is provided for use as examples for those wishing to generate their own drivers. A binary license and distribution agreement is required.

Specifications

Physical

Width:	11.02 in. (280 mm)
Height:	9.18 in. (233 mm)
Depth:	0.80 in. (20 mm)
Weight:	
CPU Board:	26 oz.
Memory Board:	24 oz.

Specifications (Continued)

Connectors

Board/ Connector	# of Pins	Function	Connector Type	Mating Connector
CPU J1*	96	MiniBus	Direct Connect	Direct Connect
CPU J2*	96	Memory	Direct Connect	Direct Connect
CPU J3	10	PS Control	10-pos 0.156 ctr	AMP 1-641150-0
CPU J4	10	Reset/LED	10-pos 0.100 ctr	3M 3473-6000
CPU J5	50	SCSI	50-pos 0.100 ctr	3M 3435-6000
CPU P1	36	Printer	DB25 female	DB25 male
CPU P2	6	RS232	RJ19 6-wire s	RJ19 6-wire p
CPU P3	6	RS232	RJ19 6-wire s	RJ19 6-wire p
CPU P4	36	RS232	DB25 male	DB25 female
CPU P5	36	RS232	DB25 male	DB25 female
MEM J1*	96	MiniBus	Direct Connect	Direct Connect
MEM J2*	96	Memory	Direct Connect	Direct Connect
V1–V4	1	Power	Banana Plug	Banana Jack
G1–G4	1	Ground	Banana Plug	Banana Jack

The Direct Connect Fixtures consist of a standard 96-pin female DIN connector and a shroud to adjust spacing. The connector is attached to the board with the female side on the top. The pins from the bottom of the connector provide a male DIN connector which extends into the shroud. The boards then "stack" together using the MiniBus and Memory Direct Connects to electrically interconnect the boards. This eliminates the need for a backplane and provides the CPU direct access to the entire memory array.

EPROM Addressing

At power-up and after a power reset, the MSHADOW flip-flop is asserted. In this condition, EPROM can be read from a starting address of 0 or 800000H. The first CPU access with the most significant address bit (A23) set to 1 clears MSHADOW. At this time, EPROM is available only from a starting address of 800000H.

Memory Map

Address Range, Hex	Read or Write	Function
000000–01FFFF	READ, WRITE	Dynamic RAM, bootload EPROM
020000–7FFFFF	READ, WRITE	Dynamic RAM only
800000–81FFFF or 900000–91FFFF	READ	Bootload EPROM
A00000–A0001E	READ, WRITE	Clock/Calendar
A00020–A0003E	READ, WRITE	Serial Ports 1 and 2
A00040–A0005E	READ, WRITE	Serial Ports 3 and 4
A00080–A00082	READ, WRITE	Parallel Port
A000A0	READ	I/O channel status register
	WRITE	I/O channel command register
A000C0	READ	NMI status
	WRITE	MiniBus/Hold
A000C2-A000C8	WRITE	LED Array
A000CA	WRITE	LED MiniBus Reset
A000CC	WRITE	Even/odd parity select
A000CE	WRITE	Set parity enable
A000E0	READ	Enable NMI
	WRITE	Registers MHL and MCL
C00000–FDFFFF	READ, WRITE	MiniBus memory address 000000–3DFFFF
FE0000–FEFFFF	READ, WRITE	MiniBus 8-bit I/O access
FF0000–FF7FFF	READ, WRITE	MiniBus 16-bit I/O access
FFFE00–FFFFFF	READ, WRITE	Interrupt Control Unit

Specifications (Continued)

Environmental

Operating temperature: 0°C to +55°C (+32°F to +131°F)

Storage Temperature: −30°C to +55°C (−22°F to +131°F)

Relative Humidity: 0 to 90%, noncondensing

Power Requirements

	+5 VDC ±5%		+12 VDC ±5%	−12 VDC ±5%
	Typ	Max	Typ	Typ
CPU Board	5.2A	6.2A	57 mA	57 mA
1 MB Mem Bd	1.25A	1.5A	—	—
2 MB Mem Bd	1.0A	1.2A	—	—
4 MB Mem Bd	1.25A	1.5A	—	—

Reliability

A comprehensive three phase testing program ensures that all products conform completely to specifications throughout their lifetime.

The first phase is an in-circuit test performed on the board after it is assembled. The circuitry on the board is exercised with a set of tests designed to uncover any manufacturing induced malfunctions.

The second phase is functional testing. The PCB is put in an oven and exercised at an elevated temperature with a full set of diagnostic programs. This phase is designed to eliminate infant mortality and to ensure board functionality over environmental as well as operational extremes.

The third phase is the system configuration test. The PCB is connected to a standard system and is tested with system level software. This test is designed to ensure the board's functionality in a system level environment.

The testing program ensures complete product functionality.

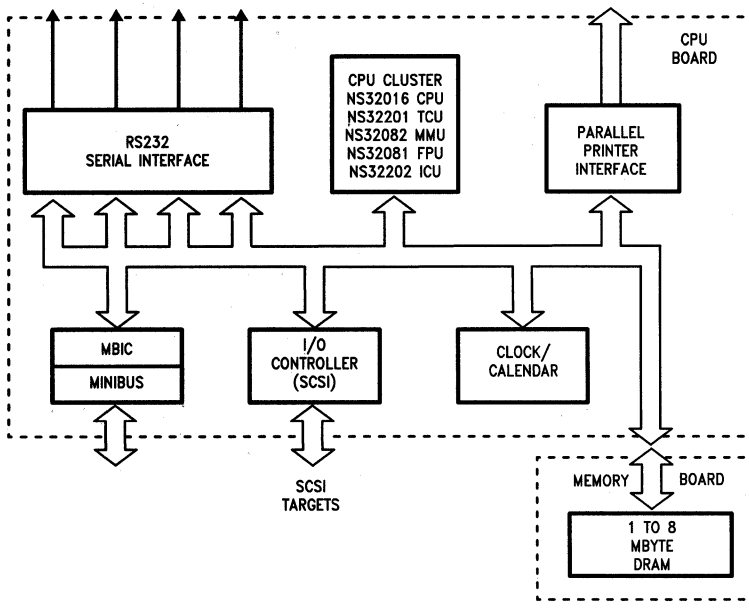
Ordering Information

ICM-3216	ICM-3216 CPU board with series 32000 CPU cluster (NS32016 CPU). Four RS232 serial ports, parallel port, SCSI interface and MiniBus.
ICM-3216-1MEM	ICM-3216 Memory board with 1 Mbyte of 150 ns DRAM.
ICM-3216-4MEM	ICM-3216 Memory board with 4 Mbytes of 150 ns DRAM.
ICM-CBL-TELCO	Cable, Telco, 6-wire, 7 ft.
ICM-CON-MAL	Adaptor, Telco to DB25 male connector
ICM-CON-FEM	Adaptor, Telco to DB25 female connector

Documentation

ICM-3216-M	ICM-3216 Hardware Reference Manual (420610289-001)
ICM-3216-SV-MS	UNIX V.2.2 Software Manual Set (970610289-001) ICM-3216 Administrator's Guide (424610287-001)
ICM-3216-MON-M	ICM-3216 Monitor User's Guide (424610289-001)
ICM-3216-RD-1	R&D pkg for ICM-3216. Pkg includes: Reconfigurable binary System V/Series 32000 ported to ICM-3216 (¼" tape cartridge); ICM-3216; ICM-3216-1MEM; complete set hardware and System V manuals. Requires software lic.
ICM-3216-RD-4	Same as ICM-3216-R/D-1 except ICM-3216-4MEM used rather than ICM-3216-1MEM.

ICM-3216 Block Diagram



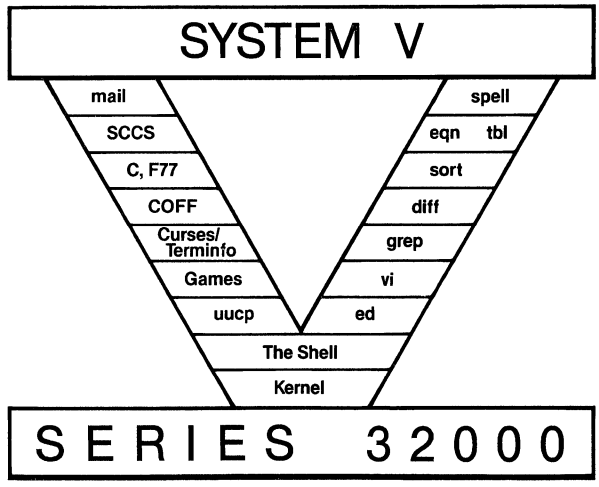
TL/T/8653–3

Section 9
Software Support

Section 9 Contents

System V/Series 32000 Operating System 9-3
Series 32000 GENIX 4.2 Operating System 9-5
Series 32000 GENIX Native and Cross-Support (GNX) Language Tools 9-7
Series 32000 EXEC ROMable Real-Time Multitasking EXECUTIVE 9-11
Series 32000 Real-Time Software Components VRTX/IOX/FMX/TRACER 9-15
Series 32000 VERDIX Ada Development System (VADS) for System V/Series 32000 Native Host 9-19
Series 32000 VERDIX Ada Development System (VADS) for VAX/VMS 9-21

System V/Series 32000® Operating System



TL/GG/8450–1

- **Certified version of AT&T's UNIX® System V, Release 2.0 Version 2**
- **Supports demand-paged virtual memory**
- **File and record locking**
- **Job Control**
- **Optimal use of Series 32000 architectural features**
- **Series 32000 C compiler**
- **Assembler, loader, run-time support library**
- **Improved Software Generation Systems (SGSs)**

General Description

System V/Series 32000 is a validated port of AT&T's UNIX System V, Release 2.0 Version 2 for the Series 32000 microprocessor family. In binary form, System V/Series 32000 is available as the operating system for the National Semiconductor Series 32000 Native Systems. It is also available in source form and can be modified to operate on Series 32000-based target systems.

System V/Series 32000 is a powerful, multitasking, multiuser operating system that is rapidly becoming a standard for 32-bit machines. Release 2.0 Version 2 of this system includes a significant improvement in performance over earlier System V releases, plus important functional enhancements.

System V/Series 32000 Features

- Demand-paged virtual memory. Each user program can access 8 Mbytes of virtual address space, enabling execution of programs with large memory requirements.
- File and Record locking, enabling effective data base management.
- Hierarchical file system with 1 Kbyte blocking for enhanced throughput in disk intensive applications.
- Command interpreter shell with job control facilities.
- Source code control system (SCCS) providing facilities to store, update and retrieve all versions of source code modules.

- UNIX to UNIX copy (uucp) inter-system communications program.
- High-level languages, including C and FORTRAN.
- A new Curses/Terminfo package, for terminal-independent application programs.
- Job Control over both foreground and background processes
- Enhanced electronic mail interface
- Each process runs in a protected linear address space of up to 16 Mbytes

Sample I/O Drivers

Console driver

Disk driver

Archive streaming tape driver

Multiport asynchronous RS232 communications driver

Centronics parallel printer driver

Drivers are in source form for OEM adaptation and require modification for specific end-equipment characteristics. Additional drivers may be added to support configuration features.

MINIMUM HARDWARE REQUIREMENTS

NS32016 or NS32032 32-bit Microprocessor (CPU)

NS32082 Memory Management Unit (MMU)

NS32201 Timing Control Unit (TCU)

NS32081 Floating-point Unit (FPU)

NS32202 Interrupt Control Unit (ICU)

512 Kbyte processor main storage (1 Mbyte recommended)

20 Mbyte disk storage

RS232 full-duplex asynchronous communications port: one minimum, two recommended

ADDITIONAL HARDWARE SUPPORTED

Additional RS232 full-duplex asynchronous communications ports

Streaming tape unit

Centronics-compatible printer

Processor main storage from 512 Kbytes to 16 Mbytes in size

Customer Support

National Semiconductor offers a full 90 day warranty period. Extended warranty provisions can be arranged by calling MCS Logistics at the toll-free numbers listed below.

The MCS Service Technical Support Engineering Center has highly trained technical specialists available to assist customers over the telephone with any product related technical problems.

For more information, please call:

(800) 538-1866,

(800) 672-1811 for California,

(800) 223-3248 for Canada.

Licensing

System V/Series 32000 is provided under license from National Semiconductor Corporation. Licensing provisions include binary distribution rights, source license fees and per-unit royalties. To obtain licensing information contact your National Semiconductor sales representative.

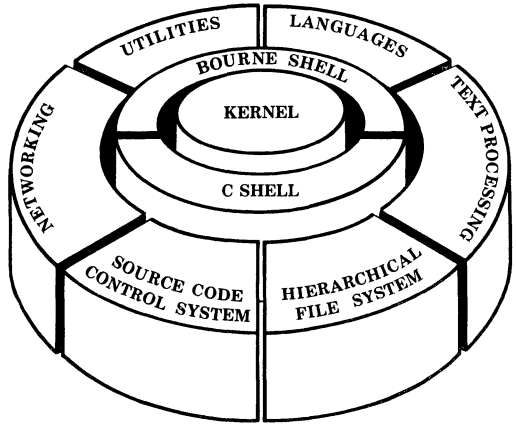
Ordering Information

NSW-SYSV-2VFR	System V/Series 32000 source on reel tape for VAX running UNIX System V as a host.
NSW-SYSV-BTFC	System V/Series 32000 on streaming cartridge tape for the ICM-3216 Board.

For future product releases contact your National Semiconductor Sales representative or call Series 32000 Software Marketing at (408) 721-5551.

National Semiconductor Corp.

Series 32000®
GENIX™ 4.2 Operating System



TL/GG/8526–1

- **Implementation of Berkeley 4.2 bsd UNIX™ for the Series 32000**
- **Demand-paged Virtual Memory tailored for National's Memory Management Unit (MMU — NS32082)**
- **Local Area Networking**
- **Dual, protected linear address space of up to 16 Megabytes**
- **Fast File System**
- **Series 32000 C Compiler**
- **Optional Series 32000 Pascal Compiler**

General Description

GENIX 4.2 is an AT&T-licensed implementation of the Berkeley 4.2 bsd UNIX operating system for the Series 32000 microprocessor family.

GENIX 4.2 is available in source form and can be user-modified to operate on Series 32000-based systems that include the Central Processing Unit (CPU), Memory Management Unit (MMU), Floating Point Unit (FPU), Timing Control Unit (TCU), and Interrupt Control Unit (ICU).

The GENIX 4.2 operating system provides a multitasking, multi-user environment specifically tailored for advanced software development. UNIX was originally designed to provide a productive working environment for the software engineer. To that end, the UNIX operating system provides a comprehensive set of utilities to make program development easier. The GENIX 4.2 operating system merges the powerful features of UNIX with the main-frame like architectural features of the Series 32000.

GENIX 4.2 Features

Demand-paged virtual memory code designed to take advantage of the NS32082 Memory Management Unit (MMU). The MMU provides two protected linear address spaces of up to 16 Mbytes each, which enables the execution of programs that have large memory requirements. National's virtual memory implementation features two levels of page tables and uses the MMU to maintain referenced and modified bits for page table entries. Software-defined bits in the page table entry automate the sharing of the text portion of any user process.

Berkeley 4.2 bsd local area networking for the Series 32000. The local area networking code implements the DARPA TCP/IP protocols, corresponding to a portion of the ISO OSI session layer (layer 3) and to all of the transport and network layers (layers 2 and 1, respectively). The system was designed to provide a framework within which new protocols and hardware can be easily supported.

Fast file system (FFS) for improved disk throughput. FFS algorithms allocate disk blocks in relation to cylinder cluster and in consideration of disk latency. Disk blocks are allocated in up to 8k byte blocks, with additional algorithms to handle the allocation of small-

er files as sub-units of the disk block to avoid wasting disk space.

C Compiler for the Series 32000 derived from the Berkeley Portable C Compiler (pcc). The C Compiler supports recent enhancements, such as the enumeration and void data types, and allows structures to be assigned and passed as arguments to a function.

Optional Pascal Compiler based on the Berkeley "pc" Pascal Compiler.

Standard UNIX utilities

- vi screen editor
- C shell with job control
- termcap and cursor control library for support of terminal independent software
- nroff and other documentation preparation utilities including the mm and ms macros
- source code control system
- uucp intersystem communications utility and associated programs

Additional enchancements

- ddt for assembly-level, kernel debugging
- dbg and idbg for development board and ISE™ debugging supplied in binary form

MINIMUM HARDWARE REQUIREMENTS FOR THE GENIX 4.2 OPERATING SYSTEM

NS32016 or NS32032 CPU

NS32082 MMU

NS32081 FPU

NS32201 TCU

NS32202 ICU

20 Megabytes of hard disk to accommodate GENIX 4.2 binaries.

1 Megabyte of physical memory

RS232 full-duplex asynchronous communications port

HOST ENVIRONMENT REQUIREMENTS:

- Approximately 100 Megabytes of disk space are required to install, modify, and recompile GENIX 4.2 Source in VAX™/Berkeley 4.2 cross mode.

DEVICE DRIVERS

GENIX 4.2 Source includes the following sample device drivers:

- Console driver
- Disk driver
- Streaming cartridge tape driver
- Multiport asynchronous RS232 terminal driver
- Parallel printer driver

Drivers require modification for specific end-equipment characteristics and may be supplemented by additional drivers to support end-product features.

SUPPORT

Support may be purchased under contract from National Semiconductor Corporation. Support includes error reporting, maintenance and feature updates, telephone assistance and quarterly bulletins.

LICENSING

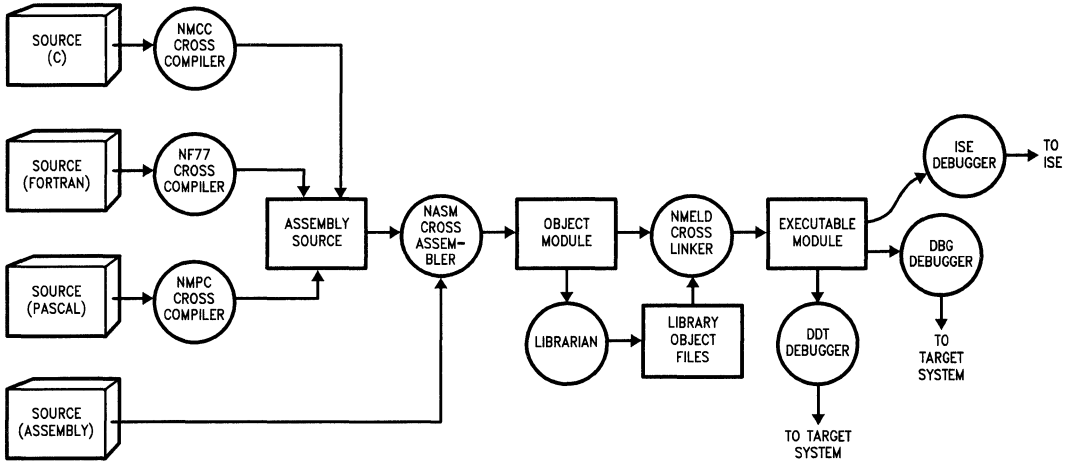
Source to the GENIX 4.2 operating system is provided under license from National Semiconductor Corporation. A valid AT&T UNIX System V source license is a prerequisite for licensing GENIX 4.2 Source. Licensing provisions include binary distribution rights, source license fees, and per-unit royalties.

ORDERING INFORMATION

NSW-GENIX-2VXR*	GENIX 4.2 Source on 9-track media; includes cross-support tools for compilation on a VAX running Berkeley 4.2 bsd.

*Software license agreement must be signed prior to order entry.

Series 32000® GENIX™ Native and Cross-Support (GNX™) Language Tools



TL/GG/8780–1

- **Implements AT&T's standard Common Object File Format (COFF)**
- **C, Pascal, and FORTRAN 77 Compilers**
- **Series 32000 assembler and linker**
- **In-System Emulator Support**
- **Interactive remote debugger with helpful command interface**
- **Available in binary for the VAX™ 4.2 bsd operating system under derivatives of the Berkeley operating system**
- **Available in binary for the VAX/VMS™ operating system**
- **Available in binary on National Semiconductor Series 32000 Native Systems**
- **Available in source for porting to other operating system environments**

Product Overview

The Series 32000 GNX Language Tools are a set of software development tools for the Series 32000 microprocessor family. Optional high-level language compilers work in conjunction with the standard components to provide tools that can be combined to meet a variety of development needs.

GENIX Native and Cross-Support (GNX) Language Tools

The Series 32000 GNX Language Tools are based on AT&T's Common Object File Format (COFF). With appropriate command-line arguments and when linked with appropriate libraries, code generated by the GNX language tools can be executed in either the GENIX/4 or GENIX/V environment. In addition, these tools can be used to develop operating-system-independent code or code designed to run in conjunction with real-time kernels, such as National's EXEC and VRTX®/Series 32000. All of National's new language tools conform to the GNX file format thereby ensuring that modules produced by any one set of tools can be linked with objects produced by any other set of GNX tools.

Standard Components

nasm — an assembler for GNX assembly language source code (produced either by a high-level language compiler or by an assembly language programmer) and produces an object file; supports NS32332 configuration register and 30-bit addressing;

nmeld — a linker that resolves references between object files and library routines and assigns relocated addresses to produce Series 32000 executable code;

nar — an archiver used to store frequently referenced objects in a library for convenient retrieval by the linker;

nlorder — finds ordering relation for an object library;

libm.a — a library that includes math routines that can be called from code written in assembly or high level languages. This library includes Bessel functions, exp, log, log10, pow, sqrt, floor, ceil, fmod, fabs, gamma, hypot, sinh, cosh, tanh, sin, cos, tan, asin, acos, atan, and atan2;

idbg16, idbg32 — debuggers for use with National's ISE16™ and ISE32™, respectively;

dbg16 — debuggers for downloading and debugging code on boards that use the NS32008, NS32016, NS32032, or NS32332 CPU;

mon16, mon32 — monitors for use with NS32016 and NS32032 provided in PROM and in assembly language source so the user can modify and install the monitor on user-designed Series 32000 hardware.

nburn — a PROM—programming utility that works in conjunction with a DATA I/O Model 19™ to program PROMs for use with the NS32008, NS32016, NS32032 and NS32332 CPUs. At the user's option, nburn imbeds escape sequences that activate the auxiliary port of the VT102 terminal, to which a DATA I/O Model 19 is connected, for one-step PROM-programming.

db library — development board support routines, such as string, scanf, printf, atof, abs, regex, getc, putc, and puts;

cvtasm — utility to assist in converting previous assembler syntaxes to GNX assembler syntax;

nsize — a utility for displaying the size of the text, uninitialized data, and initialized data segments of an object file;

nstrip — a utility to remove symbol table information from an object file;

nnm — a utility to display the symbol table of an object file.

The following two programs are available for configurations designed for operation on a VAX under derivatives of the Berkeley 4.2 bsd operating system.

ddt — a debugger specifically designed for kernel debugging;

dbmon — a monitor for use with ddt provided on PROM and in assembly language source so the user can modify and install the monitor on user-built Series 32000 hardware.

Optional Components

C Compiler

The C compiler is derived from the UNIX® C Compiler. The C compiler is augmented by an optional optimization pass and generates Series 32000 assembly code. The C compiler supports the C language as defined by Kernighan and Ritchie, plus recent enhancements, such as passing structures as arguments to functions and long variable names.

C object modules can be linked with assembly, Pascal, and FORTAN 77 object modules for mixed-language development.

A library of terminal I/O functions is also provided. These functions can be called by user-developed code to allow a program running on a development board to print data to and accept data from the console terminal. Source to these routines is provided, should the user elect to expand or modify the functionality of these routines. In addition, functions from the C library, "libc.a", that do not rely on the kernel for execution, are included.

Pascal Compiler

The Pascal compiler is an ISO-standard, optimized Pascal compiler derived from the 4.2 bsd "pc" compiler.

The Pascal compiler supports several extensions to the standard Pascal language that are designed to simplify program development, such as separate compilation of individual modules. In addition, I/O of enumerated types, output of octal and hexadecimal numbers, and comparison of strings of unequal length are supported. The Pascal library, "libpc.a", includes several useful procedures and functions, for example, a random number generator, file manipulation procedures, and clock functions.

Pascal object modules can be linked with assembly, C, and FORTRAN 77 object modules for mixed-language development. Pascal programs can call the terminal I/O functions described for the C Compiler.

FORTRAN 77 Compiler

The FORTRAN 77 Compiler is derived from the UNIX System V Release 2 "f77".

The compiler supports several enhancements that make "f77" more flexible. In particular, a command line option allows the user to increase the size of compiler tables for equivalences, external symbols, statement numbers, loops or if-then-elses, names, labels for computed gotos and the number of alternate returns.

FORTRAN object modules can be linked with assembly, C, and Pascal object modules for mixed-language development. FORTRAN programs can call the terminal I/O functions described for the C Compiler above.

Source Products

The assembler, associated tools, and the optional C, Pascal, and FORTRAN 77 Compilers are provided in binary form for use on a VAX under the 4.2 bsd operating system. The source to all programs that make up the Series 32000 GNX Language Tools is available for porting to other UNIX operating system environments.

Customer Support

National Semiconductor offers a full 90 day warranty period. Extended warranty provisions can be arranged by calling MCS Logistics at the toll-free numbers listed below.

The MCS Service Technical Support Engineering Center has highly trained technical specialists available to assist customers over the telephone with any product related technical problems.

for more information, please call:

(800) 538-1866,

(800) 672-1811 for California,

(800) 223-3248 for Canada.

Licensing

All binary versions of the Series 32000 GNX Language Tools require the execution of National's Binary User Agreement. Because the language tools include AT&T proprietary code, a System V source license is a prerequisite for obtaining source versions of these language tools.

Part Numbers

Binaries for Cross-Support Mode hosted on VAX under 4.2 bsd:

NSW-ASM-BRVX	The assembler ("nasm"), linker ("nmeld"), math library ("libm.a"), archiver ("nar"), ISE debugger ("idbg16/32") development-board-support library, development board debuggers ("ddt" and "dbg16"), monitors in source ("dbmon" and "mon16/32"), monitors in PROM, the PROM-burning utility ("nburn"), dblibrary, cvtasm, nsize, nstrip, and nnm.
NSW-C-BRVX	Optional C compiler to be used in conjunction with NSW-ASM-BRVX described above.
NSW-F77-BRVX	Optional FORTRAN 77 Compiler to be used in conjunction with NSW-ASM-BRVX described above.
NSW-PAS-BRVX	Optional Pascal Compiler to be used in conjunction with NSW-ASM-BRVX described above.

All binaries for VAX/4.2 bsd are delivered on 9-track reel tape in "tar" format.

Binaries for Cross-Support Mode hosted on VAX under VMS:

NSW-ASM-BRVM	The assembler ("nasm"), linker ("nmeld"), math library ("libm.a"), archiver ("nar"), ISE debugger ("idbg16/32") development-board-support library, development board debugger ("dbg16"), monitor in source ("mon16/32"), monitor in PROM, the PROM-burning utility ("nburn"), dblibrary, cvtasm, nsize, nstrip, and nnm.
NSW-C-BRVM	Optional C Compiler to be used in conjunction with NSW-ASM-BRVM described above.
NSW-F77-BRVM	Optional FORTRAN 77 Compiler to be used in conjunction with NSW-ASM-BRVM described above.
NSW-PAS-BRVM	Optional Pascal Compiler to be used in conjunction with NSW-ASM-BRVM described above.

All binaries for VAX/VMS are delivered on 9-track reel tape in VMS BACKUP format.

Source

NSW-ASM-SRNN	The source to NSW-ASM-BRVX, as described above.
NSW-C-SRNN	The source to NSW-C-BRVX, as described above.
NSW-PAS-SRNN	The source to NSW-PAS-BRVX, as described above.
NSW-F77-SRNN	The source to NSW-F77-BRVX, as described above.

All source tapes are delivered on 9-track reel tape written in "tar" format.

For future product releases contact your National Semiconductor sales representative or call Series 32000 Software Marketing at (408) 721-5551.

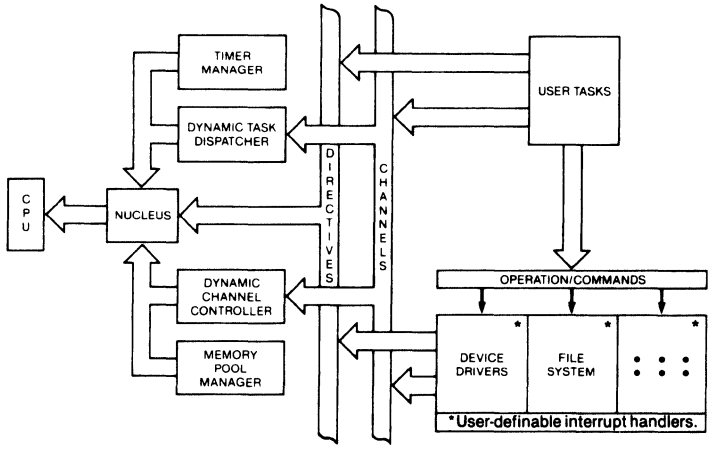
Manuals

Each software package is delivered with one copy of each appropriate manual.

NSP-ASM-M-MS:	Manual Set included with NSW-ASM-BRVM
NSP-ASM-X-MS:	Manual Set included with NSW-ASM-BRVX
NSP-C-MV:	Manual included with NSW-C-BRVM and NSW-C-BRVX
NSP-PASCAL-M:	Manual included with NSW-PAS-CAL-BRVM and NSW-C-BRVX
NSP-F77-M:	Manual included with NSW-F77-BRVM and NSW-F77-BRVX

National Semiconductor Corp.

Series 32000® EXEC ROMable Real-Time Multitasking EXECUTIVE



TL/GG/7291–1

- **Provides a multitasking executive for real-time applications**
- **Supports all Series 32000 CPUs**
- **Complete Source Code Package**
  - — Fully user configurable
  - — Hardware independent
- **Extensive user implementation support**
  - — Unique demo, program introduction
  - — C and Pascal interface libraries
  - — Sample terminal drivers
  - — Integrated with Series 32000 development boards and monitor
- **ROMable**
- **Reconfigurable**
- **Real-time clock support for time-of-day and event scheduling**
- **Allows up to 256 levels of task priority which can be dynamically assigned**
- **Up to 256 logical channels for task communication**
- **Free-memory pool control**
- **Available for VAX™/VMS™, VAX/UNIX®, SYS32™ and VR32™ development environments**

Product Overview

EXEC is National Semiconductor's real-time, multitasking executive for Series 32000 based applications. Its primary purpose is to simplify the task of designing application software and provides a base upon which users can build a wide range of application systems. EXEC requires only 2K bytes of RAM and only 4K bytes of ROM and is fully compatible with National Semiconductor's Series 32000 family and the Series 32000 development board family.

EXEC allows the user to monitor and control multiple external events that occur asynchronously in real-time, such as intertask communications, system resource access based upon task priority, real-time clock control, and interrupt handling. These functions greatly simplify application development in such areas as instrumentation and control, test and measurement, and data communications. In these applications, EXEC provides an environment in which systems programmers can immediately implement software for their particular application without regard to the details of the system interaction.

9

EXEC executive is fully modular and can be readily configured to suit application needs. It is both hardware and location independent, thus providing a fundamental base on which to build a wide range of applications systems. In addition, it provides a buslike structure that helps to integrate software with the underlying hardware through predefined data structures and interconnect procedures. This architecture ensures maximum standardization for both compatibility and future expansion.

Features

Structured Environment—The EXEC executive and its associated modules support and encourage modular, structured programming, thus providing a consistent structure from application to application, which allows experience gained and software written on one system to be easily transferred to another. Frequently, entire programs may be used in multiple applications, *even if different CPU boards are involved.*

Hardware-Oriented Interface—The EXEC executive provides an intertask and task/executive communications architecture that is similar to hardware communications. Instead of an array of "mailboxes" (or "message centers"), EXEC uses channels. This interface is consistent throughout the range of facilities offered, thus reducing the number of concepts to be learned, providing greater control at the task level, and increasing the efficiency of the system and the programming effort.

Wide Choice of CPUs—EXEC is compatible with the full line of 32-bit Series 32000 CPUs offered. These include the NS32008, NS32032, and the NS32C016. Users will be able to move a NS32016 system:

- to an NS32008 for cost-effectiveness,
- to an NS32032 for increased computing power, or
- to an NS32C016 for low-power applications.

Time-Of-Day Clock—The EXEC executive has an integral system/time-of-day clock. Included is a real-time clock configurable to a resolution of 1ms. This eliminates the need to allocate the extra memory otherwise required for this feature.

Small Nucleus—The EXEC nucleus was hand-coded in assembly language rather than being compiled from intermediate or high-level languages. The resulting product is therefore smaller and allows the incorporation of more features within an optimum size.

Priority-Oriented Scheduler—The EXEC scheduler ensures that the highest priority task that is ready to execute is given control, so the system is responsive to its external world. Dynamic reprioritization of tasks is supported for the most sophisticated of multitasking systems.

Real-Time Speed—Because EXEC was hand-coded in assembly language, several advantages with regard to speed are gained. Task swapping, channel and message management, and I/O interfacing are executed more quickly than could be expected of a system written in a higher level language.

Direct Interrupt Processing—The EXEC architecture employs interrupt channels which allow device-specific interrupt handling routines to interface directly with the interrupt source. This accomplishes servicing of interrupts without the overhead of task swapping, yet allows the operating system to maintain the integrity of the system. Combining this interrupt service architecture with a device-efficient nucleus results in an operating system that better supports demanding, real-time applications.

User Configurability—EXEC executive-based applications may be configured from a wide range of facilities, selecting only those that meet the specific requirements of the application system. The resultant system contains only the modules necessary for its use, allowing the EXEC executive to fit a wide range of applications from small, special-purpose, dedicated applications to large, general-purpose systems.

Event Driven—In the EXEC executive, each user task exists in its own "closed environment"—a virtual processor. Each virtual processor can synchronize with external/internal occurrences through events. EXEC supports a wide variety of events, including synchronization with task activities, external device operations, and the real-time clock.

Memory Pool Manager—The EXEC executive has an integral memory pool manager. This feature not only reduces the amount of RAM required in an application system (potentially reduces board count), but also allows active modules more buffer area within any given space constraint.

Internal Structure

EXEC may be viewed as composed of a set of functions. These functions are:

1. Nucleus— performs task and channel management and controls executing memory.
2. Timer Manager—performs time-dependent control.
3. Dynamic Task Dispatcher—performs dynamic creation and installation of tasks at run-time.
4. Dynamic Channel Controller—performs dynamic creation and installation of software and interrupt channels at run-time.
5. Memory Pool Manager—performs memory allocation and deallocation.

The Timer Manager, Dynamic Task Dispatcher, Dynamic Channel Controller, and the Memory Pool Manager all operate under direction of the Nucleus, which assigns tasks to run on the hardware CPU.

System Functions

EXEC controls CPU allocation by resolving conflicting needs of individual tasks, and monitors external events. The Event Manager, Task Manager, Channel Manager, Memory Manager, and Timer Manager provide system facilities that are directly accessible from the user task level. A representative sampling of system functions are summarized below:

- **Task and Event Management**

1. TSKBD —Build a task and schedule it to run.
2. SUSPD —Suspend a task.
3. GTPRI —Get task priority.
4. STPRI —Change run-time task priority.
5. WAITE —Wait for an event or combination of event to occur before resuming task processing.
6. TSTEV —Test the current state of an event.

- **Intertask Communication**

1. RECV(W) —Receive data from a channel and, optionally, wait for an event to occur.
2. SEND(W) —Send a message to a channel and, optionally, wait for an event to occur.
3. SIGNL —Synchronize with another task through event flags; signal completion.
4. BLDSC —Build software channel.

- **Interrupt Handling**

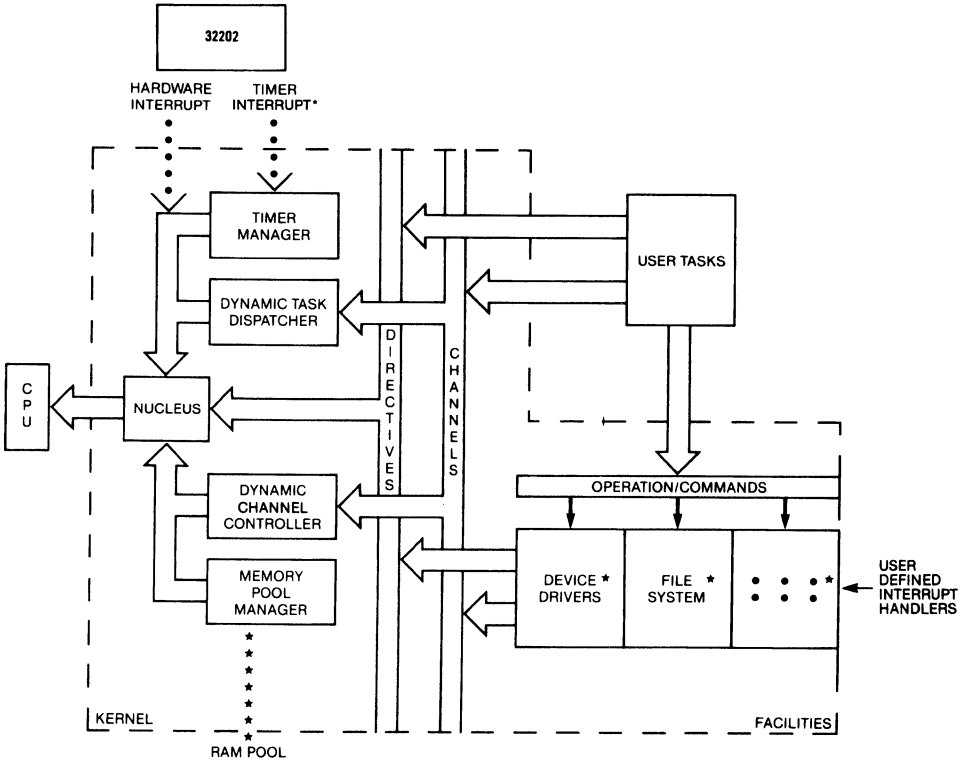
1. INTEX —Interrupt exit from executive.
2. BLDIC —Build an interrupt channel.

- **Memory Pool Management**

1. ALLOC —Allocate a block of pooled memory.
2. DALOC —Deallocate memory back to pool.

- **Timer Management**

1. MRKT(W) —Mark a time delay and, optionally, wait for an event to occur.
2. CMRKT —Cancel previously posted mark-time event.
3. GTIMD —Get current time of day.
4. STIMD —Set current time of day.



TL/GG/7291-2

* May or may not be from 32202 ICU.

FIGURE 1. EXEC Structure

Ordering Information

VAX/VMS Environment

Order Number: NSW-EXEC-SRVM*

Shipping Configuration: Software on 1600 bpi magnetic tape (9-track VMS copy format). EXEC reference manual.

Prerequisite: NSW-ASM-BRVM cross software package, at current revision level.

VAX/UNIX Environment

Order Number: NSW-EXEC-SRVX*

Shipping Configuration: Software on 1600 bpi magnetic tape (UNIX tar tape format). EXEC reference manual.

Prerequisite: NSW-ASM-BRVX* cross software package, at current revision level.

SYS32 Environment

Order Number: NSW-EXEC-SCSG

Shipping Configuration: Software on SYS32 format streamer tape cartridge. EXEC reference manual.

Prerequisite: SYS32 Development System with current revision level software.

VR32 Environment

Order Number: NSW-EXEC-SDQF

Shipping Configuration: Software on VR32 Format diskettes (floppy disks). EXEC-COF reference manual

Prerequisite: VR32 TARGET/DEVELOPMENT System with current revision level software.

For future product releases contact your National Semiconductor sales representative or call Series 32000 Software Marketing at (408) 721-5551.

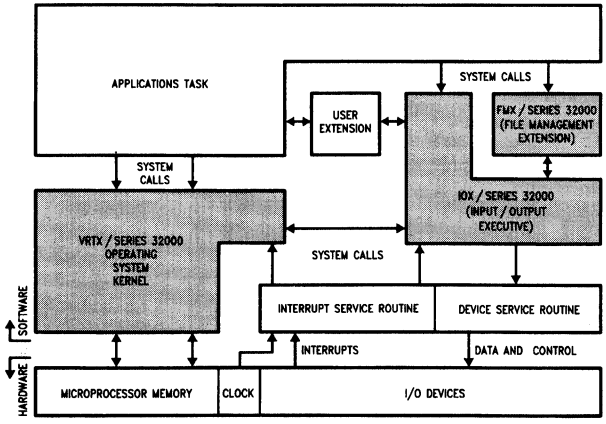
Documentation

EXEC ROMable Real-Time Multitasking EXECUTIVE Reference Manual. Included with software package. May also be ordered separately.

Order Number: NSP-EXEC-M (same manual for all configurations except NSW-EXEC-SDQF which requires NSP-EXEC-COF-M).

*Software license agreement must be signed prior to order entry.

Series 32000® Real-Time Software Components VRTX®/IOX®/FMX®/TRACER™ Series 32000



TL/GG/8781–1

- **Components are plug-in building blocks (PROMs)**
- **Components can be located anywhere in memory**
- **No linking of components with application code**
- **Component functions can be customized by user (hooks)**
- **Comprehensive manuals with many examples**
- **Off-the-shelf real-time multitasking executive**
- **I/O Manager supports sequential and random access I/O**
- **File Manager supports PC-DOS file system and directories**
- **Interactive PROM-resident multitasking debugger**
- **Interface libraries for applications written in C**
- **C Run-time Support Library**

Product Overview

Real-time software components form a family of building blocks that speed the development of Series 32000-based embedded systems. The central member of the family is the VRTX/Series 32000 kernel which can be used alone or in combination with the other components to build a more complete operating system. The IOX/Series 32000 and FMX/Series 32000 components support a file system that is media compatible with PC-DOS. The TRACER/Series 32000 multitasking debugger can be used in VRTX/Series 32000-based systems during development for debug, download and test.

All the components reside in PROM's installed in the target system. They can be placed anywhere in the address space and make minimal assumptions about the hardware environment. Small user-written routines supply the information about the local implementation of interrupts, timers, I/O devices, etc. Application tasks interface to the components with Series 32000 SVC (Supervisor Call) interrupts, thus code for the components does not require linking with user-written code. There is no dependence on the conventions of any Series 32000 linker, object format or compiler.

VRTX/Series 32000

VRTX/Series 32000 is a real-time multitasking executive that supports the Series 32000 Microprocessor Family in embedded systems. The executive resides in PROM in the target system, manages the multitasking environment and responds to requests for services from application tasks.

Task Management

The basic logical unit controlled by VRTX/Series 32000 is the task which is a logically complete path through user code that requires system resources. Each task has a priority level used by VRTX to determine how access to the CPU is allocated. Up to 256 priority levels are available. VRTX allocates the CPU sequentially to the highest priority task that is ready to execute. Tasks can create, delete, suspend, and modify the priority of themselves and other tasks. Task delays and time-slicing are also available.

Intertask Communication and Synchronization

Tasks can communicate and synchronize with other tasks via exchange of pointer-length messages through mailboxes. These permit mutual exclusion and resource-locking. VRTX also has directives for dynamically building and managing message queues.

Interrupt Services

VRTX has directives for use by user-written interrupt handlers that provide the interface between tasks and devices. They permit the interrupt handler to influence the scheduling of critically important tasks.

Memory Management

VRTX provides directives for managing the free memory pool. To minimize fragmentation and overhead, storage is allocated and released as fixed size blocks from within memory partitions. Partitions can be built dynamically. There are no constraints on block size.

Special Device Support

Since many applications require a real-time clock and a character I/O device, support for them is integrated into VRTX/Series 32000. Designers need only supply a small hardware-dependent interrupt service routine for each. VRTX will then manage all the logical operations to supply the clock management and character I/O services to application tasks and interrupt handlers.

Extensions

VRTX/Series 32000 accommodates applications with special requirements by supplying three hooks at key points in its execution. They permit the designer to modify VRTX processing without having to modify VRTX itself. Whenever VRTX reaches a hook it checks for the presence of an application routine. VRTX hooks are called at task create, delete, and context switch. There are no constraints on hook use. They can be used for saving/restoring the floating-point environment or for maintaining a counter in the task control block to monitor task execution.

VRTX System Calls

Type	Call	Description
Initialization	VRTX_INIT	Initialize VRTX
	VRTX_GO	Start multitasking
Task Management	SC_TCREATE	Create a task
	SC_TDELETE	Delete a task
	SC_TSUSPEND	Suspend a task
	SC_TRESUME	Resume execution of suspended task
	SC_TPRIORITY	Change task priority
	SC_LOCK	Disable task rescheduling
	SC_UNLOCK	Enable task rescheduling
	SC_TSLICE	Enable time slicing
Communications Services	SC_POST	Post message to mailbox
	SC_PEND	Pend for message at mailbox
	SC_ACCEPT	Accept message at mailbox
	SC_QCREATE	Create message queue
	SC_QPOST	Post message to queue
	SC_QPEND	Pend for message from queue
	SC_QACCEPT	Accept message from queue
	SC_QINQUIRY	Get queue status
Memory Management	SC_GBLOCK	Get memory block
	SC_RBLOCK	Release memory block
	SC_PCREATE	Create memory partition
	SC_PEXTEND	Extend memory partition
Timer Services	SC_GTIME	Get system time
	SC_STIME	Set system time
	SC_TDELAY	Suspend task temporarily
Character I/O	SC_GETC	Get a character
	SC_PUTC	Put a character
	SC_WAITC	Wait for special character
Interrupt Services	UI_ENTER	Enter interrupt handler
	UI_TIMER	Post time increment from interrupt
	UI_RXCHR	Post received character from interrupt
	UI_TXRDY	Post transmit ready from interrupt
	UI_EXIT	Exit from interrupt handler

IOX/Series 32000

IOX/Series 32000 is an input/output executive that provides I/O services in VRTX/Series 32000-based systems. It presents a uniform I/O interface to application tasks and supports overlapped I/O as well as device and channel sharing. IOX supports sequential and random access I/O with both direct and buffered read/write. IOX also provides for extending its functionality by using request service routines (hooks), special I/O handlers and facilities for attaching File Management Executives (FMX's). I/O devices are connected to IOX by means of low-level Device Service Routines and Interrupt Service Routines.

IOX System Calls		
Type	**Call**	**Description**
Initialization	IO_INIT	Initialize IOX data structures
Device Definition	IO_DFCHR	Define a character device
	IO_DFBLK	Define a block device
	IO_DFDSK	Define a disk device
	IO_RMDEV	Remove device definition
Buffered I/O	IO_OPEN	Connect a channel to a device
	IO_CLOSE	Disconnect channel from a device
	IO_GET	Read bytes from buffered channel
	IO_PUT	Write bytes to buffered channel
Direct I/O	IO_OPEN	Connect a channel to a device
	IO_CLOSE	Disconnect channel from a device
	IO_READ	Read a block from direct channel
	IO_WRITE	Write a block to direct channel
	IO_CNTRL	Perform device control
	IO_WAIT	Wait for outstanding I/O requests
	IO_RESET	Reset I/O channel
Device Service	IO_POST	Post I/O request completion
	IO_STMR	Start request timer
	IO_CTMR	Cancel request timer
	IO_TIMER	Announce I/O timer interrupt
	IO_RXCHR	Put character into receiver buffer
	IO_RXCHM	Put characters into receiver buffer
	IO_ECHO	Put character into echo buffer
	IO_ECHOM	Put characters into echo buffer
	IO_TXRDY	Get character from transmitter buffer
	IO_TXRDM	Get characters from transmitter buffer
	IO_EXCPT	Call exception routine
Extension	IO_ATCHC	Connect an I/O handler

FMX-DOS/Series 32000

FMX-DOS/Series 32000 is a File Management Executive that adds multitasking PC-DOS file management to VRTX/Series 32000 and IOX/Series 32000-based systems.

FMX-DOS permits tasks to create and delete files, assign file attributes and open IOX channels to them for read and write operations. FMX-DOS supports both buffered and unbuffered read/write as well as sequential and random access methods. Directory operations to create and delete directories, and to reference files relative to a directory are provided. Volume operations perform volume formatting, mounting and dismounting.

FMX-DOS can be initialized to create PC-DOS directory and file structures on disks. Users can also customize FMX-DOS by installing their own software at hooks provided at key points in FMX-DOS processing.

FMX-DOS makes no assumption about the hardware environment; it uses IOX for all device-oriented operations.

FMX-DOS System Calls		
Type	**Call**	**Description**
Initialization	FD_INIT	Initialize FMX data structures
Volume Control	FD_FMAT	Format a volume
	FD_MOUNT	Mount a volume
	FD_DISMT	Dismount a volume
	FD_QVOL	Query volume attributes
	FD_EVOL	Evaluate volume parameters
	FD_SYNC	Synchronize a volume
Directory Operations	FM_MKDIR	Create a directory
	FM_RMDIR	Delete a directory
File Operations	FM_OPEN	Gain access to a file
	FM_CREAT	Create a file
	FM_DELET	Delete a file
	FM_GATTR	Get file attributes
	FM_SATTR	Set file attributes
	FM_RENAM	Rename a file

TRACER/Series 32000

TRACER/Series 32000 is an interactive multitasking debugger that lets the designer monitor and control VRTX/Series 32000-based systems without distorting the interaction between tasks and their environment. It runs parallel to the multitasking environment rather than as a task. TRACER/Series 32000 can access all user-supplied code, including interrupt handlers, user-supplied system call handlers and VRTX/Series 32000 extensions. System data structures, task and system state parameters are shown in easily understood form relieving the user from interpreting unformatted memory displays.

TRACER/Series 32000 has commands to:

— set, display and remove breakpoints
— display and modify registers and memory
— display task and system state parameters
— do single-step execution
— disassemble binary code
— download program files from a host system

Customer Support

National Semiconductor offers a full 90 day warranty period. Extended warranty provisions can be arranged by calling MCS Logistics at the toll-free numbers listed below.

The MCS Service Technical Support Engineering Center has highly trained technical specialists available to assist customers over the telephone with any product related technical problems.

for more information, please call:

(800) 538-1866 outside of California
(800) 672-1811 for California
(800) 223-3248 for Canada

Licensing

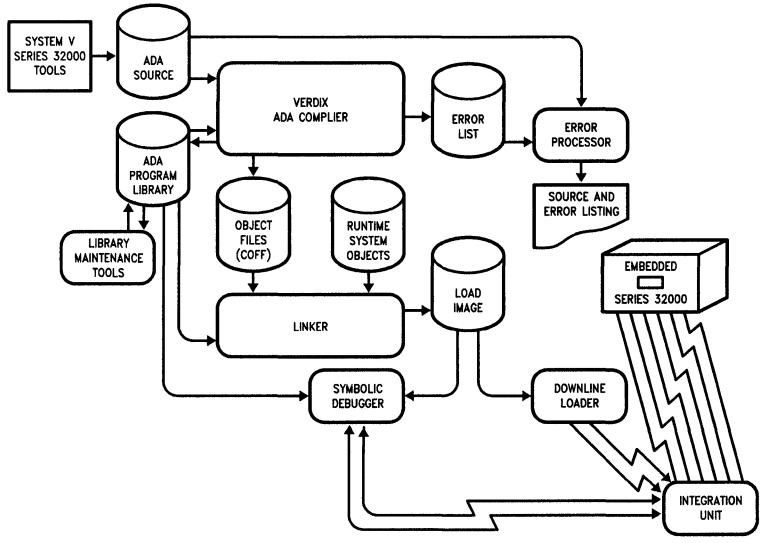
National Semiconductor licenses the user to manufacture binary copies of the software component products. The user must purchase from National Semiconductor the license rights to the number of copies needed. To obtain licensing information contact your National Semiconductor sales representative or call Series 32000 Software Marketing at (408) 721-5551.

Ordering Information

National Semiconductor will provide R&D Packages for VRTX/Series 32000, IOX/Series 32000, FMX/Series 32000 and Tracer/Series 32000 which include one master copy of the PROM's, documentation and a license to make copies for R&D purposes. To obtain more information contact your National Semiconductor sales representative or call Series 32000 Software Marketing at (408) 721-5551.

Series 32000® VERDIX™ Ada® Development System (VADS™) for System V/Series 32000 Native Host

VERDIX Ada for the Development Environment



TL/GG/8775–1

- **Validated**
- **Series 32000 native development environment for System V/Series 32000 native host**
- **Derived from Verdix's current Ada Technology (VADS)**
- **Generates System V Common Object File Format (COFF)**
- **Ada Compiler with optimization**
- **In-System emulator support**
- **Fully symbolic interactive down-line debugger**
- **Supports either Verdix's stand alone runtime system (ADEX) or can utilize VRTX/ Series 32000**

Product Overview

This Series 32000 Ada development environment supports Ada program development on a Series 32000 native host, thus efficiently generating executable code. The compiled Ada program can run on this host under the System V/Series 32000 host operating system or can be targeted to run on other Series 32000 based environments. This is a true "production quality" Ada compiler focused on high performance and intended for large scale development of both applications and systems software. This native development environment includes the compiler, linker, debugger, program library utilities, and embedded computer system runtime system all using advanced technology to provide user-friendliness and efficiency.

Series 32000 development environment has been designed and engineered to interface with the existing System V operating system and tools so that a programmer can concentrate on learning and programming Ada, rather than a new operating system. This Ada compiler operates as a re-entrant, shareable process in the System V/Series 32000 host environment and makes full use of most operating facilities. On-line help facility for VADS commands using UNIX MAN command as well as a help facility in the debugger are also provided.

Components

Ada Compiler

This compiler accepts Ada source and generates Series 32000 code which can be debugged and executed on the System V/Series 32000 host or downloaded to another Series 32000 based target environment.

The full Ada language is supported. Some of the supported features are shared and unshared generics, separates, in-lines, bit representation, machine-code insertion, interrupt tasks, monitor tasks, terminal and file I/O through/to the host for debugging. COFF compatible object files which are linkable with other languages are generated. Optimization includes value tracking global register allocation, register assignment for commons and locals, common sub-expression, branch and dead code analysis, some constraint check elimination and local and peephole optimizations.

Debugger

This Ada Debugger provides a fully symbolic debugging facility. The user can access Ada variables by name and the Ada Debugger utilizes its understanding of Ada types to display formatted values. This Ada Debugger has sophisticated break point capabilities and single step execution is also provided. This Ada Debugger provides access to the Ada source code during debugging. This facility is integrated with the breakpoint and stepping capabilities so the user has a source "window" into the program at all times. This Debugger was designed to be simple, convenient, and fast.

The user interface to the Debugger accommodates down-line as well as host system debugging. The debugger uses COFF compatible symbolic information so it can debug other languages as well, for example, C. In addition, an on-line help facility is provided while in the debugger.

Program Library Utilities

The Ada language imposes stringent requirements on an Ada Program Library. While the language provides for separate compilation of program units, each unit is compiled in the "context" of previously compiled units. The compiler must have access to this context, carefully organized in the form of a Program Library. This library has been designed to enhance the compiler performance. A set of utilities is provided to manage, manipulate, and display Program Library information.

In addition, the Ada Development System permits Ada Program Libraries to be hierarchically organized, so that units not local to one library may be found in other libraries. Thus, programmers can work without interference on local versions of individual program units, while retrieving the remainder of the program from higher-level libraries.

Ada Run Time System

The Series 32000 Ada Run Time System provides comprehensive support for tasking, debugging, exception handling, and input/output. Adex, Verdix's routine system, can be used in stand alone mode or VRTX/Series 32000 can be utilized.

The Run Time System is linked with the generated users Ada program and to facilitate resource utilization efficiency, major portions of the Run Time System have been optimized. The Run Time System also interfaces with the symbolic debugger to provide comprehensive program checkout facilities. Run Time sources for customization are also available.

Supported Hardware

— Native Series 32000 Host

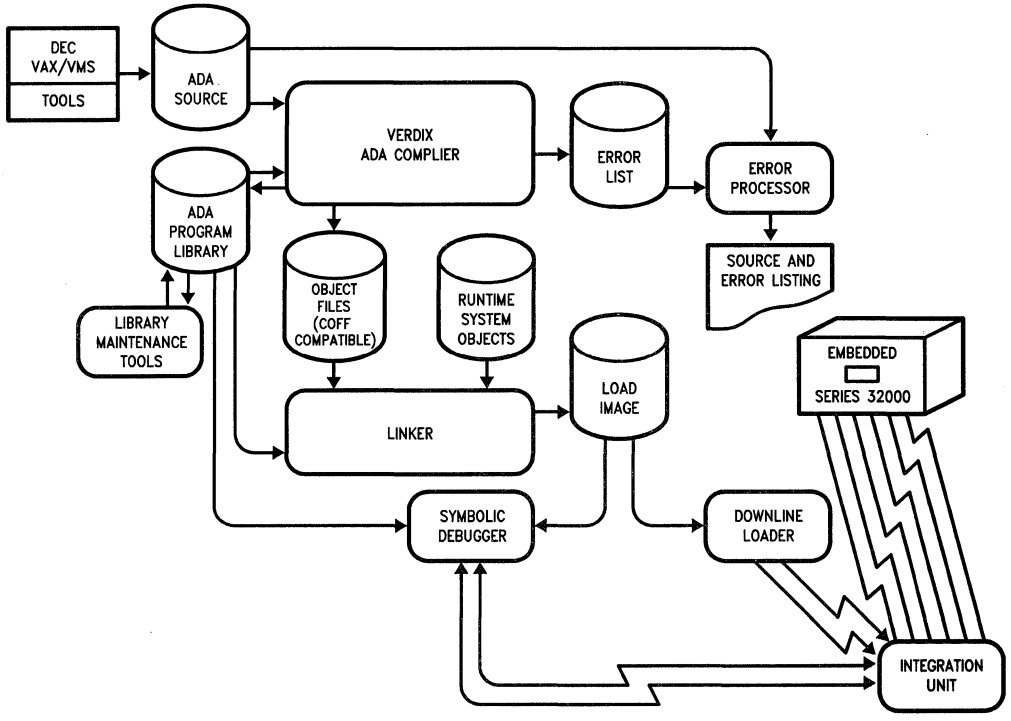
— System V 5.2 or later/Series 32000 Operating System

Shipping Package

— User and reference documentation for Series 32000 VADS

Series 32000® VERDIX™ Ada® Development System (VADS)™ for VAX™/VMS™

VERDIX Ada for the DEC VAX/VMS → Series 32000 Cross Compiler



TL/GG/8774–1

- **Validated**
- **Series 32000 Cross-Software Development Environment for VAX/VMS**
- **Derived from VERDIX's Current Ada Technology (VADS)**
- **Generates System V Common Object File Compatible Format**
- **Ada Compiler with Optimization**
- **In-System Emulator Support**
- **Fully Symbolic Interactive Down-Lline Debugger**
- **Supports Either Verdix's Stand Alone Run-Time System (ADEX) or can Utilize VRTX/Series 32000**

Product Overview

This Series 32000 Ada cross-development environment supports Ada program development on a VAX/VMS host generating executable Series 32000 target code. This is a true "production quality" Ada compiler focused on high performance and intended for large scale development of both application and systems software. This development environment includes the compiler linker, debugger, program library utilities, and embedded computer run-time system all using advanced technology to provide user-friendliness and efficiency. This development environment has been designed and engineered to interface with the existing VAX/VMS operating system and tools so that a programmer can concentrate on learning and programming Ada, rather than a new operating system. This

compiler operates as a re-entrant, shareable process in the VAX host environment and makes full use of host operating facilities. On-line help facilities for VADs commands using VMS help commands as well as a helpful facility in the debugger are also provided.

Components

Ada Compiler

This compiler, hosted on the VAX/VMS, accepts Ada source and generates Series 32000 assembly langauge code which can be linked, downloaded, assembled, and debugged for native execution on the Series 32000. The full Ada language is supported. For example, shared and unshared generics, separates, in-lines, bit representation, machine-code insertion, interrupt tasks, monitor tasks, terminal and file I/O through/to the host for debugging. COFF compatible object files which are linkable with other languages are generated. Optimization includes value tracking, global register allocation, register assignment for parameters and locals, common sub-expression, branch and dead code analysis, some constraint check elimination and local and peephole optimizations.

Debugger

The Ada Debugger provides a fully symbolic debugging facility. The user can access Ada variables by name and the Ada Debugger utilizes its understanding of Ada types to display formatted values. This Ada Debugger has sophisticated breakpoint capabilities and single step execution is also provided.

The Ada Debugger provides access to the Ada source code during debugging. This facility is integrated with the breakpoint and stepping capabilities so the user has a source "window" into the program at all times. This Debugger was designed to be simple, convenient, and fast. The Debugger uses COFF compatible symbolic information so it can debug other languages as well. For example, C.

The user interface to the Debugger accommodates down-line as well as host system debugging. That is, the Debugger can be used in the mode where a complied Ada program is downloaded to a Series 32000 target system but debugged from the host system. The ISE™ can be connected to the host as a terminal reaching the target by probe. An on-line help facility is also provided and can be used while in the Debugger.

Program Library Utilities

The Ada language imposes stringent requirements on an Ada Program Library. While the language provides for separate compilation of program units, each unit is compiled in the "context" of previously compiled units. The Compiler must have access to this context, carefully organized in the form of a Program Library. This library has been designed to enhance the compiler performance. A set of utilities is provided to manage, manipulate, and display Program Library information.

In addition, the Ada Development System permits Ada Program Libraries to be hierarchically organized, so that units not local to one library may be found in other libraries. Thus, programmers can work without interference on local versions of individual program units, while retrieving the remainder of the program from higher-level libraries.

Ada Run Time System

The Series 32000 Ada Run Time System provides comprehensive support for for tasking, debugging, exception handling, and input/output. ADEX, Verdix's Run Time System can be used in stand alone mode or VRTX/Series 32000 Real Time Operating System can be utilized. The Run Time System is linked with the generated users Ada program and to facilitate resource utilization efficiency, major portions of the Run Time System have been optimized. The Run Time System also interfaces with the symbolic debugger to provide comprehensive program check out facilities. Run Time sources for customization are also available.

SUPPORTED HARDWARE

—DEC VAX11 Family

—VMS Operating System version 2.X or later

SHIPPING PACKAGE

1600 bpi magnetic tape (9 track VMS copy format). User and reference documentation for Series 32000 VADS.

Section 10

Application Notes

Section 10 Contents

AN-383 Interfacing the NS32081 as a Floating-Point Peripheral 10-3
AB-26 Instruction Execution Times of FPU NS32081 Considered for Stand-Alone Configurations 10-11
AN-396 Debugging a 32032 Based System with an ISE16 10-12
AN-404 10 MHz, No Wait States NS32016 System 10-20
AN-405 Using Dynamic RAM with Series 32000 CPUs 10-31
AN-406 Interfacing the Series 32000 CPUs to the MULTIBUS 10-38
AN-449 A Systems-Oriented Microprocessor Bus 10-43

Interfacing the NS32081 as a Floating-Point Peripheral

National Semiconductor Corp.
Application Note 383
Microprocessor Applications
Engineering

This note is a guide for users who wish to interface the NS32081 Floating-Point Unit (FPU) as a peripheral unit to CPUs other than those of the Series 32000 family. This is not a particularly expensive procedure, but it requires some in-depth information not all of which is available in the NS32081 data sheet. Four basic topics will be covered here:

An overview of the architecture of the NS32081 as seen in a stand-alone environment.

The protocol used to sequence it through the execution of an instruction.

Special guidelines for connecting and programming the NS32081 as a peripheral component.

A sample application of these guidelines in the form of a circuit interfacing the NS32081 to the Motorola 68000 microprocessor.

References are made here to the NS32081 data sheet and the Series 32000 Instruction Set Reference Manual (Publication #420010099-001). The reader should have both these documents on hand.

1.0 Architecture Overview

1.1 REGISTER SET

The register set internal to the NS32081 FPU is shown in *Figure 1*. It consists of nine registers, each 32 bits in length:

FSR The Floating-Point Status Register. As given in the data sheet, this register holds status and mode information for the FPU. It is loaded by executing the LFSR instruction and examined using the SFSR instruction.

F0–F7 The Floating-Point Registers. Each can hold a single 32-bit single-precision floating-point value. To hold double-precision values, a register pair is referenced using the even-numbered register of the pair.

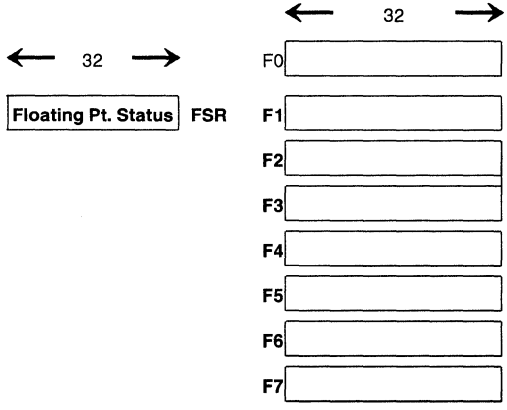


FIGURE 1. FPU Registers

Floating-point operands need not be held in registers; they may be supplied externally as part of the instruction sequence. Integer operands (appearing in conversion instructions) and values being transferred to or from the FSR must be supplied externally; they cannot be held in Floating-Point registers F0–F7.

1.2 INSTRUCTION SET AND ENCODING

The encodings used for NS32081 instructions are shown in *Figure 2*. They fall within two formats, labeled from Series 32000 tradition "Format 9" and "Format 11". These formats are distinguished by their least-significant byte (the "ID Byte"). Execution of an FPU instruction starts by passing first the ID Byte and then the rest of the instruction (the "Operation Word") to the FPU.

Fields within an instruction are interpreted by the FPU in the same manner as documented in Chapter 4 of the Series 32000 Instruction Set Reference Manual, with the exception of the 5-bit General Addressing Mode fields (*gen1, gen2*). Since the FPU does not itself perform memory accesses, it does not need to use these fields for addressing calculations. The only use it makes of these fields is to determine for each operand whether the value is to be found internal to the FPU (that is, within a register F0–F7, or whether it is to be transferred to and/or from the FPU. See *Figure 3*. A value of 0–7 in a *gen* field specifies one of the Floating-Point registers F0–F7, respectively, as the location of the corresponding operand. Any greater value specifies that the operand's location is external to the FPU and that its value will be transferred as part of the protocol. Any non-floating operand is always handled by the FPU as external, regardless of the addressing mode specified in its *gen* field. It is illegal to reference an odd-numbered register for a double-precision operand. If an odd register is referenced, the results are unpredictable.

1.3 PINOUT

The FPU is packaged in a 24-pin DIP (see *Figure 4*). The pin functions can be split into two groups: those that participate in the communication protocol between the FPU and the host system, and those that reflect the familiar requirements of LSI components.

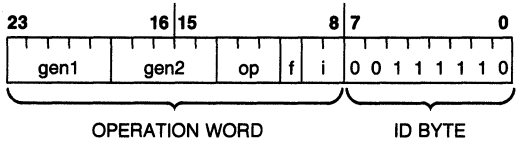
The protocol uses the following pins of the FPU:

D0–D15 The 16-bit data bus. The D0 pin holds the least-significant bit of data transferred on the bus.

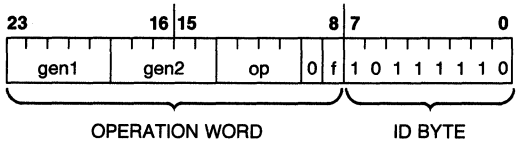
$\overline{SPC}$ A dual-purpose pin, low active. $\overline{SPC}$ is pulsed low from the host system as the data strobe for bus transfers. $\overline{SPC}$ is pulsed low by the FPU to signal that it has completed the internal execution phase of an instruction.

1.0 Architecture Overview (Continued)

ST0, ST1 The status code. This 2-bit value is sampled by the FPU on the falling edge of $\overline{SPC}$, and informs it of the current protocol phase. ST0 is the least-significant bit of the value. The need filled by the status code is most relevant to Series 32000-based systems, where it serves to allow retry of aborted instructions and to disambiguate the protocol when the $\overline{SPC}$ signal is bussed among multiple slave processors. In microprocessor-based peripheral applications, the status code can generally be provided from the CPU's address lines.



Format 9: LFSR/SFSR/Conversions



Format 11: Movement/Calculation

FIGURE 2. FPU Instruction Formats

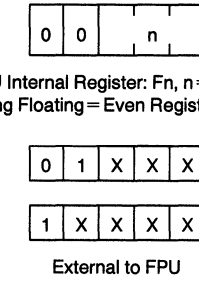
The pins providing for standard requirements are:

CLK The clock input. This is a TTL-level square wave which the FPU uses to sequence its internal calculations.

$\overline{RST}$ The reset input. This signal is used to reset the FPU's internal logic.

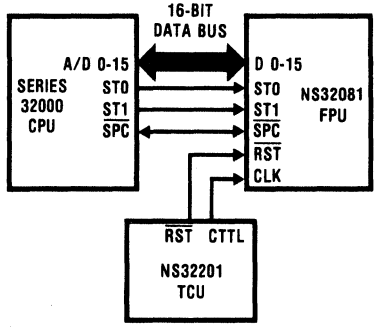
VCC The 5-volt positive supply.

GNDB, GNDL The grounding pins. GNDB serves as ground for the FPU's output buffers, and GNDL is used for the rest of the on-chip logic.

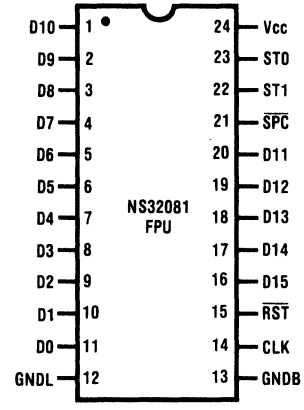


Note: All non-floating operands are always external.

FIGURE 3. FPU Addressing Modes



TL/EE/8388-1



Top View

TL/EE/8388-2

FIGURE 4. NS32081 FPU Connections

2.0 Protocol

The FPU requires a fixed sequence of transfers ("protocol") in its communication with the outside world. Each step of the protocol is identified by a status code (asserted to the FPU on pins ST0 and ST1) and by its position in the sequence, as shown in *Figure 5*.

Status Combinations:
- 11: Write ID Byte
- 01: Transfer Operation/Operand
- 10: Read Status Word

Step	Status	Action
1	11	CPU sends ID Byte on least-significant byte of bus.
2	01	CPU sends Operation Word, bytes swapped on bus.
3	01	CPU sends required operands, *gen1* first, least-significant word first.
4	xx	FPU starts internal execution.
5	xx	FPU pulses $\overline{SPC}$ low.
6	10	CPU reads Status Word (Error/Comparison Result).
7	01	CPU reads result (if any), least-significant word first.

FIGURE 5. FPU Instruction Protocol

Steps 1 and 2 transfer the instruction to the FPU. Step 1 transfers the first byte of the instruction (the ID Byte) and Step 2 transfers the rest of the instruction (the Operation Word). In Step 2, the two bytes of the Operation Word must be swapped on the bus; i.e. the most-significant byte of the Operation Word must be presented on the least-significant byte of the bus.

Step 3 is optional and repeatable depending on the instruction. It is used to transfer to the FPU any external operands that are required by the instruction. The operand specified by *gen1* is sent first, least-significant word first, followed by the operand specified by *gen2*. If an operand is only one byte in length, it is transferred on the least-significant half of the bus.

The FPU initiates Step 4 of the protocol, internal computation, upon receiving the last external operand word or, if there are no external operands, upon receiving the Operation Word of the instruction. During this time, the data bus may be used for any purpose by the rest of the system, as long as the $\overline{SPC}$ pin is kept pulled up by a resistor and is not actively driven.

Step 5 occurs when the FPU completes the instruction. The FPU pulses the $\overline{SPC}$ pin low to acknowledge that it is ready to continue the protocol. This pulse is called the "Done pulse". The bus is not used during this step, and remains floating.

In Step 6, the FPU is polled by reading a Status Word. This word indicates whether an exception has been detected by the FPU. In the Compare instruction (CMPf), it also displays the relationship between the operands and serves as the result. This transfer is mandatory, regardless of whether the information presented by the FPU is intended to be used. See *Figure 3-6* of the data sheet.

Step 7 is, like Step 3, optional and repeatable depending on the instruction. Any external result of an instruction is read from the FPU in this step, least-significant word first. If the result is a 1-byte value, it is presented by the FPU on the least-significant half of the bus (D0–D7).

Note: If in Step 6 the FPU indicates that an error has occurred, it is permissible, though not necessary, to continue the protocol through Step 7. No guarantee is made regarding the validity of the value read, but continuing through Step 7 will not cause any protocol problems.

If at any time within the protocol another ID byte is sent (ST = 11), the FPU will prepare itself internally to execute another instruction, throwing away the instruction that was in progress. This is done to support the Abort with Retry feature of the Series 32000 family.

Because of this feature, however, there is an important consideration when using the FPU in systems that support multitasking: the operating system must not allow a task using the FPU to be interrupted in the middle of an instruction protocol and then transfer control to another task that is also using the FPU. The partially-executed instruction would be thrown away, leaving the first task with a garbage result when it continues. This situation can be avoided easily in software but, depending on the system, some cooperation may be required from the user program. Other solutions involving some additional hardware are also possible.

3.0 Interfacing Guidelines

There are some special interfacing considerations that are required (see *Figure 6*):

1. The edges of the $\overline{SPC}$ pulse must have a fixed relationship to the clock signal (CLK) presented to the FPU. When writing information to the FPU, the pulse must start shortly after a rising edge of CLK and end shortly after the next rising edge of CLK. Failing to do so can cause the FPU to fail, often by causing it to freeze and not generate the Done pulse. This synchronous generation of $\overline{SPC}$ is also important when reading information from the FPU, but the $\overline{SPC}$ pulse is allowed to be two clocks in width. These requirements will be expressed in future NS32081 data sheets as a minimum setup time requirement between each edge of the $\overline{SPC}$ pulse and the next rising edge of CLK, currently set at 40 nanoseconds on the basis of preliminary characterization. The propagation delay in generating $\overline{SPC}$ through a Schottky flip-flop (e.g. 74S74) and a low-power Schottky buffer (e.g. 74LS125A) is therefore acceptable at 10 MHz. LS technology is recommended for the buffer to minimize undershoot when driving $\overline{SPC}$.
2. After the FPU generates the Done pulse, it is necessary to leave the $\overline{SPC}$ pin high for an additional two cycles of CLK before performing the Read Status Word transfer.
3. After performing the Read Status Word transfer, it is necessary to wait for an additional three cycles of CLK before reading a result from the FPU.

4.0 An Interface to the MC68000 Microprocessor

4.1 HARDWARE

A block diagram of the circuitry required to interface the MC68000 MPU to the NS32081 is shown in *Figure 7*.

First the easy part. Direct connections are possible on the data bus, which is numbered compatibly (D0–D15 on both parts), the status pins ST0–ST1 (connected to address lines A4–A5 from the 68000), and the clock (CLK on both). The system reset signal ($\overline{\text{RESET}}$ to and/or from the MC68000) should be synchronized with the clock before presenting it as $\overline{\text{RST}}$ to the FPU.

All that remains to be done is to generate $\overline{\text{SPC}}$ pulses that are within specifications whenever the 68000 accesses the FPU, and to detect the Done pulse from the FPU in a manner that will allow the 68000 to poll for it.

The approach selected for generating $\overline{\text{SPC}}$ pulses uses an address decoder that recognizes two separate address spaces; one to transfer information to or from the FPU ($\overline{\text{XFER}}$), and one to poll for the Done pulse ($\overline{\text{POLL}}$).

The 68000 signals $\overline{\text{AS}}$ (Address Strobe) and R/$\overline{\text{W}}$ (Read / not Write) are used to generate $\overline{\text{SPC}}$ timing.

Figure 8 shows the timing generated when the 68000 is writing to the FPU. The $\overline{\text{SPC}}$ pin is kept floating (held high by a pullup resistor) until bus state S4, at which point it is pulled low. On the next rising edge of CLK, $\overline{\text{SPC}}$ is actively pulled high, and is set floating afterward. It is not simply allowed to float high, as the resulting rise time can be unacceptable at speeds above about 4 MHz. A timing chain, required due to the 10-MHz 68000's treatment of its $\overline{\text{AS}}$ strobe, generates the signals TA, TB and TC, from which the $\overline{\text{SPC}}$ signal's state and enable are controlled.

Figure 9 shows the $\overline{\text{SPC}}$ timing for reading from the FPU. The basic difference is that $\overline{\text{SPC}}$ remains active for two clocks, so that the FPU holds data on the bus until it is sampled by the 68000. Again, $\overline{\text{SPC}}$ is actively driven high before being released.

Note: Although $\overline{\text{SPC}}$ must be driven high before being released, it must not be actively driven for more than two clocks after the trailing edge of $\overline{\text{SPC}}$. This is because the FPU can respond as quickly as three clocks after that edge with a Done pulse.

A simpler scheme in which the $\overline{\text{SPC}}$ pulse is identical for both reading and writing (1-clock wide always, but starting ½ clock later with CLK into the FPU inverted) was considered, but was rejected because the data hold time presented by the 68000 on a Write cycle would be inadequate at 10 MHz.

Any $\overline{\text{SPC}}$ pulse appearing while the $\overline{\text{XFER}}$ Select signal is inactive is interpreted as a Done pulse, which is latched in a flip-flop within the Done Detector block. When the 68000 performs a Read cycle from the address that generates the $\overline{\text{POLL}}$ select signal, the contents of the flip-flop are placed on data bus bit D15. Since this is the sign bit of a 16-bit value, the 68000 can perform a fast test of the bit using a MOVE.W instruction and a conditional branch (BPL) to wait for the FPU.

The schematic for the $\overline{\text{SPC}}$ generator and the Done pulse detector is given in *Figures 10a* and *10b*. The flip-flop labeled SPC generates the edges of the $\overline{\text{SPC}}$ pulse (on the signal $\overline{\text{SPCT}}$). The timing chain (TA, TB) provides the enable control to the buffer driving $\overline{\text{SPC}}$ to the FPU, as well as the signal to terminate the $\overline{\text{SPC}}$ pulse (either TB or TC, depending on the direction of the data transfer). Note that the timing chain assumes a full-speed memory cycle of four clocks in accessing the FPU, and will fail otherwise. The circuit generating the Data Acknowledge signal to the 68000 (DTACK, not shown) must guarantee this. In any system that must use a longer access, some modification to the timing chain will be necessary.

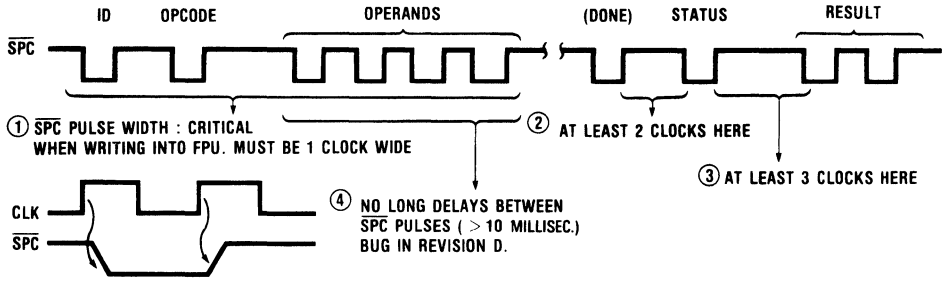
The flip-flop labeled DONE (*Figure 10b*) is the Done pulse detector. It is cleared by performing a data transfer into the FPU and is set by a Done pulse on $\overline{\text{SPC}}$. A buffer, enabled by the $\overline{\text{POLL}}$ select signal, connects its output to data bus bit 15.

4.2 SOFTWARE

Some notes on programming the FPU in a 68000 environment:

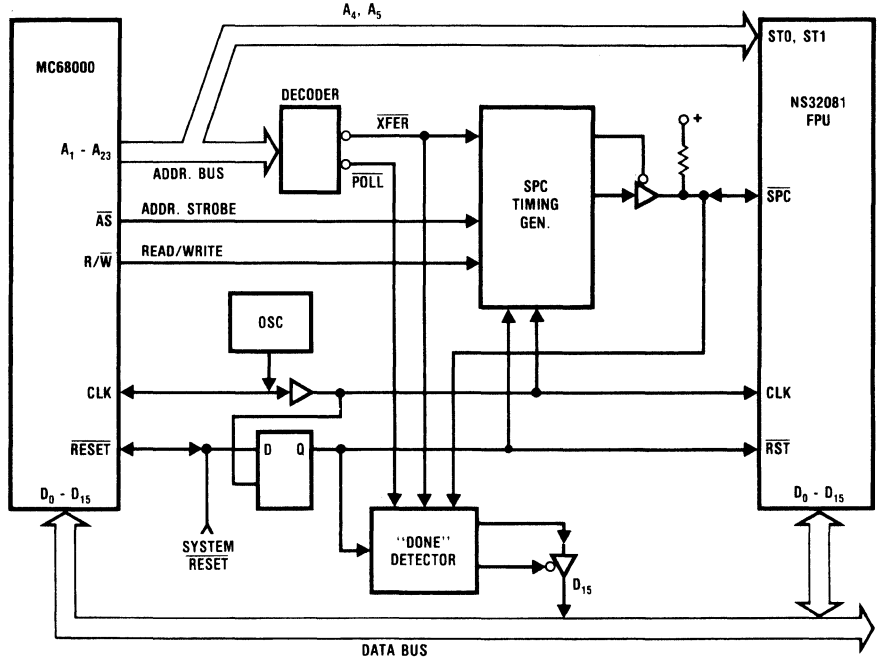
1. The byte addressing convention in the 68000 differs from that of the Series 32000 family. In particular, a byte with an even address is transferred on the most-significant half of the bus by the 68000, but the FPU expects to see it on the least-significant byte. When transferring a single byte to or from the FPU, either do so with an odd address specified, or transfer the byte as the least-significant half of a 16-bit value at an even address.
2. The 68000 transfers 32-bit operands by sending the most-significant 16 bits first. The FPU expects values to be transferred in the opposite order. Make certain that operands are transferred in the correct order (the 68000 SWAP instruction can be helpful for this).

A sample program that sequences the FPU through the execution of an ADDF instruction is listed in *Figure 11*. As this example is intended for clarity rather than efficiency, improvements are possible. The $\overline{\text{XFER}}$ select is assumed to be generated by addresses of the form 06xxxx (hex) and the $\overline{\text{POLL}}$ select is assumed to be generated by addresses of the form 07xxxx.



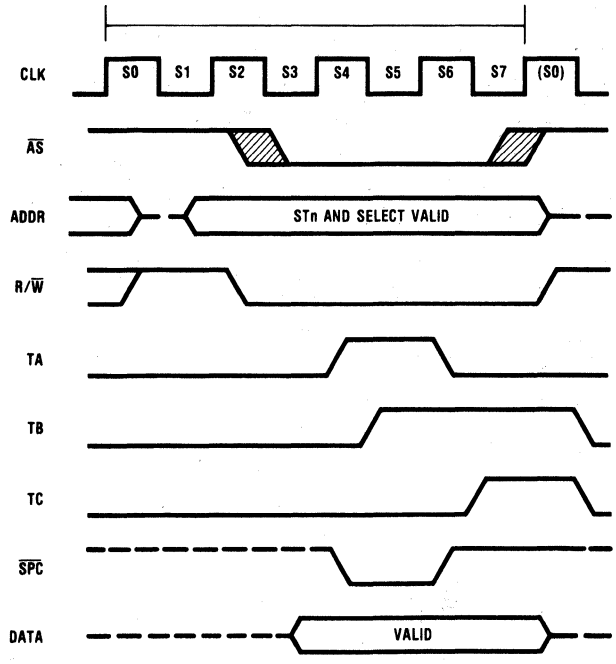
TL/EE/8388–3

FIGURE 6. Interfacing to FPU: Cautions



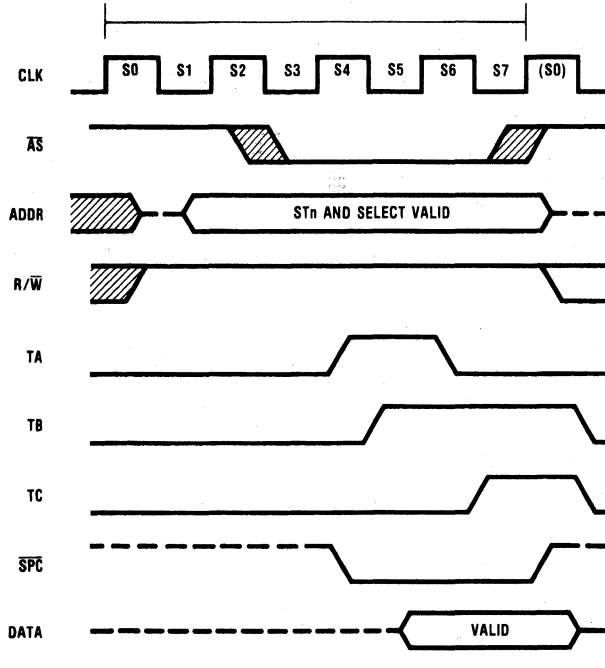
TL/EE/8388–4

FIGURE 7. 68000-32081 Interface Block Diagram



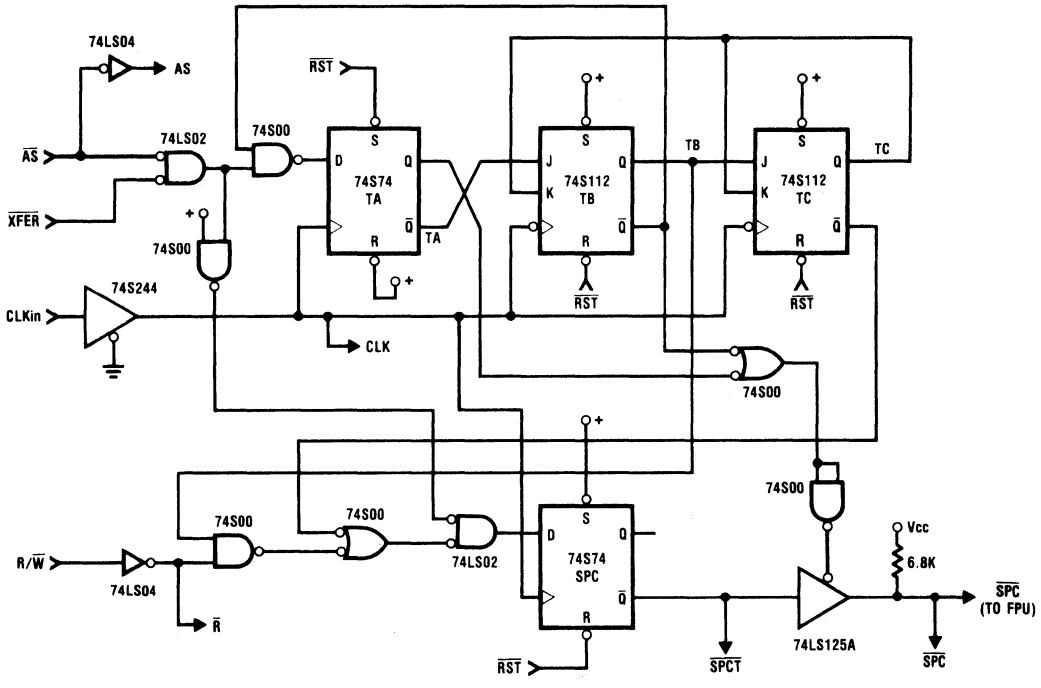
TL/EE/8388–5

FIGURE 8. 68000 Write to FPU



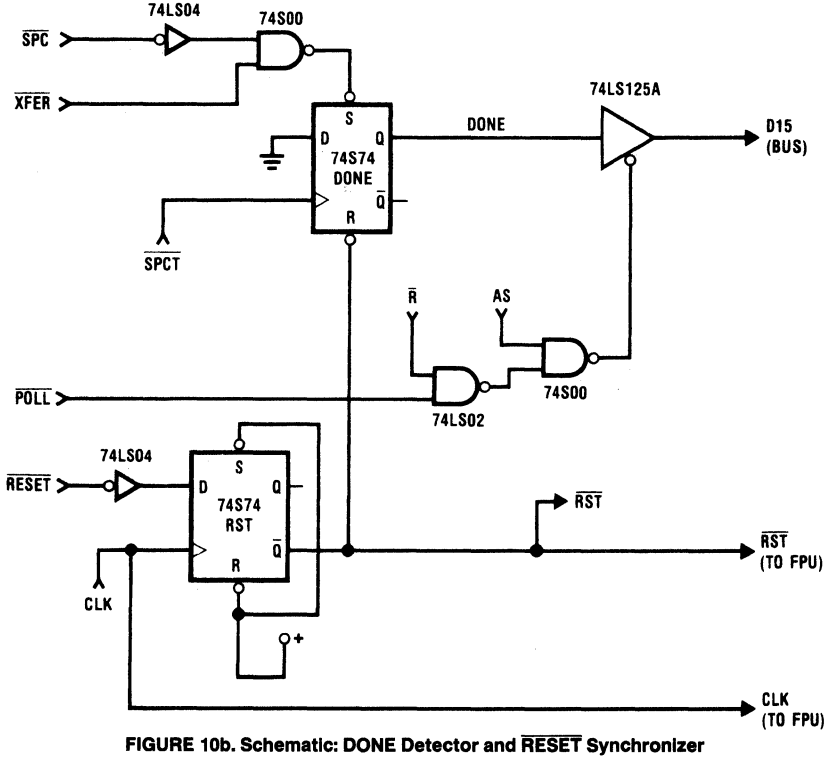
TL/EE/8388–6

FIGURE 9. 68000 Read from FPU



TL/EE/8388-7

FIGURE 10a. Schematic: $\overline{SPC}$ Timing Generator



TL/EE/8388-8

FIGURE 10b. Schematic: DONE Detector and $\overline{RESET}$ Synchronizer

```
*     Register Contents:
*
*     A0 = 00070000     Address of DONE flip-flop.
*     A1 = 00060010     Address for ST=1 transfer (Transfer Operand).
*     A2 = 00060020     Address for ST=2 transfer (Read Status Word).
*     A3 = 00060030     Address for ST=3 transfer (Broadcast ID).
*
*     D0 = 000000BE     ID byte for ADDF instruction.
*     D1 = 00000184     Operation Word for ADDF. (Note bytes swapped.)
*     D2 = 3F800000     First operand = 1.0.
*     D3 = 3F800000     Second operand = 1.0.
*     D4                Receives Status Word from FPU.
*     D5                Receives result from FPU.
*     D7                Scratch register (for DONE bit test).
*
START MOVE.W  D0,(A3)        Send ID byte.
      MOVE.W  D1,(A1)        Send Operation Word.
      SWAP    D2             Send operands. The swapping
      MOVE.L  D2,(A1)         is included because the
      SWAP    D2              FPU expects the least-
      SWAP    D3              significant word first.
      MOVE.L  D3,(A1)         (Can be avoided, with care.)
      SWAP    D3
*
POLL  MOVE.W  (A0),D7        Check the DONE flip-flop,
      BPL     POLL            loop until FPU is finished.
*                             (DONE bit is sign bit, tested
*                             by the MOVE instruction.)
*
      MOVE.W  (A2),D4        Read Status Word.
      MOVE.L  (A1),D5        Read result.
      SWAP    D5             Swap halves of result.
```

FIGURE 11. Single-Precision Addition (Demo Routine)

Instruction Execution Times of FPU NS32081 Considered for Stand-Alone Configurations

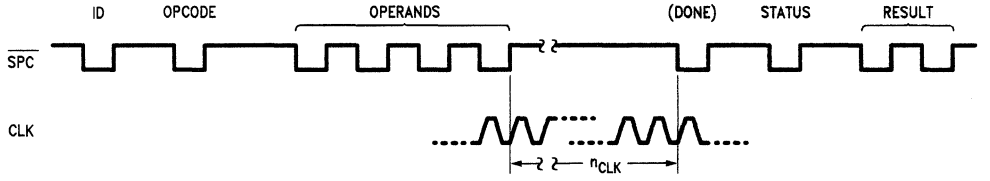
National Semiconductor Corp.
Application Brief 26
Systems & Applications Group

The table below gives execution timing information for the FPU NS32081.

The number of clock cycles nCLK is counted from the last SPC pulse, strobing the last operation word or operand into the FPU, and the Done-SPC pulse, which signals the CPU that the result is available (see *Figure 1*). The values are therefore independent of the operand's addressing modes and do not include the CPU/FPU protocol time. This makes it easy to determine the FPU execution times in stand-alone configurations.

The values are derived from measurements, the worst case is always assumed. The results are given in clock cycles (CLK).

Operation	Number of Clock-Cycles nCLK
Add, Subtract	63
Multiply Float	37
Multiply Long	51
Divide Float	78
Divide Long	108
Compare	38



TL/EE/8760–1

FIGURE 1

Debugging a 32032 Based System with an ISE/16™

National Semiconductor Corp.
Application Note 396
Microprocessor Applications
Engineering

An ISE/16 can be successfully used to debug a 32032-based system by using it in conjunction with a simple adapter. The adapter makes the 32032-based system look like a 32016-based system, as far as the ISE/16 is concerned. A debugging configuration using the adapter is shown in *Figure 1*; *Figure 2* shows the adapter block diagram. As can be seen, the adapter consists of address/data buffers that map the 32032's 32-bit wide data path into a 16-bit wide data path, and logic to generate the necessary control signals including the byte enable signals $\overline{BE0}$,...,$\overline{BE3}$.

Whenever a 32032-based system is being debugged via the ISE/16 and the adapter, a certain performance degradation should be expected. This is due to two factors. First, all memory accesses are limited to either 8 or 16 bits; if a double-word-aligned 32-bit quantity is referenced, two memory cycles are required instead of one. Second, the clock frequency has to be reduced because of the extra level of buffering and cables interposed between the target system and the ISE/16.

CIRCUIT SOLUTIONS

Two circuit solutions are shown in *Figures 4* and *5*. They differ in the way they handle FPU or other slave processors. The first circuit, shown in *Figure 4*, allows any slave in the target system to be accessed. The one in *Figure 5* does not allow slave accesses to the target system; if an FPU is needed, it must be placed in the adapter board. This second solution has the advantage of being simpler than the first one. In the case of *Figure 4*, some extra logic, including a PAL, is needed to check the slave ID byte during an ID broadcast cycle to decide whether or not to enable the data buffers during slave read cycles.

Note that, during MMU slave read cycles the data buffers must not be enabled since the MMU will drive the bus. This extra logic must also prevent the address/data buffers from getting enabled during either slave access cycles or idle cycles. The delays shown in the circuit diagrams are used to avoid possible bus contentions. The one marked 'D1' is used in both circuits and its function is to delay the buffer enable signals during memory write cycles.

Note that the address buffer (74AS244) used to output the address signals A16,...,A23, is disabled at the beginning of the CPU state T2. This could cause signal contentions if the data swap-buffer that handles the signal lines AD16,...,AD23 were enabled at the same time.

The second delay is present only in the first circuit and is used to delay the $\overline{SPC}$ pulse until the $\overline{DDIN}$ signal from the CPU is valid. This is to eliminate spurious signals on the enable pins of the low-order-word data buffers during slave transfers, thus avoiding possible conflicts between these buffers and either the CPU or the slave. Neither of the above delays needs to be accurate and each can be implemented by cascading some spare gates. *Figure 3* shows the PAL® equations in PALASM™ format for the PAL that controls the slave accesses. *Figure 6* provides a timing diagram for memory access cycles.

As shown in *Figures 4* and *5*, several signal lines from the adapter to the target system are not buffered since the target cables are assumed to be very short. If longer cables are needed, these signals should be buffered.

The address/data buffers are Advanced Schottky devices with propagation delays of approximately 8 ns. These delays must be added to the ISE/16 buffer and cable propagation delays, and to the adapter cable delays, to determine the resulting timing at the 32032 socket on the target system. This is needed to determine the clock frequency to be used during debugging.

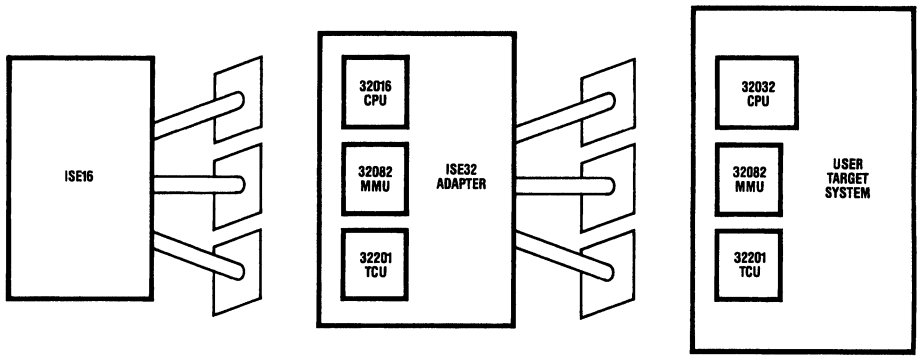
See Chapter 6 of the ISE/16 User Manual for more details.

JUMPER SETTINGS

The jumpers on the adapter board should be configured according to the user requirements. There are three basic cases to consider.

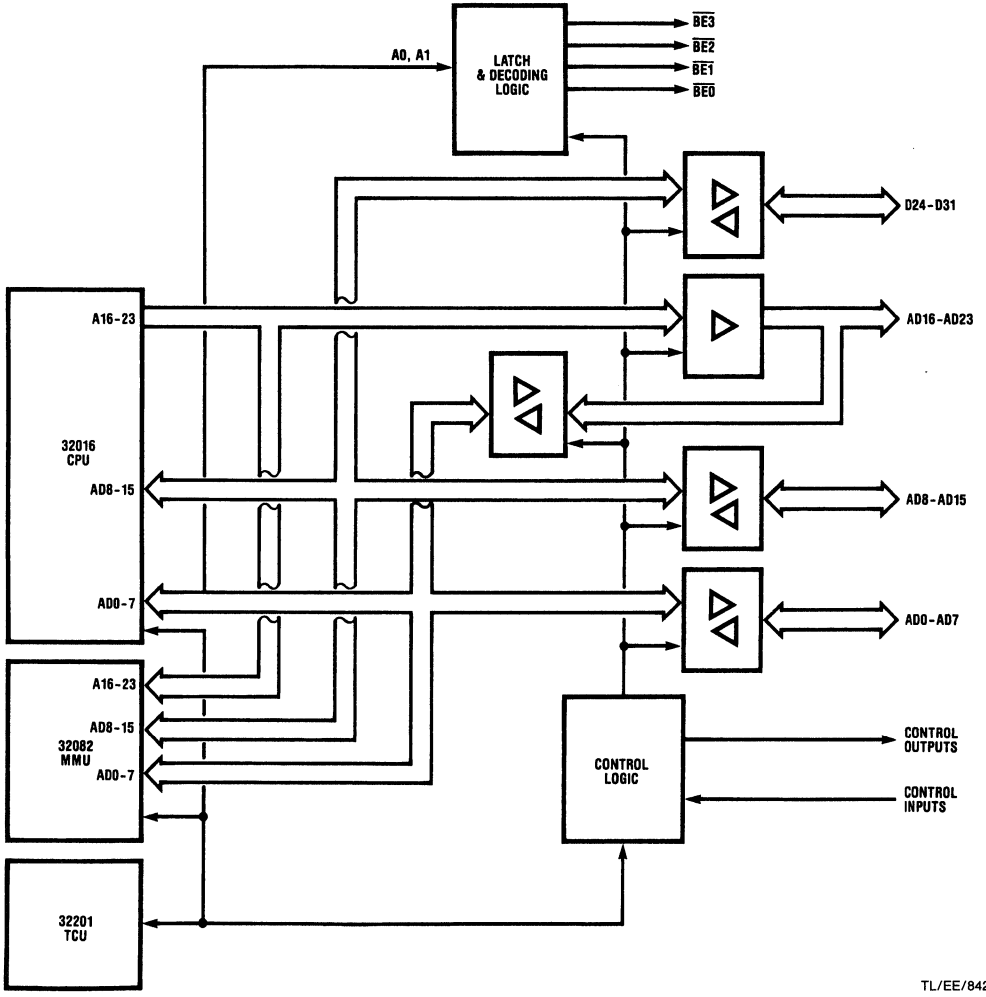
1. The target system does not use the MMU. In this case neither the MMU target cable from the ISE/16 nor the MMU cable from the adapter to the user system are needed. Jumpers W1 (1-3, 2-4), W2, W3 and W4 (1-2) must be installed.
2. The MMU is used on the target system, but the signals A24 and $\overline{INT}$ from the MMU are not used. In this case the MMU target cable from the ISE/16 is required but the MMU cable from the adapter to the target system is not needed. The $\overline{PAV}$ and $\overline{HLDAO}$ signals from the MMU can be routed to the $\overline{ADS}$ and $\overline{HLDA}$ lines of the CPU cable, by installing jumpers W1 (1-2) and W4 (1-2). Jumpers W2 and W3 must be removed. In addition, on the target system, the signals $\overline{ADS}$ and $\overline{HLDA}$ must be connected to $\overline{PAV}$ and $\overline{HLDAO}$ respectively.
3. The MMU is used on the target system, and so are A24 and $\overline{INT}$. In this case both the MMU target cable from the ISE/16 and the MMU cable from the adapter to the target system are required. The jumpers W1 (1-3) and W4 (1-3) on the adapter must be installed while W2 and W3 must be removed.

The setting of jumper W5 is only relevant if either the clock from the target system is required during debugging or a 32016 CPU together with a TCU and MMU is used instead of the ISE/16. If the ISE/16 internal clock is used during debugging the setting of W5 is irrelevant.



TL/EE/8427–1

FIGURE 1. ISE/32 Adapter General Configuration

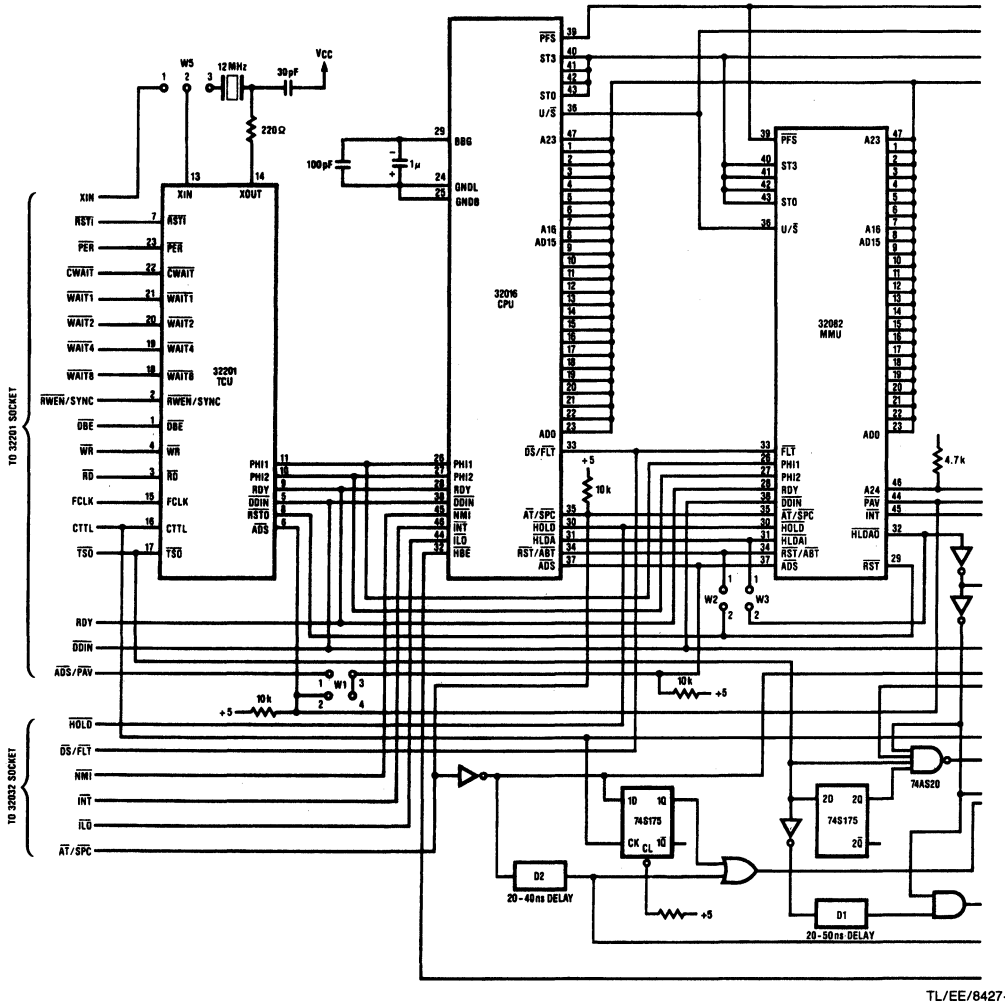


TL/EE/8427–2

FIGURE 2. ISE/32 Adapter Block Diagram

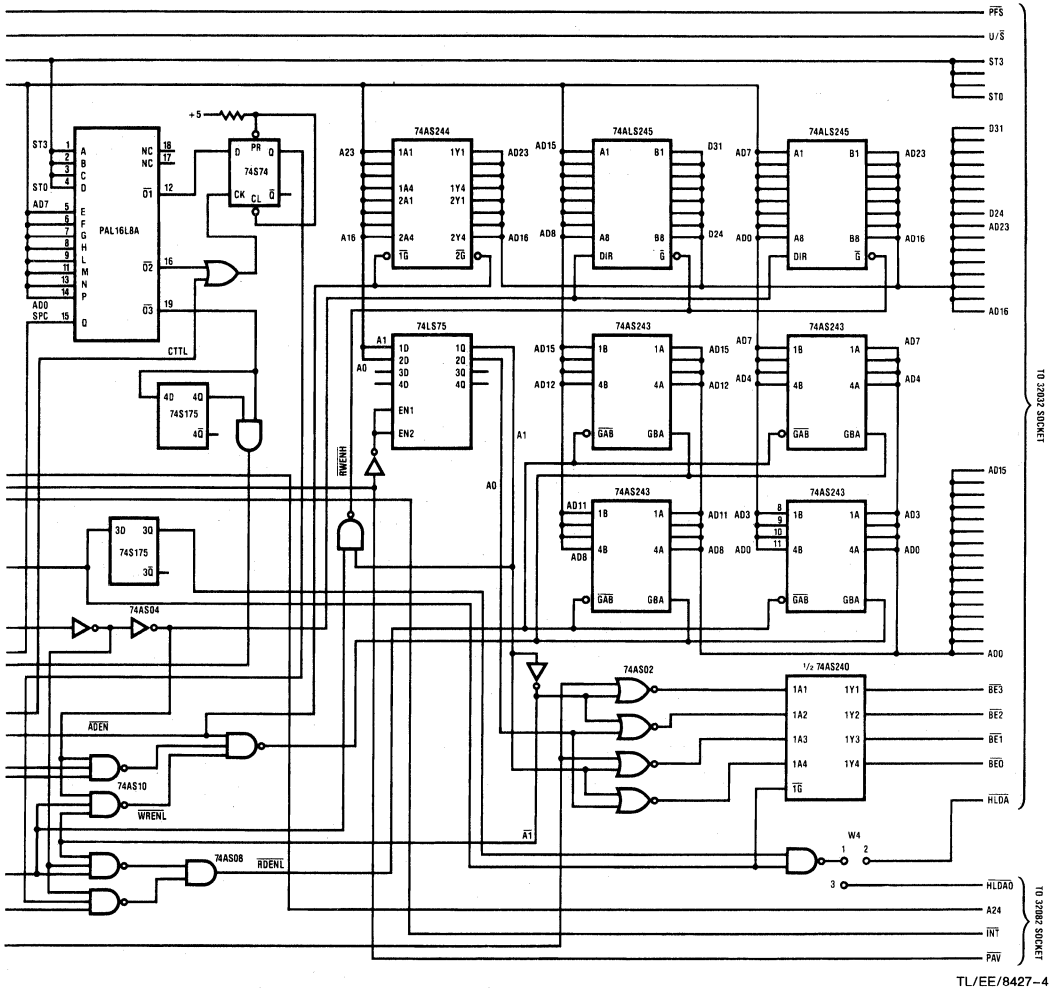
```
PAL16L8A
PART#
SLAVE ACCESS CONTROL
NATIONAL SEMICONDUCTOR
A B C D E F G H L GND
M O1 N P Q O2 NC NC O3 VCC
/O1 = /E*/F*/G*H*L*M*N*/P
/O2 = A*B*C*D*Q
/O3 = A*B*C + A*B*/C*D + /A*/B*D
DESC
```

FIGURE 3. PAL Equations in 'PALASM' Format



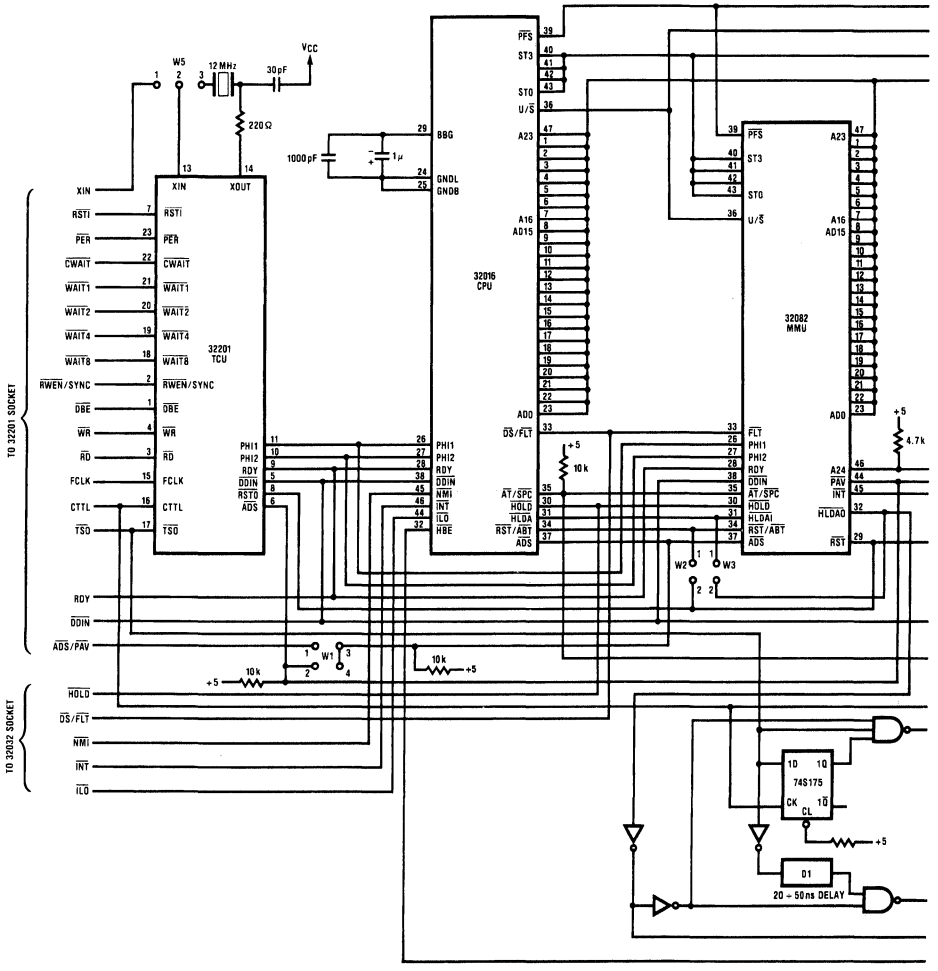
TL/EE/8427–3

FIGURE 4. ISE/32 adapter circuit diagram. Slave accesses to the target system are allowed.



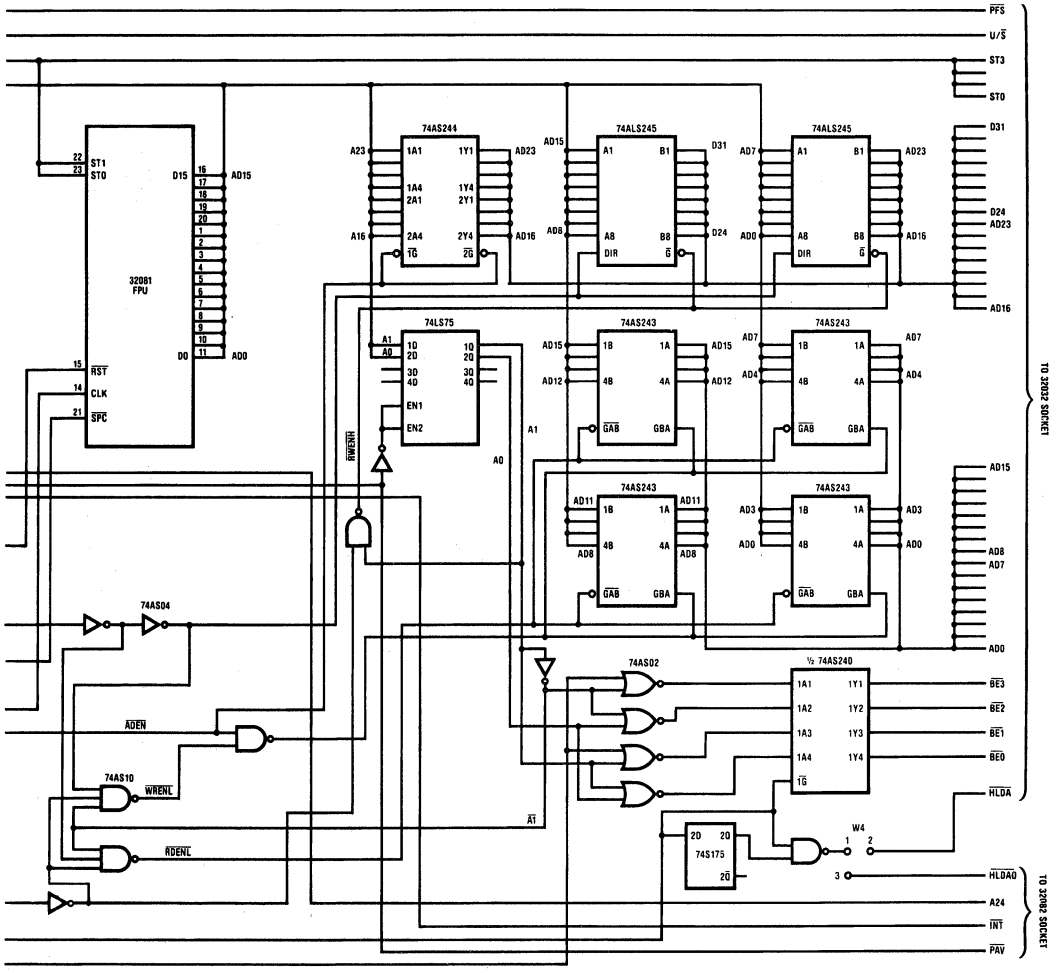
TL/EE/8427–4

FIGURE 4. ISE/32 adapter circuit diagram. Slave accesses to the target system are allowed. (Continued)



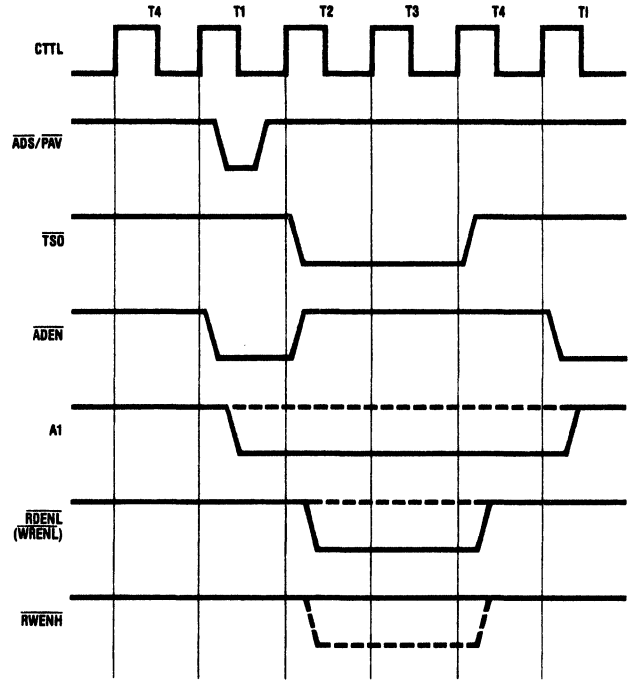
TL/EE/8427–5

FIGURE 5. ISE/32 adapter circuit diagram. Slave accesses to the target system are not allowed.



TL/EE/8427–6

FIGURE 5. ISE/32 adapter circuit diagram. Slave accesses to the target system are not allowed. (Continued)



TL/EE/8427–7

FIGURE 6. Memory Read (Write) Timing Diagram

10 MHz, No Wait States NS32016 System

National Semiconductor Corp.
Application Note 404
Microprocessor Applications
Engineering

INTRODUCTION

Recent microprocessor applications such as high resolution graphics, multiuser workstations, data communication, industrial automation, etc. have placed growing demands on microprocessor throughputs. Higher throughputs, together with increasing complexity of microprocessor systems, require slave support in addition to high speed, powerful microprocessors.

The Series 32000® Microprocessor family serves the needs of high end microprocessor applications. The NS32016 Central Processing Unit (CPU) has a powerful register and instruction set. The NS32082 Memory Management Unit (MMU) and the NS32081 Floating-Point Unit (FPU) function as slave processors for the CPU. All the chips in the family run at 10 MHz. Together, the chips provide the throughput required by high end microprocessor applications. This application note discusses the design considerations for building a 10 MHz NS32016 system with no wait states.

Series 32000 Chip Set:

The NS32016 system described here, uses the Series 32000 chip set consisting of:

1. The NS32016 Central Processing Unit (CPU)
2. The NS32082 Memory Management Unit (MMU)
3. The NS32081 Floating-Point Unit (FPU)
4. The NS32201 Timing Control Unit (TCU)
5. The NS32202 Interrupt Control Unit (ICU)

Details of the five chips are provided in the Series 32000 Data Book. *Figure 1* illustrates the interconnections of a simple NS32016 based system capable of running at 10 MHz without wait states. As shown in *Figure 1*, the CPU, MMU and FPU are interconnected on a multiplexed Address/Data Bus. The TCU provides the clocks and the control signals required by the system. The multiplexed bus is separated into Data and Address buses by using bidirectional Data bus drivers and fall-through Address latches. The ICU, being a peripheral, is interfaced to the demultiplexed Address and Data buses. The ICU Status input (ST1) is driven by a logical combination of ST1 from the CPU and address line A5. This allows the CPU to read both the INTA and RETI vectors from the SVCT register of the ICU.

DESIGN CONSIDERATIONS

The design of a 10 MHz Microprocessor system with no wait states requires system memory to run at comparable speeds. Typically, system memory consists of Read Only Memory (ROM) and Read/Write or Random Access Memory (RAM). A 10 MHz, NS32016-based system functioning without wait states requires careful memory timing consideration.

A read cycle without wait states requires data from memory to be valid prior to the falling edge of the PHI2 clock during the T3 state of the CPU. This allows about 155 nanoseconds following the leading edge of the read strobe for data to be stable. In a memory write cycle, the data is available to the memory for about 215 nanoseconds following the leading edge of the write strobe. It is assumed that the Address lines are valid at the memory pins at the time the read or write strobe goes active.

SRAM INTERFACE:

With high speed, 8K × 8-bit Static RAMs (SRAMs) such as the NMC6264s, which have a 120-nanosecond maximum access time, an interface without wait states is feasible. Besides requiring no wait states, SRAMs do not require the refresh circuitry that Dynamic RAMs (DRAMs) need. Neither do they require the error checking and correcting circuitry that DRAMs need for correcting soft errors. The timing diagrams for SRAM Read and Write cycles with the MMU are illustrated in *Figures 4* and *5* respectively. The SRAMs are organized into even and odd banks. The Write Enable ($\overline{WE}$) signals for the SRAMs are generated by logically combining address line A0 and $\overline{HBE}$ with the $\overline{WR}$ signal from the TCU. Both even and odd SRAM banks are always enabled during Read cycles.

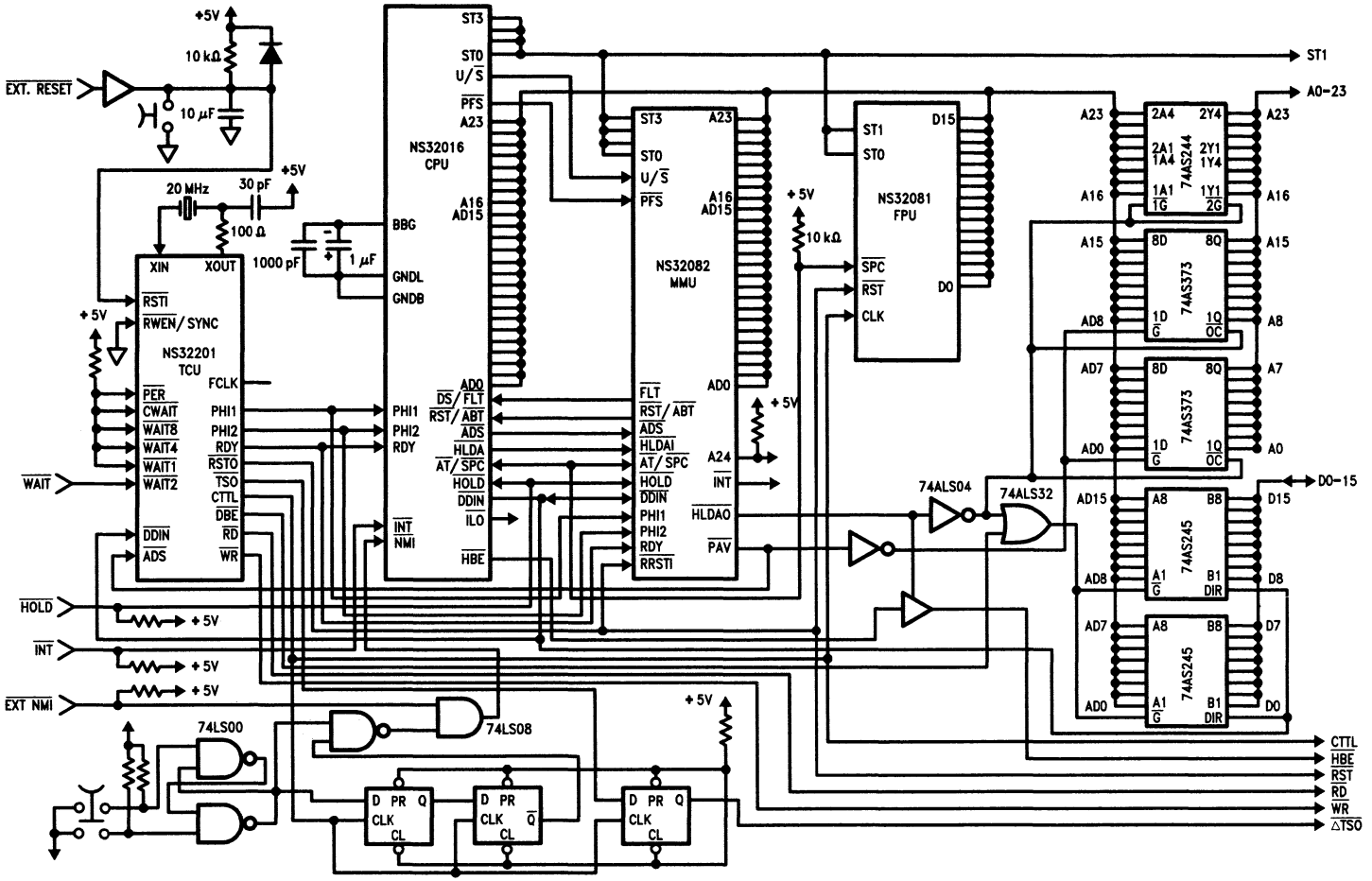
EPROM INTERFACE:

With current technology, EPROMs up to 64-Kbyte densities are available. In particular, the 27128 type EPROMs are available with 150-nanosecond maximum access times. These EPROMs can be used in the NS32016-based system without wait states. The timing diagram for the EPROM Read cycle, with the MMU is illustrated in *Figure 3*. The Output Enable ($\overline{OE}$) inputs to the EPROMs are connected to the $\overline{RD}$ signal. This will cause both even and odd bytes of the set of EPROMs to be enabled for a byte or word read from the CPU. This does not affect the data as the CPU will read the appropriate byte(s). A single DMPAL16L8A device is used to generate all the required chip select signals.

I/O INTERFACE:

CPU accesses to the serial communications devices require the insertion of at least two wait states. This is accomplished by activating the TCU $\overline{WAIT2}$ input during such accesses.

Furthermore, the leading edges of the Read and Write strobes are delayed by one clock cycle. This is necessary since the time delays of the Read and Write strobes from Address Valid, required by the communications devices, are larger than the delays provided by the 32000 chip set during normal bus accesses. The system uses two NS16450s. This facilitates its use in stand-alone, stand-aside or transparent configuration. The two NS16450s have their oscillator pins (XTAL1 and XTAL2) connected to the crystal circuit as illustrated in *Figure 1*. The two NS16450s are interfaced to standard RS232C communication ports with jumpers. The jumpers allow the configuration of either port as a data-terminal or as a data-set.



TL/EE/8506-7

FIGURE 1. Circuit Diagram for the 10 MHz No Wait States NS32016 Based System

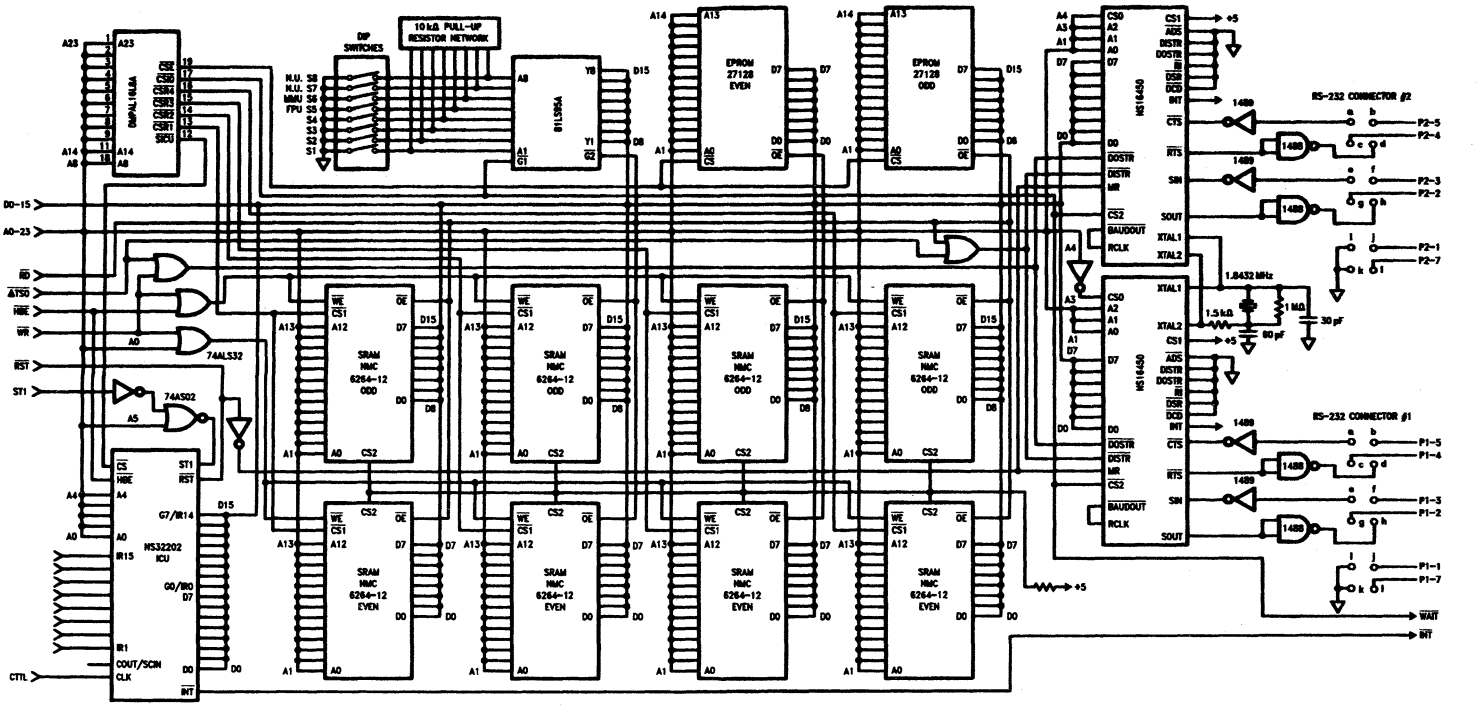


FIGURE 1. Circuit Diagram for the 10 MHz No Wait States NS32016 Based System (Continued)

One port can be used to communicate with a host computer. The other can be used to interface the system to a terminal. If MON16 software is used, it is possible to communicate from the terminal to a host computer such as a National Semiconductor SYS/32™ or a VAX™. Files stored in the host can be down-loaded into the NS32016-based system memory and executed at 10 MHz without wait states.

The Non-Maskable Interrupt signal ($\overline{NMI}$) is used to return from "runaway" programs to the monitor without destroying the contents of the Program Counter and Processor Status Register. The circuit shown in *Figure 1* provides an $\overline{NMI}$ pulse signal to the CPU.

```
PAL16L8A
Part #
Chip Select Generation
National Semiconductor
A23 A22 A21 A20 A19 A18 A17 A16 A15 GND
A14 SICU CSR1 CSR2 CSR3 CSR4 CSIO A8 CSE VCC
/CSE = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * /A16 * /A15
/CSR1 = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * /A16 * A15 * /A14
/CSR2 = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * /A16 * A15 * A14
/CSR3 = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * A16 * /A15 * /A14
/CSR4 = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * A16 * /A15 * A14
/CSIO = A23 * A22 * A21 * A20 * A19 * A18 * A17 * A16 * A15 * /A14
/SICU = A23 * A22 * A21 * A20 * A19 * A18 * A17 * A16 * A15 * A14 * /A8
```

FIGURE 2. PAL Equations in PALASM™ Format

CONFIGURATION SWITCHES:

Dip switches have been used in the circuit for system configuration as illustrated in *Figure 1*. The CPU reads them at power-on or system reset to set the baud rate of the I/O ports and the CPU configuration register. Switches S1, S2, S3 and S4 set the baud rate. Table I lists the various baud rates possible with MON16 software. Switch S5 indicates the presence of an FPU in the system and S6 indicates the presence of an MMU in the system (Table II).

TABLE I

S4	S3	S2	S1	Baud Rate
ON	ON	ON	ON	19200
ON	ON	ON	OFF	9600
ON	ON	OFF	ON	7200
ON	ON	OFF	OFF	4800
ON	OFF	ON	ON	3600
ON	OFF	ON	OFF	2400
ON	OFF	OFF	ON	2000
ON	OFF	OFF	OFF	1800
OFF	ON	ON	ON	1200
OFF	ON	ON	OFF	600
OFF	ON	OFF	ON	300
OFF	ON	OFF	OFF	150
OFF	OFF	ON	ON	134
OFF	OFF	ON	OFF	110
OFF	OFF	OFF	ON	75
OFF	OFF	OFF	OFF	50

TABLE II

S6	S5	Slave Processors
ON	ON	MMU and FPU
ON	OFF	MMU
OFF	ON	FPU
OFF	OFF	neither

RS232C JUMPER CONNECTIONS:

If a particular port in the system is to be connected to a terminal, the associated jumpers need to be configured for a Data Set. With reference to *Figure 1*, the jumper connections for a Data Set configuration are as follows:

a–c, b–d, e–g, f–h, i–j, k–l.

If the port is to be connected to a host computer, the associated jumpers need to be configured for a Data Terminal. The jumper connections for a Data Terminal configuration are as follows:

a–b, c–d, e–f, g–h, i–j, k–l.

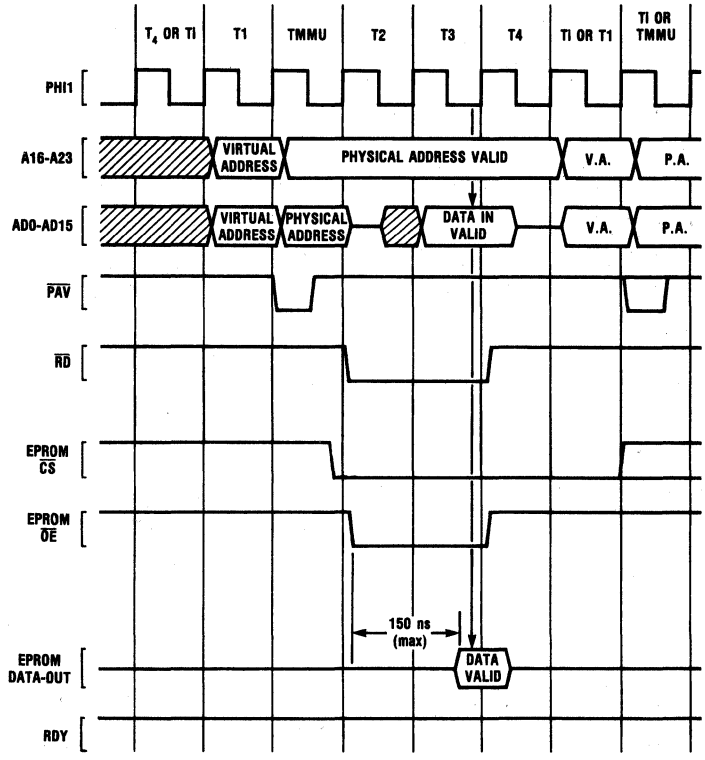
Note: a–b => connect node 'a' to node 'b'.

MEMORY MAP

The memory map of the system described in this note is slightly different from the memory map of the DB32016 CPU board. This has been done to simplify the chip-select generation logic. This requires minor changes to some 'equate' statements in the MON16 modules in addition to the I/O drivers changes to support the NS16450s instead of the 8251s. *Figure 2* shows the PAL equations. The memory map is shown in Table III.

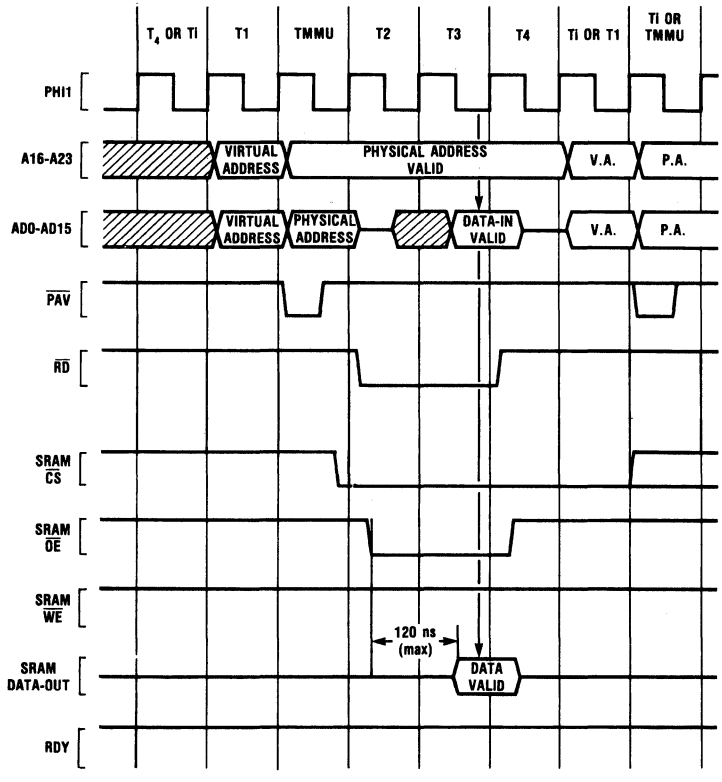
TABLE III

Devices	Memory Locations
EPROMs	$000000-$007FFF
SRAMs	$008000-$017FFF
Serial Port 1	$FF8000-$FF800F
Serial Port 2	$FF8010-$FF801F
ICU-Registers	$FFFE00-$FFFE3F
CNFG Switches	$FF8003



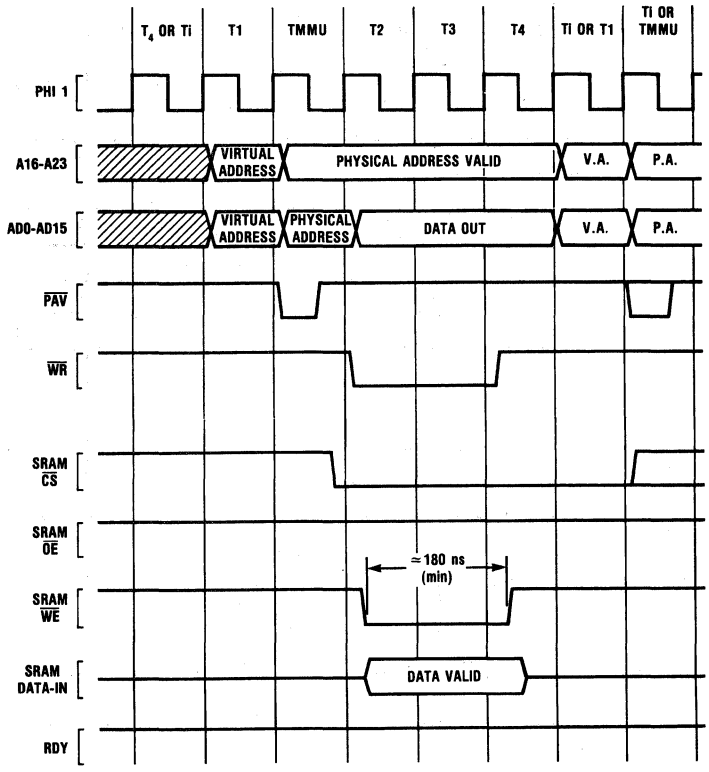
TL/EE/8506–3

FIGURE 3. Memory-Managed EPROM Read Cycle



TL/EE/8506–4

FIGURE 4. Memory-Managed SRAM Read Cycle



TL/EE/8506–5

FIGURE 5. Memory-Managed SRAM Write Cycle

Notes:

1. In all memory and I/O cycles, if the MMU is not used, then the T_{mmu} cycle is absent. See *Figures 3, 4* and *5*. This will not change the memory or I/O cycle requirements. The $\overline{PAV}$ signal will be replaced by the $\overline{ADS}$ signal which is strobed by the CPU in the T1 cycle.

2. The chip selects for the ICU internal registers and the Configuration latch have been partially decoded in order to reduce the chip count for the system.

DECOUPLING CAPACITOR REQUIREMENTS:

Line to ground noise on the system can be eliminated by using decoupling capacitors. For the Series 32000 chip set, the decoupling capacitor details are given in the Series 32000 Data Book. For the random logic used in the system, a 0.1 μF ceramic capacitor for each bipolar TTL device is recommended. For MOS devices, a 0.1 μF ceramic capacitor for every row of four to five devices may be used. At the power input to the system, a 100 μF tantalum capacitor in parallel with a 0.1 μF ceramic capacitor may be used.

PORTING MON16:

The changes made to MON16 to run on this system are in modules MONINT.ASM and MONSUB.ASM. All changes appear in lower-case letters in the listing. The code pertaining to the ICU counters in MON16 has been removed as it is not used.

```
;          MONINT**************************
;   USART CONSTANTS
;
iobeg:    .equ    @h'ff8000              ;IO start address
usrtl:    .equ    @h'ff8000              ;UART0
usrtcsl:  .equ    @h'ff800a              ;UART0 line status register
usrt2:    .equ    @h'ff8010              ;UART1
usrtcs2:  .equ    @h'ff801a              ;UART1 line status register
datap:    .equ    0                      ;UARTs rev/trans buffer registers
out_rdy:  .equ    5                      ;UARTs tx_rdy bit of the LSR reg.
in_rdy:   .equ    0                      ;UARTs rx_rdy bit of the LSR reg.
switchp:  .equ    @h'ff8003              ;Dip switches port address




;
;   RESET ROUTINE




;
;     init UARTs
;
          save    [r2, r3, r4, r5]
          addr    usrtl, r4
          addr    usrt2, r5
          movb    h'80, 6 (r4)
          movb    h'80, 6 (r5)
;set UARTs baud rate
;         movzbd switchp, r2             ;load switch for baud rate
;         andb    h'0f, r2
;         movb    acetb:w[r2:w],0(r4)    ;set UART0 baud rate
;         movb    acetb+1:w[r2:w],2 (r4)
;         movb    acetb:w[r2:w],0(r5)    ;set UART1 baud rate
;         movb    acetb+1:w[r2:w],2 (r5)
;***for debugging only***
          movb    h'0c,0 (r4)            ;set UART0 to 9600 baud
          movb    h'0,2 (r4)
          movb    h'18,0 (r5)            ;set UART1 to 4800 baud
          movb    h'0,2 (r5)
;************************
          movb    h'3,6 (r4)             ;set LCR of the UARTs
          movb    h'3,6 (r5)
          movb    h'0f,8 (r4)
          movb    h'0f,8 (r5)
          movb    0 (r4), r3
          movb    0 (r5), r3
          restore[r2,r3,r4,r5]
```

Note: Version 2.00 of MON16 has been used for the NS32016 system.

```
;
;the divisor values for UARTs with crystal frequency of 1.8432 MHz
;
acetb:   .word 6,12,16,24              ;19200, 9600, 7200, 4800
         .word 32,48,58,64             ;3600, 2400, 2000, 1800
         .word 96,192,384,768          ;1200, 600, 300, 150
         .word 857,1047,1536,2304      ;134.5, 110, 75, 50
;
         MOVD   H'1B10000,SVMSR        ;INIT SVMSR SET TU BEN UB FT UT
         MOVD   H'90000,MNMSR          ;MONITOR MSR: = TU,A0
         movb   switchp,r1             ;GET MMU & FPU BITS FROM SWITCHES
         COMB   R1,R1                  ;CONVERT TO CFG BYTE
         ANDB   SW__MMU+SW__FPU, R1
         ASHB   -3,R1
         MOVZBD CFGN,R2                ;PREPARE CALL TP GET__PUT
         MOVB   PUTI,TOS
         MOVQD  0,TOS                  ;GET__PUT (GET,0,CFG);
         CXP    GETPUT
         TBITB  CNFMMU,CONFIG          ;IF MMU THEN
         BFC    RST14:B
         LMR    MSR,MNMSR              ; LOAD MSR;
RST14:   CXP    MAINLP                 ;TYPE RESET MESSAGE




;
         .MODULE MONSUB**********
;
;        USART  CONSTANTS
;
iobeg:   .equ   @h'ff8000       ;IO start address
usrt1:   .equ   @h'ff8000       ;UART0
usrtcs1: .equ   @h'ff800a       ;UART0 line status register
usrt2:   .equ   @h'ff8010       ;UART1
usrtcs2: .equ   @h'ff801a       ;UART1 line status register
datap:   .equ   0
out__rdy:.equ   5
in__rdy: .equ   0
usrtoff: .equ   h'a             ;line status reg offset from buffer reg
;
```

```
;
;    R D C H R      ( DUMMY READ CHAR PROCEDURE )
;
;
RDCHR:   .PROC                    ; PROCEDURE RDCHR (WAIT,TRM)
RD_CHR: .BLKB                     ; PROCEDURE VALUE
RD_WAIT:.BLKB                     ; WAIT/NOWAIT FLAG
RD_TRM: .BLKB                     ; TERMINAL NUMBER
        .RETURNS
        .BLKW                     ; RETURN CHR,CHR_RDY FLAG
        .VAR   [R1,R2]
        .BEGIN
        addr   usrt1,rl           ; R1: ADDRESS OF TRMINAL A
        CMPQB  TRMA,RD_TRM        ; IF TRMINAL_NUM < > 0 THEN
        BEQ    RDCHRLP:B
        addr   usrt2,rl           ; R1: ADDRESS OF TRMINAL B
RDCHRLP:                          ; DO WHILE IN_RDY=0 AND RD_WAIT=TRUE
        tbitb  in_rdy,usrtoff(rl)       ;   INPUT IN_RDY
        BFS    RDCHR3:B
;       BR     RDCHR3:B           ; *** FOR DEBUG ONLY ***
        CMPQB  TRUE,RD_WAIT
        BEQ    RDCHRLP            ; END;
        BR     RDCHREX:B
RDCHR3: MOVB   0(R1),RD_CHR  ; RDCHR;=USART DATA
        MOVQB  TRUE,RD_WAIT       ; RD_WAIT:=TRUE
RDCHREX:
        .ENDPROC




;
;       P R C H R      ( PRINT CHARACTER )
;
;       FUNCTION — SEND ONE CHARACTER TO TERMINAL
;
;       CALLING SEQUENCE PRCHR(ENDF,WAIT,CHR,TRM)
;
;        ENDF/WAIT
;               BOOLEANO  IN/OUT ON INPUT FLAGE WAIT TO END OF OPERATION
;                                OR REURN
;                                ON OUTPUT INDICATES END OF OPERATION
;       CHR  — CHARACTER INPUT CHARACTER TO BE PRINTED
;       TRM  — INTEGER   INPUT TERMINAL NUMBER
;
PRCHR:   .PROC
WAIT_PR:.BLKB                     ; WAIT : BOOLEAN
CHR PR: .BLKB                     ; ASCII CHR
```

```
TRM__CHR:.BLKB                  ; TERMINAL NUMBER
         .RETURNS
         .BLKB                  ; OUTPUT WAIT WAIT:BOOLEAN
         .VAR   [R1,R2]
         .BEGIN
         addr   usrt1,r1        ;R1: ADDRESS OF TERMINAL A
         CMPQB  TRMA,TRM__CHR   ;IF TERMINAL__NUM<>0 THEN
         BEQ    PRCHRLP:B
         addr   usrt2,r1        R1: ADDRESS OF TERMINAL B
PRCHRLP:
         tbitb  out__rdy,usrtoff(r1) ;IF TX-RDY = 0
         BFS    PRCHR3:B        ;THEN
;        BR     PRCHR3:B             ;***DEBUG ONLY***
         CMPQB  FALSE,WAIT__PR  ; IF WAIT THEN REPEAT
         BNE    PRCHRLP
         BR     PRCHREX:B            ; ELSE WAIT:=FALSE
PRCHR3:  MOVB   CHR__PR,0 (R1)
         MOVQB  TRUE,WAIT__PR   ;ELSE WRITE (DATA-PORT, CHR)
```

CONCLUSION

This application note describes a method of designing a 10-MHz, no-wait-state NS32016-based system with off-the-shelf memory and I/O chips. The system has a powerful instruction set, suitable for high level language compilers. With available cross-support software (NSX16™) and firmware (MON16), the NS32016 system can be used to communicate with a host computer such as a SYS/32. Programs can be written in high level languages such as C on the SYS/32. These programs can then be compiled and assembled to be down-loaded into the NS32016-based system memory to be executed.

Using Dynamic RAM With Series 32000® CPUs

National Semiconductor Corp.
Application Note 405
Microprocessor Applications
Engineering

Recent advances in semiconductor technology have led to high-density, high-speed, low-cost dynamic random access memories (DRAMs), making large high-performance memory systems practical. DRAMs have complex timing and refresh requirements that can be met in different ways, depending on the size, speed, and processor interface requirements of the memory being designed. For low or intermediate performance, off-the-shelf components like the DP8419 can be used with a small amount of random logic. For higher performance, specialized high-speed circuitry must be designed

This application note presents the results of a timing analysis, and describes a DRAM interface for the NS32016 optimized for speed, simplicity and cost.

A future application note will discuss such features as error detection and correction, scrubbing, page mode and/or nibble mode support, in conjunction with future CPUs, such as the NS32332.

TIMING ANALYSIS RESULTS

Figures 1 and *2* show the number of CPU wait states required during a DRAM access cycle, for different CPU clock frequencies and DRAM access times.

Figure 1 is related to a DRAM interface using the DP8419 DRAM controller. Descriptions of the circuitry for use with the DP8419 and related timing diagrams are omitted. See the "DP8400 Memory Interface Family Applications" book for details.

Figure 2 shows the same data for a DRAM interface using standard TTL components, specially designed for the NS32016.

The special-purpose interface requires fewer wait states than the DP8419-based interface, especially at high frequencies.

These results assume a minimum amount of buffering between DRAM and CPU.

The results do not apply when CPU and DRAM reside on different circuit boards communicating through the system bus, since extra wait states may be required to provide for synchronization operations and extra levels of buffering.

INTERFACE DESCRIPTION

The DRAM interface presented here has been optimized for overall access time, while requiring moderate speed DRAMs, given the CPU clock frequency.

This may be significant when a relatively large DRAM array must be designed since a substantial saving can be achieved.

The result of these considerations has been the design of a high-speed DRAM interface capable of working with a CPU clock frequency of up to 15-MHz and 100-nsec DRAM chips, without wait states.

The only assumption has been that the DRAM array is directly accessible through the CPU local bus.

RAM Access Time in nsec	CPU Wait States Required							
250	0	1	1	1	1	2		
200	0	0	1	1	1	1	2	2
150	0	0	0	0	1	1	1	1
120	0	0	0	0	0	1	1	1
100	0	0	0	0	0	0	1	1
	6	7	8	9	10	11	12	13 CPU Clock Frequency in MHz

FIGURE 1. Memory Speed vs. CPU Wait States When Using the DP8419 DRAM Controller

RAM Access Time in nsec	CPU Wait States Required									
250	0	0	1	1	1	1				
200	0	0	0	0	1	1	1	1		
150	0	0	0	0	0	0	1	1	1	1
120	0	0	0	0	0	0	0	0	1	1
100	0	0	0	0	0	0	0	0	0	0
	6	7	8	9	10	11	12	13	14	15 CPU Clock Frequency in MHz

FIGURE 2. Memory Speed vs. CPU Wait States When Using Random Logic

This configuration presents some speed advantages; for example, the amount of buffering interposed between CPU and DRAM array is minimal. This translates into shorter propagation delays for address, data and other relevant signals.

Another advantage is that the interface can work in complete synchronization with the CPU. This significantly improves performance since no time is spent for synchronization. Reliability also improves since the possibility of metastable states in synchronizing flip-flops is eliminated.

A block diagram of the DRAM interface is shown in *Figure 3*. *Figures 4* through *7* show circuit diagrams and timing diagrams.

Interface operation details follow.

$\overline{\text{RAS}}$ AND $\overline{\text{CAS}}$ GENERATION

This is the most critical part of the entire interface circuit. To avoid wait states during a CPU read cycle, the DRAM must provide the data before the falling edge of clock phase PH12 during state T3. This requires that the $\overline{\text{RAS}}$ signal be generated early in the CPU bus cycle to meet the DRAM access time. On the other hand, the $\overline{\text{RAS}}$ signal can be asserted only after the row address is valid and the $\overline{\text{RAS}}$ precharge time from a previous CPU access or refresh cycle has elapsed.

The interface circuit shown in *Figures 4* and *5* relies on two advanced clock signals obtained from CTTL through a delay line and some standard TTL gates.

The advanced clock signals, CTTLA and $\overline{\text{CTTLB}}$, are used to clock the circuit that arbitrates between CPU access requests and refresh requests. The $\overline{\text{CTTLB}}$ signal is also used to enable an advanced $\overline{\text{RAS}}$ generation circuit, which causes the $\overline{\text{RAS}}$ signal to be asserted earlier than the CPU access-grant signal from the arbitration circuit. This speeds up the $\overline{\text{RAS}}$ signal by about 10 ns by avoiding the time required by the arbitration circuit to change state.

A different delay line is used to generate the $\overline{\text{CAS}}$ signal and to switch the multiplexers for the column addresses. Note that the $\overline{\text{CAS}}$ signal during write cycles is delayed until the beginning of CPU state T3, to guarantee that the data being written to the DRAM is valid at the time $\overline{\text{CAS}}$ is asserted. The $\overline{\text{CAS}}$ signal is deasserted after the trailing edge of $\overline{\text{RAS}}$ to guarantee the minimum pulse width requirement.

The timing diagrams in *Figures 6* and *7* show the signal sequences for both read and write cycles.

ADDRESS MULTIPLEXING

The multiplexing of the various addresses for the DRAM chips is accomplished via four 74AS153 multiplexer chips in addition to some standard TTL gates used to multiplex the top two address bits needed for 256k DRAMs. The resulting nine address lines are then buffered and sent to the DRAMs through series damping resistors. The function of these resistors is to minimize ringing.

REFRESH

The refresh circuitry includes an address counter, a timer and a number of flip-flops used to generate the refresh cycle and to latch the refresh request until the end of the refresh cycle.

The address counter is an 8-bit counter implemented by cascading the two 4-bit counters of a 74LS393 chip. This counter provides up to 256 refresh addresses and is incremented at the end of each refresh cycle.

The refresh timer is responsible for generating the refresh request signal whenever a refresh cycle is needed. This timer is implemented by cascading two 4-bit counters. Both counters are clocked by the $\overline{\text{CTTLB}}$ signal; the first is a presettable binary counter that divides the clock signal by a specified value; the second can be either a BCD or a binary counter depending on the CPU clock frequency.

With this arrangement, a refresh request is generated after a fixed time interval from the previous request, regardless of the CPU activity. A more sophisticated circuit that generates requests when the CPU is idle could also be implemented. However, such a circuit has not been considered here because the performance degradation due to the refresh is relatively small (less than 3.3 percent), and the improvement attainable by using a more sophisticated circuit would not justify the extra hardware required.

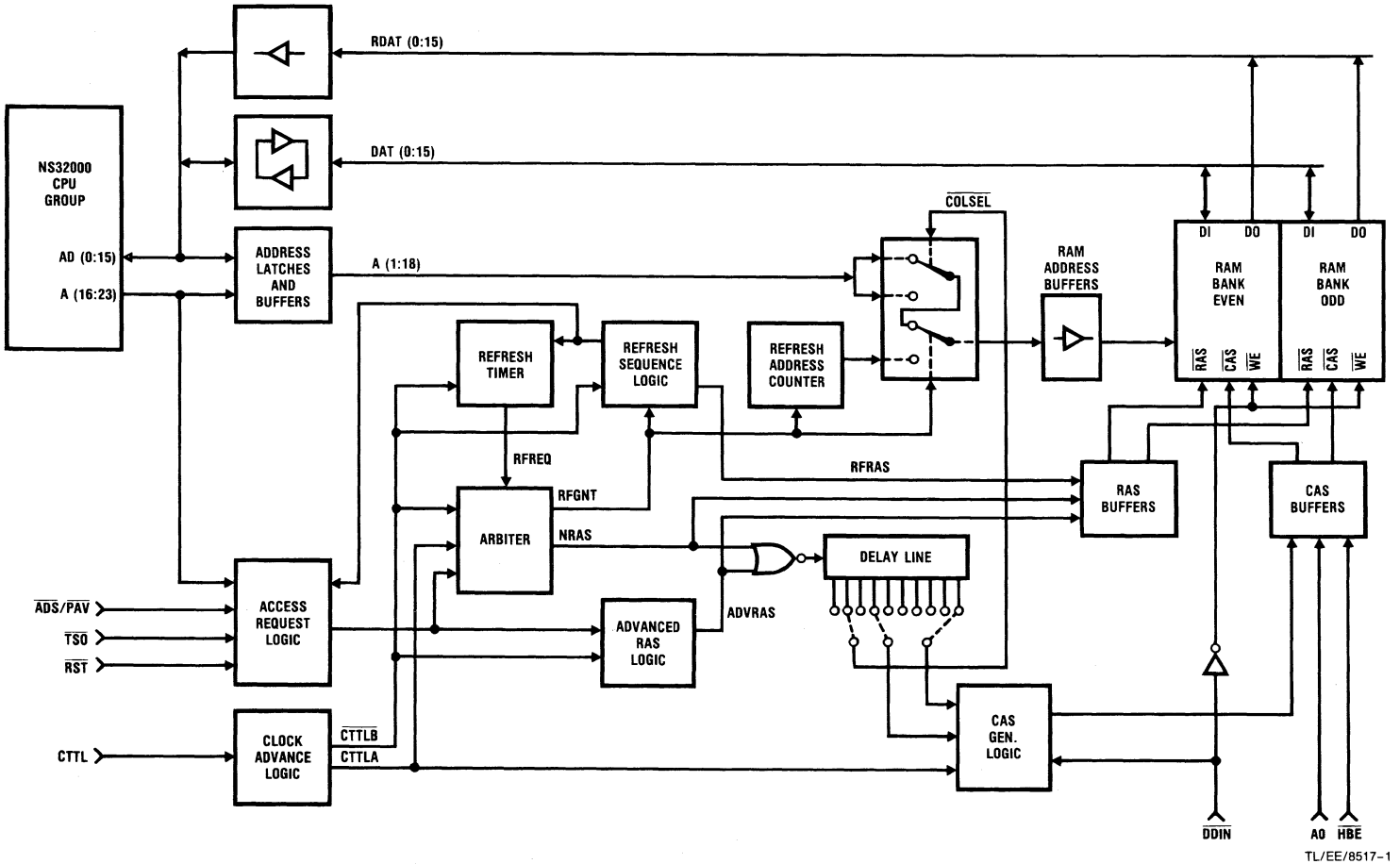
CONCLUSIONS

The DRAM interface described in this application uses two TTL-buffered delay lines to obtain speed advantages. One delay line is used to time the $\overline{\text{CAS}}$ signal and to enable the column address. The other is used to generate the advanced clock signals from CTTL.

Below 10 MHz, the advanced clocks might not be required, and the related delay line can be eliminated. When this is done, however, higher speed DRAMs must be used. If, on the other hand, advanced clocks must be used for frequencies lower than 10 MHz, a delay line with a larger delay (e.g. DDU-7J-100) might be needed.

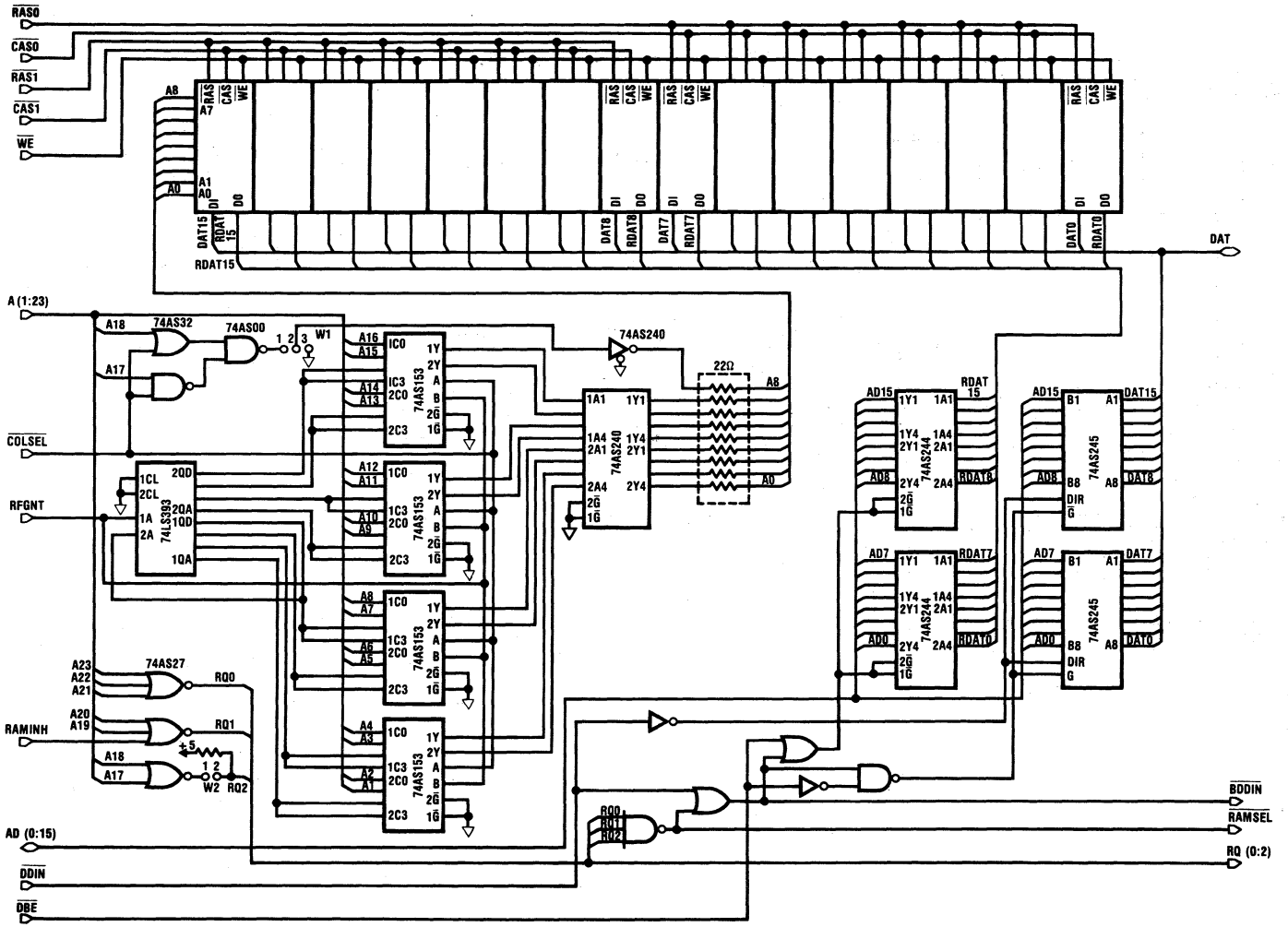
Delay lines are extremely versatile for this kind of application due to their accuracy and the fact that different delays are easily available to accommodate different DRAM types.

The savings attainable by using slower DRAM chips, in addition to the reliability improvement and cleaner design, make delay lines a valid alternative, even though their cost is relatively high in comparison to standard TTL gates.



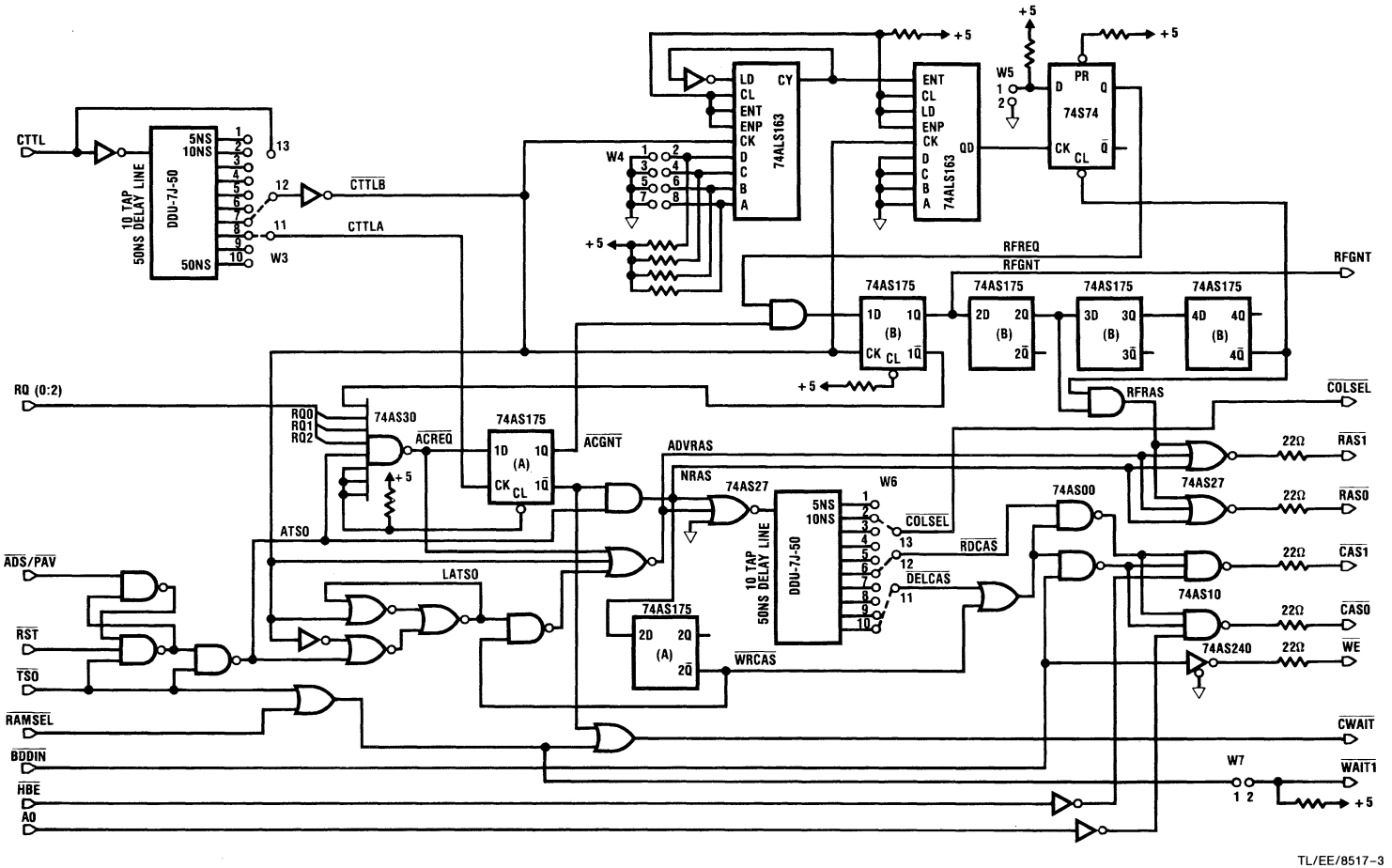
TL/EE/8517-1

FIGURE 3. DRAM Interface Block Diagram



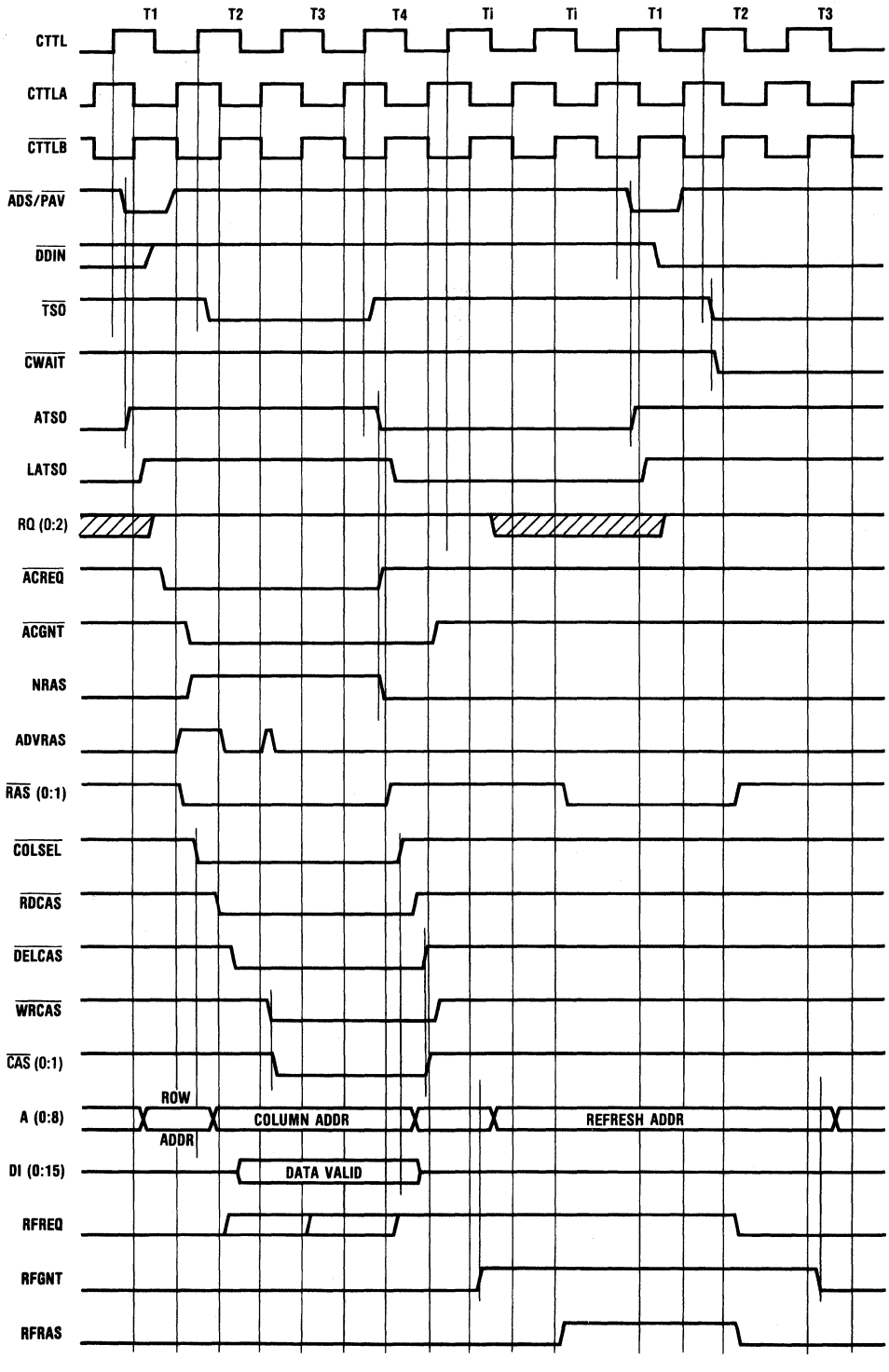
TL/EE/8517–2

FIGURE 4. DRAM Interface Circuit Diagram (a)



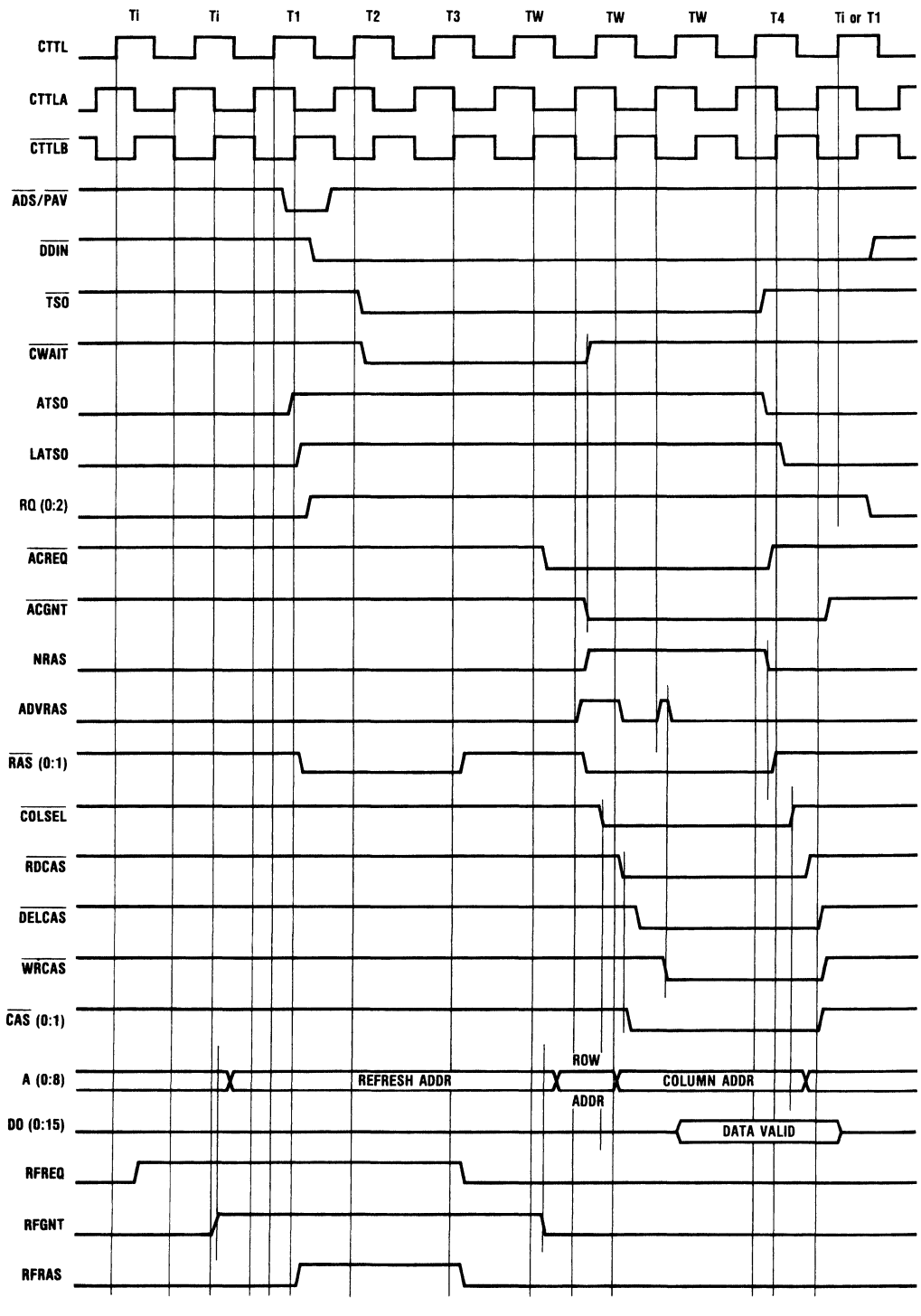
TL/EE/8517–3

FIGURE 5. DRAM Interface Circuit Diagram (b)



TL/EE/8517-4

FIGURE 6. Write Cycle Followed By a Refresh Cycle



TL/EE/8517–5

FIGURE 7. Refresh Cycle Followed By a Read Cycle

Interfacing the Series 32000® CPUs to the MULTIBUS®

National Semiconductor Corp.
Application Note 406
Microprocessor Applications
Engineering

One of the key elements in a computer system is the system bus which holds all the hardware components together. The bus contains the necessary signals to allow the hardware components to interact with each other. Memory and I/O transfers, system interrupts, etc., can all be handled by the system bus.

A variety of alternatives is available to the designer whenever an interface to a particular bus is to be designed. For example, for a low performance CPU, interface performance might not be of prime concern. In this case, the use of a standard LSI device implementing the interface functions might be recommended. If, on the other hand, a high performance CPU is being used and the performance of the interface is critical to overall system performance, a better approach might be to design special circuitry using standard TTL logic in conjunction with high speed PAL®s or gate-arrays. This application note presents an implementation of the arbitration section of a MULTIBUS interface, specially designed for the Series 32000 CPUs.

CIRCUIT DESCRIPTION

The MULTIBUS arbitration circuitry described here uses a high-speed PAL and some standard TTL logic to implement the arbiter state machine and other control functions. This solution, in addition to speed advantages, also provides a higher degree of flexibility as opposed to one which relies on LSI devices.

A detailed circuit diagram of this arbitration circuitry and the arbiter state diagram are shown in *Figures 1* and *2*.

Figure 3 shows a timing diagram of a bus acquisition and release sequence. In this case a higher priority master was in control of the bus and a lower priority master issued a bus request during the current master bus transfer cycle.

As the state diagram shows, the arbiter uses a nonrelease philosophy, called "Parking." In this scheme, once the bus is acquired, it will not be released at the end of the access cycle unless the force-release signal $\overline{\text{FREL}}$ is low or a bus request has been issued by another master. The latter condition is detected by monitoring the bus lines $\overline{\text{BPRN}}$ and $\overline{\text{CBRQ}}$. A bus release sequence is started as soon as $\overline{\text{CBRQ}}$ is asserted low or $\overline{\text{BPRN}}$ goes high. The arbiter enters state S3 and asserts the end-cycle detection enable signal $\overline{\text{ECDEC}}$ to allow appropriate logic to detect the end of the current MULTIBUS transfer cycle from the on-board CPU. When the end-cycle condition is detected, further MULTIBUS accesses from the on-board CPU are inhibited, and the signal ECYC is asserted to notify the arbiter that the bus can be released. Note that if an interlocked operation is in progress, the end-cycle condition occurs at the end of the last cycle of the interlocked operation rather than at the end of the current cycle.

This interlocked mechanism for the bus release is necessary to guarantee proper operation of the arbitration circuitry, regardless of the bus clock and CPU clock frequencies.

The nonrelease philosophy is very effective in that it can save the bus exchange overhead for the current master when no other master is requesting the bus. An increase in reliability also results, since the number of synchronization operations required for bus arbitrations is minimized.

In this application, particular care has been taken for the bus interlock. This is important, especially if a dual-port memory is used, since hardware deadlocks could result if interlocked cycles are not handled properly.

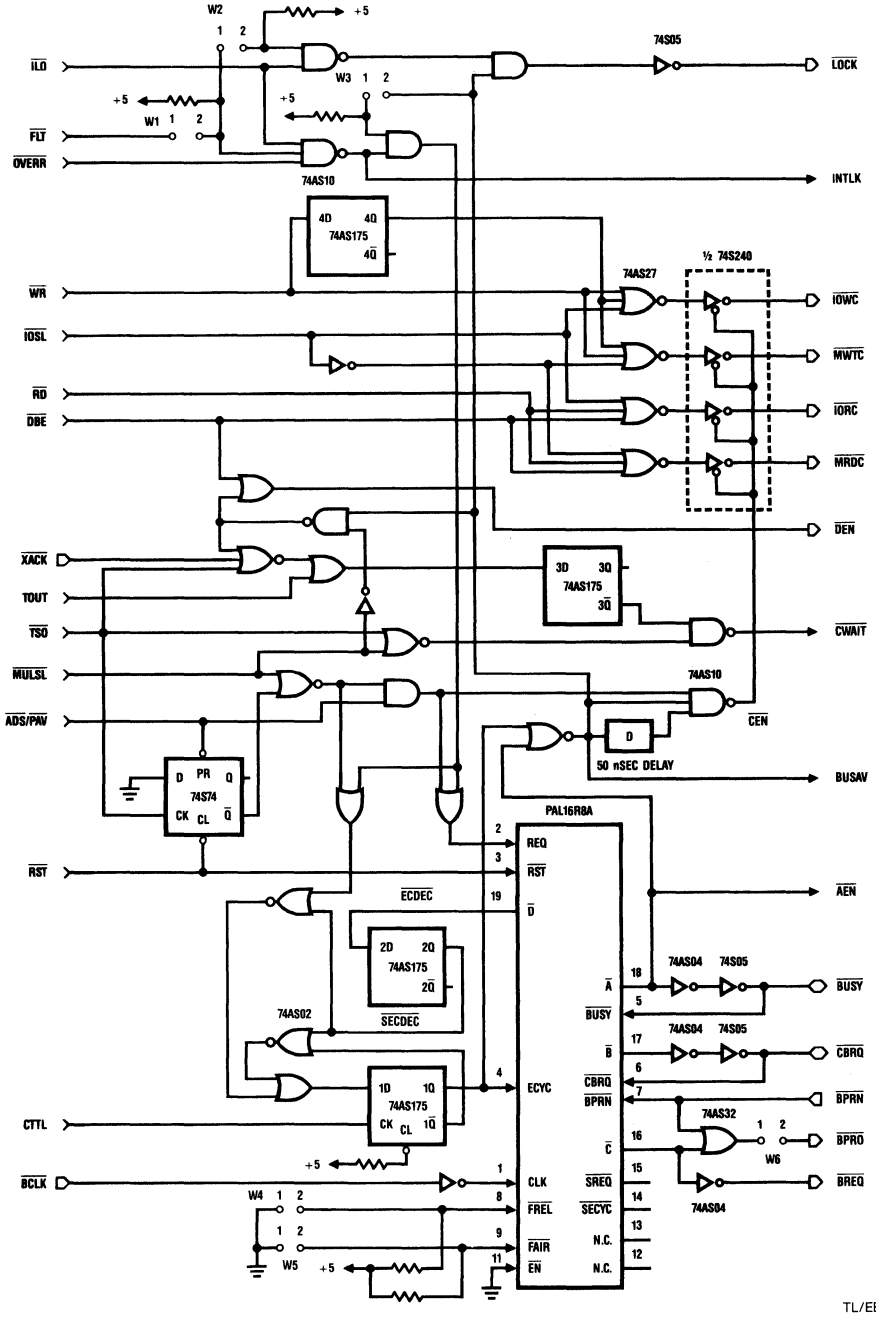
Consider the case of interlocked operations involving multiple access cycles, with the first access directed to the dual-port memory and some or all of the subsequent accesses directed to the system memory. This could happen, for example, when the MMU cycles are interlocked, the first-level page table resides in the dual-port memory, and some shared second-level page tables are kept either in the system memory or in the dual-port memory of another master. In this case, if the dual-port memory control logic grants an interlocked access to the MMU (thus setting a locking mechanism to prevent accesses from external masters until the end of the interlocked operation), and in the meantime the bus is acquired by an external master trying to access the dual-port memory, a hardware deadlock will result. This is because the dual-port memory is not unlocked until the MMU accesses the system memory to complete its operation, while the MULTIBUS is not released by the external master until the dual-port memory access is granted.

This problem can be solved by requesting the bus at the beginning of the interlocked operation, even though the interlocked cycle is not directed to the system memory, and delaying the first interlocked access to the dual-port memory until the bus has been acquired.

Another relevant point is the generation of the MULTIBUS read and write signals. These are derived from the TCU signals $\overline{\text{RD}}$ and $\overline{\text{WR}}$. The leading edges of these signals are delayed to meet the MULTIBUS timing requirements. The MULTIBUS read signals $\overline{\text{MRTC}}$ and $\overline{\text{IORC}}$ are delayed until the $\overline{\text{DBE}}$ signal becomes active, since only the address set-up time requirement must be met. A larger delay is needed instead, for the MULTIBUS write signals $\overline{\text{MWTC}}$ and $\overline{\text{IOWC}}$ to meet the data set-up time requirement. Note that the delay line serves this purpose only during the first bus access immediately following an arbitration sequence.

SIGNALS DESCRIPTION

Details on most signals used in the MULTIBUS arbitration circuitry can be found in the appropriate MULTIBUS specification manuals and Series 32000 data sheets. A description of those signals not found in these documents follows.



TL/EE/8518–1

FIGURE 1. 32000 MULTIBUS Interface Circuit Diagram

Bus Override ($\overline{\text{OVERR}}$): This signal is from an override flip-flop and can be used to hold the bus during burst transfers. The maximum duration for this signal to be asserted must not exceed the shortest time-out setting in the system. Note that the signal $\overline{\text{LOCK}}$ is not activated when the bus is overridden by the signal $\overline{\text{OVERR}}$. This is because the $\overline{\text{LOCK}}$ signal can only be asserted for a maximum of 12 microseconds, while the bus can be overridden for several milliseconds, depending upon the system requirements.

IO Select ($\overline{\text{IOSL}}$): This signal is generated by the address decoder and is asserted when the MULTIBUS I/O space is accessed.

MULTIBUS Select ($\overline{\text{MULSL}}$): $\overline{\text{MULSL}}$ is from the address decoder and is asserted when a MULTIBUS memory or I/O access is requested.

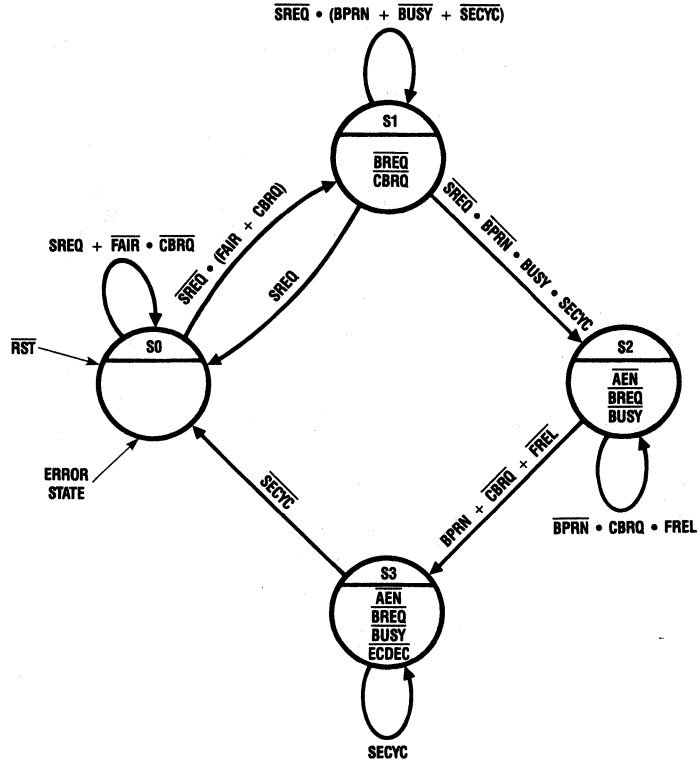
Time-Out (TOUT): This signal is generated by a fail-safe timer and is used to terminate the cycle if a bus grant, a peripheral, or memory acknowledge is not generated within a certain period of time.

Address Buffers Enable ($\overline{\text{AEN}}$): When $\overline{\text{AEN}}$ is active, the MULTIBUS address buffers are enabled.

Data Buffers Enable ($\overline{\text{DEN}}$): When $\overline{\text{DEN}}$ is active, the MULTIBUS data buffers are enabled.

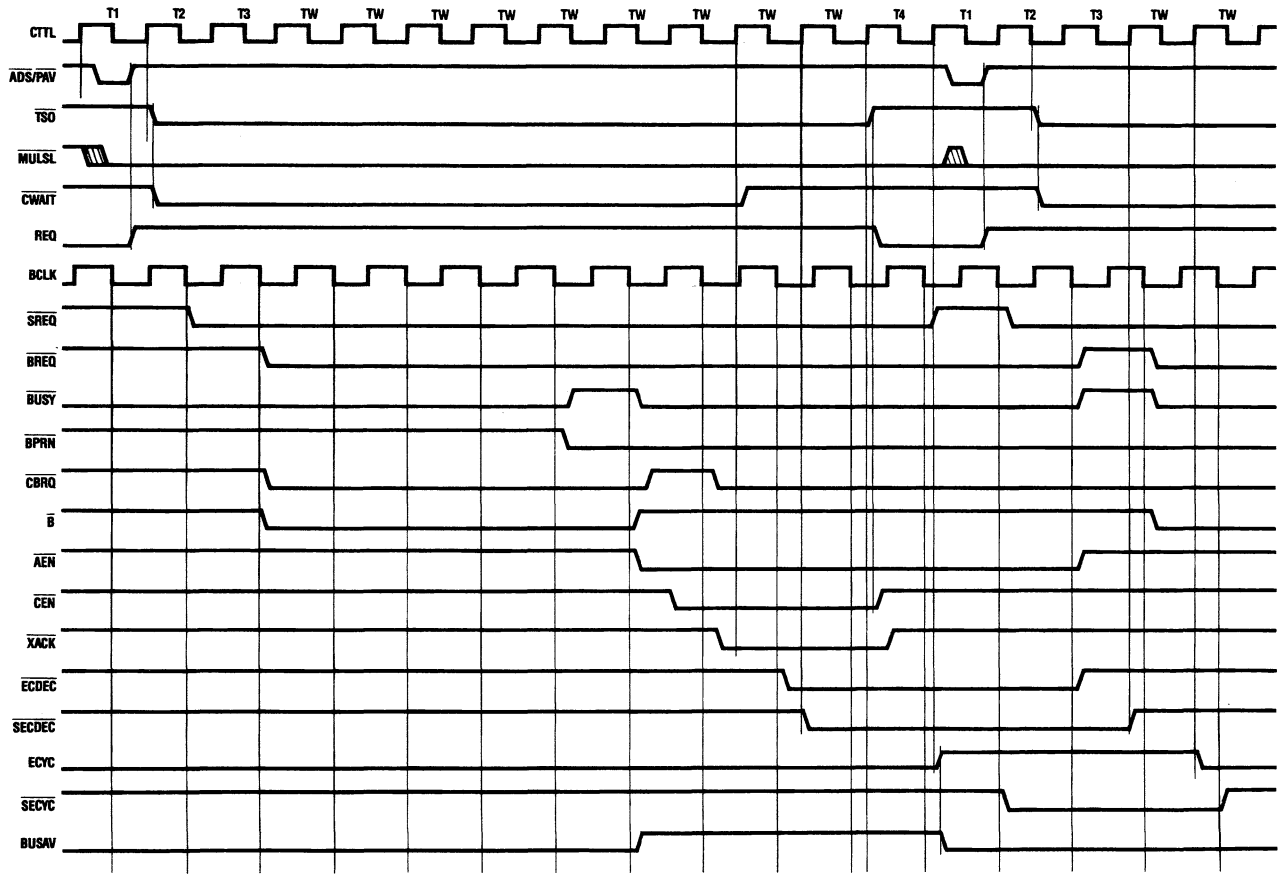
Interlocked Cycle (INTLK): Active high. This signal, when active, indicates that an interlocked cycle either is being started or is in progress. INTLK is used by the dual-port memory control logic to control interlocked accesses.

Bus Available (BUSAV): This signal is used to notify the dual-port memory control logic that the bus has been acquired and an interlocked access can be granted. This is necessary to avoid hardware deadlocks.



TL/EE/8518–2

FIGURE 2. MULTIBUS Arbiter State Diagram



TL/EE/8518–3

FIGURE 3. MULTIBUS Acquisition and Release Timing Diagram

JUMPER SETTINGS

The following explains how to use the different jumpers provided in the MULTIBUS arbitration circuitry.

W1: This jumper should be installed if MMU cycles must be interlocked. This may be necessary if some page tables must be shared by different masters.

W2: W2 should be installed if W1 is installed and some shared page tables are kept in the dual-port memory of another master.

W3: If this jumper is not installed, an interlocked access cycle generates a MULTIBUS request, even though the cycle is not directed to the MULTIBUS. This may be necessary to avoid deadlocks. This jumper can be installed if all the cycles of an interlocked operation are always directed to either the MULTIBUS or the dual-port memory. This could improve the system performance by eliminating the need to acquire the bus when an interlocked operation only accesses the dual-port memory. This jumper should not be installed when using the present versions of the Series 32000 CPUs because these CPUs can prefetch between the read and write cycles of an interlocked operation.

W4: When this jumper is installed, the bus is released at the end of each cycle. This should be avoided if all the masters in the system support $\overline{CBRQ}$, since a significant performance degradation would result. However, if some lower priority master does not support $\overline{CBRQ}$, W4 must be installed, otherwise the lower priority master (not supporting $\overline{CBRQ}$) will not be able to acquire the bus.

W5: When W5 is installed, a "fairness" arbitration mechanism is activated. In this case a new bus request is issued only when all other external requests have been serviced. This allows a serial arbitration scheme to be used with three masters (or more, if the bus clock frequency is reduced) heavily using the bus, without the risk of the bus being monopolized by the two top priority masters.

W6: W6 must be installed when a serial (daisy chain) arbitration scheme is used. It must be removed when using a parallel scheme.

Arbiter Logic Equations

$S0 := S0 * (SREQ + \overline{FAIR} * \overline{CBRQ}) + S1 * SREQ + S3 * \overline{SECYC} + \overline{RST} + ERRST$

$S1 := (S0 * \overline{SREQ} * (FAIR + CBRQ) + S1 * \overline{SREQ} * (BPRN + \overline{BUSY} + \overline{SECYC})) * RST$

$S2 := (S1 * \overline{SREQ} * \overline{BPRN} * BUSY * SECYC + S2 * \overline{BPRN} * CBRQ * FREL) * RST$

$S3 := (S2 * (BPRN + \overline{CBRQ} + \overline{FREL}) + S3 * SECYC) * RST$

State Encodings

$S0 = A \bullet B \bullet \overline{C} \bullet D$

$S1 = A \bullet \overline{B} \bullet C \bullet D$

$S2 = \overline{A} \bullet B \bullet C \bullet D$

$S3 = \overline{A} \bullet B \bullet C \bullet \overline{D}$

Error state

$ERRST = A \bullet B \bullet C \bullet D$

FIGURE 4. Arbiter Logic Equations and State Encodings

```
PAL16R8A
PART #
MULTIBUS ARBITER
NATIONAL SEMICONDUCTOR
CLK REQ RST ECYC BUSY CBRQ BPRN FREL FAIR GND
EN NC NC SECYC SREQ C B A D VCC
/A := A * /B * C * D * /SREQ * /BPRN * BUSY * SECYC * RST + /A * B * C * D * RST
   + /A * B * C * /D * SECYC * RST
/B := A * B * /C * D * /SREQ * FAIR * RST + A * B * /C * D * /SREQ * CBRQ * RST
   + A * /B * C * D * /SREQ * BPRN * RST + A * /B * C * D * /SREQ * /BUSY * RST
   + A * /B * C * D * /SREQ * /SECYC * RST
/C := A * B * /C * D * SREQ + A * B * /C * D * /FAIR * /CBRQ + A * /B * C * D * SREQ
   + /A * B * C * /D * /SECYC + /RST + A * B * C * D
/D := /A * B * C * D * BPRN * RST + /A * B * C * D * /CBRQ * RST
   + /A * B * C * D * /FREL * RST + /A * B * C * /D * SECYC * RST
/SREQ := REQ
/SECYC := ECYC
```

FIGURE 5. PAL Equations in "PALASM" Format

A Systems-Oriented Microprocessor Bus

National Semiconductor Corp.
Application Note 449

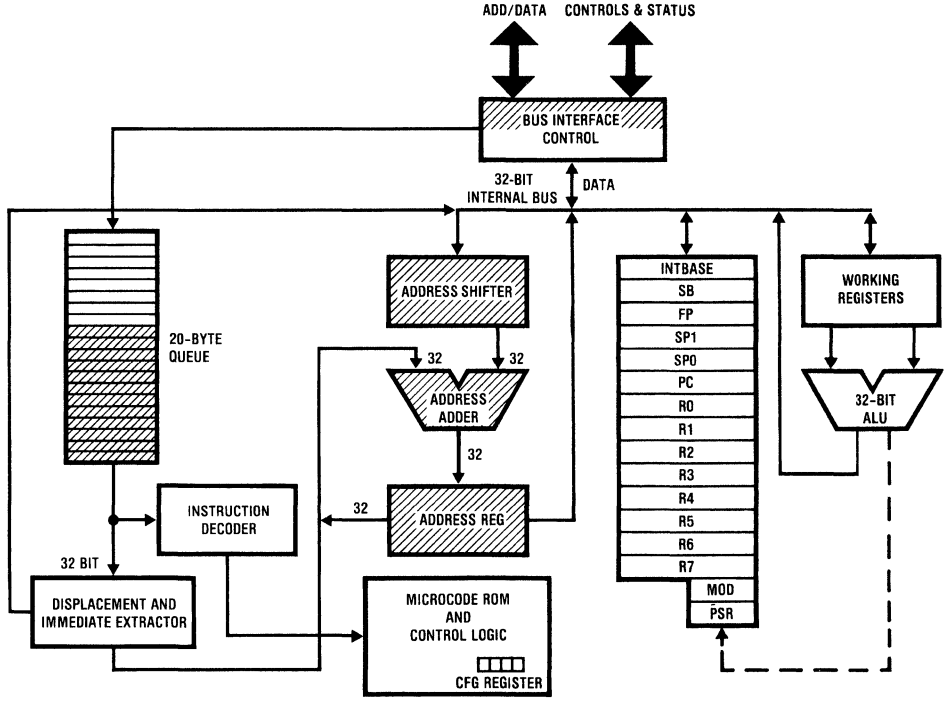
INTRODUCTION

Viewed from the standpoint of hardware throughput, the performance attained by state of the art architectural and technological implementations is a result not only of what gets done per instruction, but also of the available memory speed and bandwidth as perceived by the processor. In addition to the memory parameters, the effects of DMA, processors' contentions and the system bus itself needs to be considered. The clock frequency provided by the process and design techniques is limited by the complexity of architecture and its implementation. This mechanism limiting the performance of the "theoretical" machine running by itself; the actual in-system performance is further limited by the protocols interfacing the machine to system memory and the bandwidth of memory.

Until recently, advances in processes and architectural implementations were the main tools used to improve performance in a uniprocessor or few processors' environment, due largely to the price and size of computing clusters which were considered to be the main resources of the system and the resulting price of communications to memory and I/O. The introduction of computing clusters such as the Series 32000® family, consisting of CPU (Central Processor Unit), MMU (Memory Management Unit), FPU (Floating Point Unit), TCU (Timing Control Unit), has changed the price/performance criteria to the point where the use of multiple processors has become a viable way to leapfrog the far too slow progress in process and integration capabilities.

Given the appropriate tasks and software, a multiprocessor system can increase its total throughput by a very significant factor compared to the performance that available processes and hardware have acquired for the single processor. Furthermore, this throughput is obtainable incrementally, making the multiprocessing a more flexible and better long range investment for both OEMs and end users.

The NS32332, a new CPU in the Series 32000 family, has addressed the issues of fast system/memory bus protocol and multiprocessing support in addition to internal architectural enhancements. Together with the NS32382 MMU and NS32310 FPA, it becomes the third computing cluster to be released by NSC with a 15 MHz relative performance evaluated at 2–3 times the NS32032 cluster at 10 MHz. The size and price of Series 32000 computing clusters together with the denser packing of other components have combined to



TL/EE/8762-1

FIGURE 1. NS32332 Internal Architecture. Shading indicates enhancements to the NS32032 implementation. The principal new elements are a high-speed ALU and address shifter in the address-computation unit, an additional 12 bytes in the instruction prefetch queue (used with burst transfers), and a new high speed external interface.

provide multiprocessing performance equivalent to the performance of superminis and mainframes with the added bonus of reliability and low cost maintenance features. System architectures using private caches and global memory for example, are now in a position to contemplate delivering 10 MIPS per board due to the compact size of the new cluster, with maximum performance per box driven by the bandwidth and physical length of the system bus.

Note: The MIPS referenced are Series 32000 instruction set MIPS.

NS32332 ARCHITECTURE

The NS32332 (*Figure 1*) has been configured as a hardware microarchitectural improvement of the NS32032 CPU, using the relevant parts of its database, automatically ensuring the correctness of the programmer's architectural model. The main NS32032 referenced architectural enhancements relate to the addition of new dedicated addressing hardware (consisting of a high speed ALU, a barrel shifter and an address register), a very efficient increased (20 bytes) instruction prefetch queue, a new system/memory bus interface/protocol, increased efficiency slave processor protocol and finally enhancements of microcode.

The resulting machine is a three stage pipelined microprogrammed CPU. Compared to the NS32032, it has a larger address space (32 bits), faster address calculation, faster execution time and faster memory references causing less "contention per mips" both internally and externally with the system and other processors. The improvements have been made possible using a more advanced process which enabled the implementation of additional circuitry on the chip as well as a higher frequency, the full effect of which is ensured by the zero-waits implementation possible at 15 MHz.

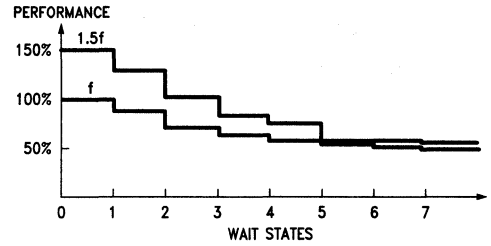
PHYSICAL DESCRIPTION

Featuring full 32 bit internal and external address/data paths, the 10.1 x 8.7 mm chip contains 80,000 transistor sites and is housed in an 84 pin PGA package. It is implemented in 2.8 micron drawn minimum feature, single metal, single poly NMOS and operates at 15 MHz from one +5V supply dissipating a worst case evaluated power of 3W. The externally generated clock delivers two non-overlapping phases and provides the required system synchronizations via a TTL compatible clock and special circuitry for multiprocessor clock synchronization.

MEMORY PARAMETERS

The efficiency of the machine/memory protocol determines the maximum frequency at which the machine will run without wait states, the machine transaction rate with memory, and the minimization of contention between processors.

Pipelined machines' performance is impacted by the introduction of wait states due to memory access time (*Figure 2*), exhibiting sometimes 25% performance loss on the first wait state with decreasing percentages for losses due to subsequent wait states. Non-optimized protocols may negate the achievements of process and circuit technologies by losing the performance advantage to inefficient system interface. For a given machine/memory protocol, the operating frequency and the number of clocks allowed for memory access will determine the required memory access time; conversely, the maximum achievable zero-wait-states frequency, is determined by the available system memory access times and the number of clocks allowed by the protocol between address generation and data setup (memory read cycle).



TL/EE/8762–2

FIGURE 2. Processor with inefficient system/memory bus. Performance, high without wait states, drops significantly when introduced in a real system. Note the marginal effect clock frequency (f) increase has in the presence of wait states.

A second parameter is the memory bandwidth required by the machine to support the transaction rate it needs to develop its throughput. Given that a relationship exists between the MIPS rate of the machine and the average number of memory accesses it needs to make per instruction (workload dependent), the available memory bandwidth is perceived to directly impact performance, memory cycle times, protocol efficiency and contention with DMA, cache operations and/or other processors will combine to determine the throughput achievable in a real system.

As will be shown later, the NS32332 cluster has met the above requirements by providing a protocol supportive of zero-waits at top frequency, requiring minimal bandwidth by using burst transactions, achieving in the process a combination of high speed access time and low memory bandwidth.

MULTIPROCESSING SUPPORT

Extending performance beyond the capabilities of both chip technology and system memory bandwidth, multiprocessing architectures can efficiently harness the power of many processors by providing the appropriate software environment and by increasing the available memory bandwidth. Multiprocessing support consists of providing the features required to support the integrity of program execution across multiple processors and the hardware implementation of memory bandwidth extension. Particularly demanding is the architecture using private caches and global memory. In addition to supporting system interlocks to provide atomic semaphore operations, the computing cluster must ideally provide efficient matching to cache speeds at the maximum cluster frequency. It must obtain zero-wait-states while still ideally providing physical cache addressing, and have low memory bandwidth to minimize the impact of cache operations.

From the point of view of cluster design, these requirements translate to:

1. Providing hardware support for indivisible CPU and MMU operations.
2. Zero-wait-states access timing has to include the MMU translation delay.
3. Reduce memory bandwidth
   - by short minimal protocol
   - interleaved memory techniques
   - pipelined arbitration
   - fast turnaround of I/O TRI-STATE®s

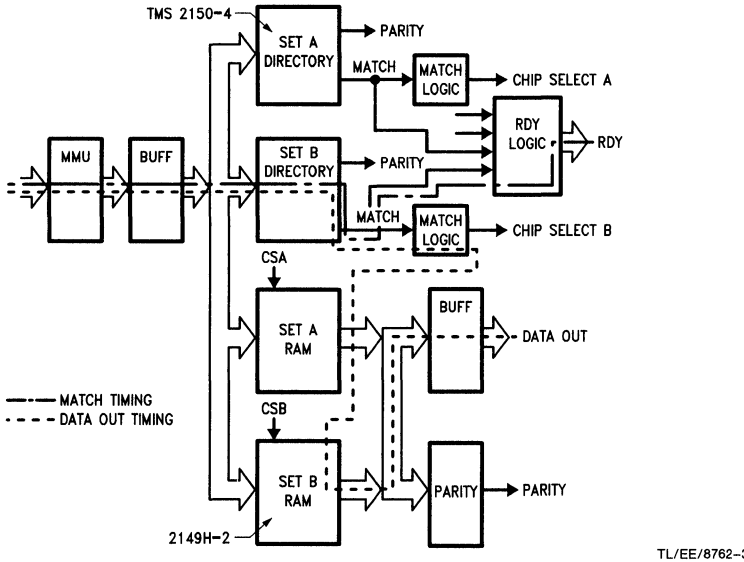


FIGURE 3. Simplified block diagram of two way set associative cache shows the match and data paths during cache read. MMU internal cache match and data cache match are overlapped. MMU translation drives directories and data RAM in a fall-through mode.

4. Efficient memory management

   high translation buffer hit ratio

   high speed MMU-system memory transactions

NS32332 PERFORMANCE AND FEATURES

Featuring 4 Gbyte uniform addressing, the 32-address-bits, 32-data-bits NS32332 is designed to provide at 15 MHz 2–3 times the throughput of the NS32032 at 10 MHz. The performance enchancement is partly due to the 1.5 frequency increase (all of which is utilizable due to zero-waits in realistic 15 MHz systems) and partly due to architectural enhancements (the new address hardware contributes 25%, the bus improvements 15%, the deeper queue 8%, the microcode enhancements contribute 2%). Performance improvement has been approached at the system level via a balanced design where the internal architecture has decreased the "clocks per instruction", the process and circuits have increased the frequency and the system/memory bus has been designed to provide the bandwidth and access time required for full in-system throughput.

Internally, less clocks per instruction are obtained by two ALUs sharing the work using dedicated buses. The multiple bus Address Unit hardware has improved performance by providing higher speed effective address computation and access to internal registers, a barrel shifter on one input provides array indexing of items sized 1, 2, 4, or 8 bytes with no additional time delay.

The deeper instruction prefetch queue, supported by the burst protocol contributes to reducing the clocks per instruction by minimizing the average instruction access time since instructions can now be fetched not only faster but also causing less contention with data. Similarly, the data path speed has been improved due to the decrease of contention with the instructions path.

The process and circuit techniques are providing a frequency of 15 MHz in a fairly simple chip (80,000 transistor sites) for its instruction set and throughput. The other chips, far simpler than the CPU, are running at the same clock frequency to provide a fully integrated computing cluster.

The NS32332 system/memory bus has been designed to provide the longest time possible for use of RAM (at zero-waits, the better part of two 66 ns clocks, is available between non-multiplexed physical address and read data required). To support the more stringent requirements of multiprocessing caches, the bus design was driven by the need to provide a physically addressed, zero-waits design, not only on the data path but also in generating the cache match signal. In *Figure 3,* the two critical paths are shown on a simplified diagram of a two-way set associative cache. The data access is less demanding as the physical address needs to propagate only via the cache data RAM. The match propagation path has to ascertain that both the internal MMU and cache directories have found a match and still drive the Ready logic with appropriate setup time to stop the transaction if a miss has to be acted on.

In cache read, for example (*Figure 4*), the critical paths of the two way set associative cache have both been supported for a physical cache design using existing components without the need of wait states. Compared to the NS32032, the major enhancements are focused on accepting the Ready signal one half clock later and a new approach for MMU design which has the translation buffer operate in fall-through mode for increased speed, overlapping virtual address propagation and physical address translation, to save the one or two waits required by other processors. The new bus protocol has reduced the clock's count to only four clocks per transaction, MMU translation included, with only two clocks dedicated to actual memory communication, thus decreasing the required memory bandwidth in circuit

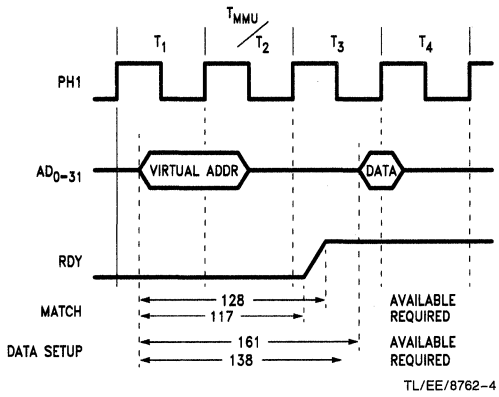


FIGURE 4. NS32332 Cache Read. The protocol at 15 MHz is designed to drive realistic cache memory without wait states.

implementations where bus switching is performed by AS buffers. Less memory bandwidth sensitivity has resulted. The impact of DMA and processor contention is minimal; with AS buffers two processors may be connected to the same memory bus.

Further reductions of access time and bandwidth requirements are achieved by the provision of a dynamically controllable burst protocol (*Figure 5*), which improves the perceived memory access time by factor 1.6 for interleaved memory systems, nibble mode or static column RAMs. In nibble mode, for example, for inexpensive implementations, the presence of one wait state will be felt only during the first access causing the whole transaction to behave as if the memory was exhibiting only 1/4 wait state. An additional benefit obtained by the burst mode is the reduction of internal contention between the instructions and data transactions. The burst mode is especially efficient for instructions prefetching, long/non-aligned operands and for bus sizes smaller than 32 bits. The burst protocol involves the use of a Burst Request line driven by the CPU and a Burst Acknowledge line driven by the memory or device addressed. Both lines are dynamically sampled to check for burst continuation. The removal of Burst Acknowledge will dynamically transform the remainder of the transaction into a renewed normal memory access. Data during the burst is moved at 2-clock intervals without regenerating the address; data rate during the burst may reach a maximum of 30 Mbytes/sec, about twice faster than comparable processors. Control of the Ready line may be employed to reduce burst speeds to match the needs of slower memories/buses.

The new bus provides enhanced fault tolerant support via dedicated Bus Retry and Bus Error lines. A dynamically variable data bus width (8, 16, or 32 bits) allows the machine to execute code of varying widths for economy, available system bus data widths or fault tolerant purposes. The burst feature makes bus width reduction be less impacting than normal accessing, for instance, only six rather than eight clocks are required to transfer a 32-bit operand over a 16-bit bus. Bus arbitration is handled at three levels depending upon the nature of the transaction; for interchip communications it is provided by the slave protocol; for chips-to-system communication, by exercising of the CPU Float line and for computing cluster-to-system accesses, by options beginning with the classical but slow Hold/Holdack protocol and ending with a pipelined arbitration scheme to take full advantage of the available memory/system bandwidth. Pipelined arbitration is further supported by early processor BIU transaction status (for determining priorities between processors) and an advanced status strobe line. Prevention of system bus deadlocks is supported by either the bus retry function or cycle hold function, both of which can regenerate the transaction. Interlocked arbitration is made possible via dedicated lines driven by the CPU and MMU chips. The local/system arbiter may then decide if and when to allow the interlocked access using if necessary the bus retry function.

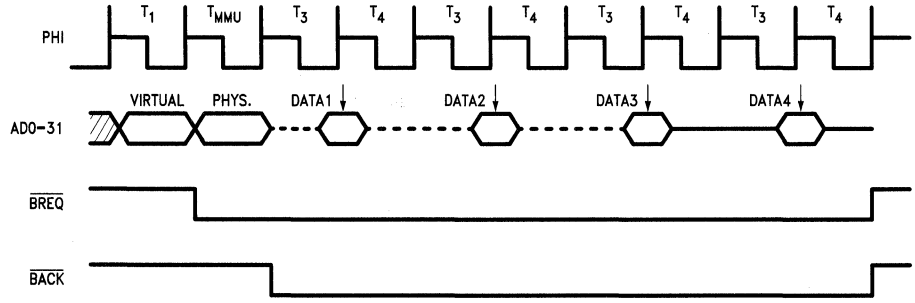


FIGURE 5. Burst transactions begin with a normal transaction followed by additional data transfers at twice the speed. The $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ lines are used to implement the handshake between memory and cluster.

The slave bus is implemented using the same a/d lines as the system/memory bus. The local-system/memory buffers are used to disconnect the local bus from the system bus during local transactions. Slave transactions are two cycles long and may use the full 32-bit bus width. Compared to the NS32032, the slave overhead has been reduced considerably by internal improvements in the instruction decode/operand move paths as well as by the new 32-bit bus.

The NS32332 protocols' complement supports all of the present and future computing cluster members to provide for easy hardware upgrading and economy of solutions. Switching slave configurations is programmed by the use of configuration registers.

SYSTEM DESIGN STRATEGY

The computing cluster implementation provides the highest, easiest to purchase throughput per square inch; the small size and price encourage its use in multiprocessor architectures. In single processor configurations, aside from performance, the computer cluster provides the least price and development risk for new products. The Series 32000 family provides continuously improving top performance computing clusters, tools, compilers and operating systems, covering the most important features of mainframes and minis. The system designer is now free to concentrate on the real software and hardware issues concerning the implementation of system architecture with his efforts spent in the directions which make his system competitive and timely. The up-down compatibility of the architecture across chip generations ensures the continued value of software investments, a benefit carried over to the end user.

SUMMARY

A first step in the direction of multiprocessing has been taken by the Series 32000 family of computing clusters. With unchanging long term architecture and multiple hardware protocols, the NS32332 cluster provides increased performance without devaluing the software investment, packing 2–3 times the performance of its predecessors in a very small, inexpensive set of chips. As increased cache sizes become possible via denser packing, multiprocessing configurations find themselves in a good position to incrementally deliver throughput previously found only in superminis or mainframes. The NS32332 cluster contributes strongly to the performance, size and price of the new machines and to their longevity.

Section 11
Appendices

Section 11 Appendices

Glossary of Terms 11-3
Programming Reference Guide 11-10
Series 32000 Quick Reference Timing 11-42
Physical Dimensions 11-63
Data Bookshelf
Sales and Distribution Offices

Glossary

In our efforts to be concise and precise, we often invent new words or acronyms to use as shorthand representations of "things" that require much longer names if the jargon is not used. Being humans, we then become very impressed with our ability to exclude those not in "the know" and another "in" group is formed. This glossary has been developed to help bridge this language gap. We know it will help. We hope you will use it.

Abort—The first step of recovery when an instruction or its operand(s) is not available in main memory. An Abort is initiated by the Memory Management Unit (MMU) and handled by the CPU.

Absolute Address—An address that is permanently assigned to a fixed location in main memory. In assembly code, a pattern of characters that identifies a fixed storage location.

Access Time—The time interval between when a request for information is made and the instant this information is available.

Access Class—The five Series 32000 access classes are memory read, memory write, memory read-modify-write, memory address, and register address. The access class informs the Series 32000 CPU how to interpret a reference to a general operand. Each instruction assigns an access class to each of it two operands, which in turn fully defines the action of any addressing mode in referencing that operand.

Accumulator—A device which stores the result of an ALU operation.

Ada—A high level language designed for the Department of Defense. It gives preference to full English words. It is meant to be the standard military language.

Address—An expression, usually numerical, which designates a specific location in a storage or memory device.

Address-Data Register—A register which may contain either address or data, sometimes referred to as a general-purpose register.

Address Strobe—Control signal used to tell external devices when the address is valid on the external address bus.

Address Translation—The process by which a logical address emanating from the CPU is transformed into a physical address to main memory. This is performed by the Memory Management Unit (MMU) in Series 32000 systems. Logical address to Physical address mapping is established by the operating system when it brings pages into main memory.

Addressing Mode—The manner in which an operand is accessed. Series 32000 CPUs have nine addressing modes: Register, Register Relative, Memory Relative, Immediate, Absolute, External, Top-of Stack, Memory Space, and Scaled Indexing.

Algorithm—A set of procedures to which a given result is obtained.

Alignment—The issue of whether an instruction must begin on a byte, double byte, or quad byte address boundary.

ALU—Arithmetic Logic Unit. A computational subsystem which performs the arithmetic and logical operations of a digital system.

Array—A structured data type consisting of a number of elements, all of the same data type, such that each data element can be individually identified by an integer index. Arrays represent a basic storage data type used in all high-level languages.

ASCII—(*American National Standard Code for Information Interchange,* 1968). This standard code uses a character set generally coded as 7-bit characters (8-bits when using parity check). Originally defined to allow human readable information to be passed to a terminal, it is used for information interchange among data processing systems, communication systems, and associated equipment. The ASCII set consists of alphabetic, numeric, and control characters. Synonymous with USASCII.

Assemble—To prepare a machine language program (also called machine code or object code) from a symbolic language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses. Machine code is a series of ones and zeros which a computer "understands".

Assembler—This program changes the programmer's source program (written in English assembly language and understandable to the programmer) to the 1's and 0's that the machine "understands". In particular, the Assembler converts assembly language to machine code. This machine code output is called the OBJECT file.

Assembly Language—A step up in the language chain. This is a set of instructions which is made up of alpha numeric characters which, with study, are understandable to the programmer. Different type of machines have different assembly languages, so the assembly language programmer must learn a different set of instructions each time s/he changes machine.

Associative Cache—A dual storage area where each data entry has an associated "tag" entry. The tags are simultaneously compared to the input value (a logical address) in the case of the MMU, and if a matching tag is found, the associated data entry is output. An associative cache is present within the MMU in Series 32000 systems to provide logical-to-physical address translation.

Asynchronous Device—A device in which the speed of operation is not related to any frequency in the system to which it is connected.

BASIC—This acronym stands for Beginner's All-purpose Symbolic Instruction Code. BASIC is one of the most "English like" of the high level languages and is usually the first programming language learned.

Baud Rate—Data transfer rate. For most serial transmission protocols, this is synonymous with bits-per-second (bps).

BCD—Binary Coded Decimal. A binary numbering system for coding decimal numbers. A 4-bit grouping provides a binary value range from 0000 to 1001, and codes the decimal digits "0" through "9". To count to 9 requires a single 4-bit grouping; to count to 99 takes two groupings of 4 bits; to count to 999 takes three groupings of 4 bits, etc.

Benchmark—In terms of computers, this refers to a software program designed to perform some task which will demonstrate the relative processing speed of one computer versus another.

Glossary (Continued)

Bit—An abbreviation of "binary digit". It is a unit of information represented by either a one or a zero.

Bit Field—A group of bits addressable as a single entity. A bit field is fully specified by the location of its least significant bit and its length in bits. In Series 32000 systems, bit fields may be from one to 32 bits in length.

Branch—A nonsequential flow in a software instruction stream.

Breakpoint—A place in a routine specified by an instruction, instruction digit, or other condition, where the software program flow will be interrupted by external intervention or by a monitor routine.

Buffer—An isolating circuit used to avoid reaction of a driven circuit on the corresponding driver circuit. Buffers also supply increased current drive capacity.

Bus—A group of conductors used for transmitting signals or power.

Bus Cycle—The time necessary to complete one transfer of information requiring the use of external address, data and control buses.

Byte—Eight bits.

Byte Enable—$\overline{BE0}$ to $\overline{BE3}$. CPU control signals which activate memory banks, each bank providing one byte of data per address.

C—A highly structured high level language developed by Bell Laboratories to optimize the size and efficiency of the program. This language has gained much popularity because it allows the programmer to get close to the hardware (low level) as well as being a high level language. Before C, the programmer who had to address the hardware had to use assembly language or machine code.

Cache—See Associative Cache.

Cache Hit—In the MMU, logical-to-physical address translation takes place via the associative cache. For this to happen, the addressed page must be resident in physical memory such that a logical address tag is present in the MMU's translation cache.

Cache Miss—When a logical address is presented to the MMU, and no physical address translation entry is found in the MMU's associative cache.

Cascaded—Stringing together of units to expand the operation of the unit. Interrupt Control Units present in a Series 32000 system which are in addition the Master ICU are referred to as "cascaded" ICUs; i.e., interrupts cascade from a second-level ICU through the master ICU to the CPU.

Clock—A device that generates a periodic signal used for synchronization.

Clock Cycle—After making a low-to-high transition, the clock will have completed one cycle when it is about to make another low-to-high transition. This time is equal to $1/f$ where f = the clock frequency.

COBOL—This acronym stands for "Common Business Oriented Language". It is a language especially good for bookkeeping and accounting.

COFF-COMMON OBJECT FILE FORMAT is a standard way of constructing files developed by AT&T for the express purpose of making all files similar. This will help reduce the situation where large files developed by one organization won't run on another organization's equipment simply because the software interfaces are different. It provides a great potential for savings in both time and money.

Compile—To take a program written in a High-Level Language such as C, Pascal, or FORTRAN and convert it into an object-code format which can be loaded into a computer's main memory. During compilation, symbolic HLL statements, called source code, are converted into one or more machine instructions which the CPU "understands". A compiler also calls the assemble function.

Compiler—The program that converts from Source to Machine Code. The conversion is from a particular high level language to machine code. For example, the C compiler will convert a C source program written by a programmer to machine code. This machine code output is in the same format as that of the assembler and is also called an OBJECT file.

CPU—Central Processing Unit. The portion of a computer system that contains the arithmetic logic unit, register file, and other control oriented subsystems. It performs arithmetic operations, controls instruction processing, and provides timing signals and other housekeeping operations.

Cross Support—The alternative to using a "Native" development like SYS32 to develop your programs is to use Cross Support software. "Native" means that the CPU in the development system is the same as the CPU in the system being developed. Cross support software is all of the necessary programs for development that operate on one CPU, but generate code for another CPU. Use of the VAX to generate Series 32000 code is a good example of cross support.

Demand-Paged Virtual Memory—A virtual memory method in which memory is divided into blocks of equal size which are referred to as pages. These pages are then moved back and forth between main memory and secondary storage as required by the CPU. Demand paging reduces the problem of memory fragmentation which results in unused memory space.

Dispatch Table—In Series 32000 systems, this is an area of memory which contains interrupt descriptors for all possible hardware interrupts and software traps. The interrupt descriptor directs the CPU to the module descriptor for the procedure which is designed to handle that particular interrupt.

Displacement—A numerical offset from a known point of reference. Displacements are used in programming to facilitate position independent code, such that a given program can be loaded anywhere in memory. In Series 32000 processors, a displacement is contained in the instruction itself.

Glossary (Continued)

DMA—Direct Memory Access. A method that uses a small processor (DMA Controller) whose sole task is that of controlling input-output or data movement. With DMA, data is moved into or out of the system without CPU intervention once the DMA controller has been initialized by the CPU and activated.

Double-Precision—With reference to 32000 floating-point arithmetic, a double-precision number has a 52-bit fraction field, 11-bit exponent field and a sign bit (64-bits total).

Double Word—Two words, i.e., 32 bits.

Editor—A program which allows a person to write and modify text. This program can be as complicated as the situation requires, from the very simple line editor to the most complicated word processor. Letters, numbers and unprintable control characters are stored in memory so that they can be recalled for modification or printing. The programmer uses this device to enter the program into the computer. At this stage, the program is recognizable to both the programmer and the computer as lines of English text. This English version of the program is known as the SOURCE.

Emulate—To imitate one system with another, such that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system.

Exception—An occurrence which must be resolved through CPU intervention. An exception results in the suspension of normal program flow. In Series 32000 systems, exceptions occur as a result of a hardware reset, interrupt or software traps. Execution of floating-point instructions may also result in occurrences which must be resolved through CPU intervention.

Exponent—In scientific notation, a numeral that indicates the power to which the base is raised.

EXEC2—NSC's Real Time Executive for Series 32000.

FIFO—First-in first-out. A FIFO device is one from which data can be read out only in the same order as it was entered, but not necessarily at the same rate.

Floating-Point—A method by which computers deal with numbers having a fractional component. In general, it pertains to a system in which the location of the decimal/binary point does not remain fixed with respect to one end of numerical expressions, but is regularly recalculated. The location of the point is usually given by expressing a power of the base.

FORTRAN—A high level language written for the scientific community. It makes heavy use of algebraic expressions and arithmetic statements.

FP—Frame Pointer. CPU register which points to a dynamically allocated data area created at the beginning of a procedure by the ENTER instruction.

FPU—Floating-Point Unit is a slave processor in Series 32000 systems which implements in hardware all calculations needed to support floating-point arithmetic, which otherwise would have to be implemented in software. The NS32081 FPU provides high-speed floating point instructions for single (32-bit) and double (64-bit) precision. Supports IEEE standard for binary floating point arithmetic. Compatible with NS32032, NS32C032, NS32016, NS32C016 and NS32008 CPUs.

Fragmented—The term used to describe the presence of small, unused blocks of memory. The problem is especially common in segmented memory systems, and results in inefficient use of memory storage.

Frame—A block of memory on the stack that provides local storage for parameters in the current procedure.

GENIX—The NSC version of the UNIX operating system, ported to work with the Series 32000. It also has all of the necessary utilities added so that program development can be accomplished.

Hardware—Physical equipment, e.g., mechanical, magnetic, electrical, or electronic devices, as opposed to the software programs or method in which the hardware is used.

High Level Languages—These are languages which are not dependent on the type of computer on which they run. A program written in a high level language will generally run on any computer for which there is a compiler for that language. This feature makes high level languages "Portable", i.e., the same program will run on many different types of computers. A HLL requires a compiler or interpreter that translates each HLL statement into a series of machine language instructions for a particular machine.

ICU—Interrupt Control Unit. A memory-mapped microprocessor support chip in Series 32000 systems which handles external interrupts as well as additional software traps. The ICU provides a vector to the CPU to identify the servicing software procedure.

Indexing—In computers, a method of address modification that is by means of index registers.

Index Register—A register whose contents may be added to or subtracted from the operand address.

Indirect Addressing—Programming method where the initial address is the storage location of a word which is the actual address. This indirect address is the location of the data to be operated upon.

Instruction—A statement that specifies an operation and the values or locations of its operands, i.e., it tells the CPU what to do and to what.

Instruction Cycle—The period of time during which a programmed system executes a particular instruction.

Instruction Fetch—The action of accessing the next instruction from memory, often overlapped by its partial execution.

Instruction Queue—With Series 32000 CPUs, this is a small area of RAM organized as a FIFO buffer which stores prefetched instructions until the CPU is ready to execute them.

Interpreter—A program which translates HLL statements into machine instructions at run time, i.e., while the program is executing, and is co-resident with the user program.

Glossary (Continued)

Interrupt—To signal the CPU to stop a software program in such a way that it can be resumed and branch to another section of code. Interrupts can be caused by events external or internal to the CPU, and by either software or hardware.

INTBASE—Interrupt Base Register. In the Series 32000, a 32-bit CPU register which holds the address of the dispatch table containing addresses for interrupts and traps.

ISE—In-System Emulator. A computer system which imitates the operation of another in terms of software execution. In microprocessor system development, the ISE takes the place of the microprocessor by means of a connector at the end of an umbilical cable. Not only does the ISE perform all the functions of the microprocessor, but it also allows the engineer to debug his system by setting breakpoints on various conditions, permits tracing of program flow, and provides substitution memory which may be used in place of actual target system memory.

ISV—Independent Software Vendor. A vendor, independent from National Semiconductor, who ports or develops software for Series 32000 components. They in turn sell this software to our customers who are designing Series 32000 based products.

Kernel—This is the name given to the core of the operating system. Other programs are added to the kernel to provide the features of the operating system. The kernel provides control and synchronization.

Language—A set of characters and symbols and the rules for using them. In our context, it is the "English like" format of the instructions which are understood by both the programmer and the computer.

Library—High level languages as well as assembly language contain many routines which are used over and over again. To prevent the programmer from having to write the routine every time it is needed, these routines are stored in libraries to be referenced each time they are needed. These libraries are also OBJECT files.

Linear Address Space—An address space where addresses start at location zero and proceed in a linear fashion (i.e., with no holes or breaks) to the upper limit imposed by the total number of bits in a logical address.

Link Base—In the Series 32000, Module Descriptor entry which points to a table in memory containing entries which reference variables or entry points in Modules external to the one presently executing.

Linker—Large programs are generally broken down to component parts and farmed out to several programmers. Each one of these parts is called a MODULE. Each programmer will develop the module using either high level or assembly language, then "assemble" assembly language modules or "compile" high level language modules. A programmer tells the linker how to connect these modules to make the program run. The linker makes these connections, resolves all questions about data needed by one module, but contained in another, finds all library routines, and cleans up any other loose ends. The output from the linker is called BINARY file and is the file that will run on the computer.

Logical Address Space—The range of addresses which a programmer can assign in a software program. This range is determined by the length of the computer's address registers.

LSB—Least Significant Bit. The bit in a string of bits representing the lowest value.

Machine Code—The code that a computer recognizes. Specifies internal register files and operations that directly control the computer's internal hardware.

Machine Language—The ones and zeros which are "understood" by the machine. This is often called "Binary Code." The programmer must be able to understand the bit patterns to be able to decipher the language. Each machine has a unique machine language.

Main Memory—The program and data storage area in a computer system which is physically addressed by the microprocessor or MMU address lines.

Mantissa—In a floating-point number, this is the fractional component.

Mapping—The process whereby the operating system assigns physical addresses in main memory to the logical addresses assigned by the software.

Memory-Mapped—Referring to peripheral hardware devices which are addressed as if they were part of the computer's memory space. They are accessed in the same manner as main memory, i.e., through memory read/write operations.

Microcode—A sequence of primitive instructions that control the internal hardware of a computer. Their execution is initiated by the decoding of a software instruction. Microcode is maintained in special storage and often used in place of hardwired logic.

Microcomputer—A computer system whose Central Processing Unit is a Microprocessor. Generally refers to a board-level product.

Minicomputer—A "box-level" computer with system capabilities generally between that of a microcomputer and a mainframe.

MMU—Memory Management Unit. This is a slave processor in Series 32000 which aids in the implementation of demand-paged virtual memory. It provides logical to physical address translation and initiates an instruction abort to the CPU when a desired memory location is not in main memory.

MOD—Mod Register. In the Series 32000, a 16-bit CPU register which holds the address of the Module Descriptor of the currently executing software module.

Module—An independent subprogram that performs a specific function and is usually part of a task, i.e., part of a larger program.

Module Descriptor—In the Series 32000, a set of four 32-bit entries found in main memory. Three are currently defined and point to the static data area, link table, and first instruction of the module it describes. The fourth is reserved.

Glossary (Continued)

Modularity—A software concept which provides a means of overcoming natural human limitations for dealing with programming complexity by specifying the subdivision of large and complex programming tasks into smaller and simpler subprograms, or modules, each of which performs some well-defined portion of the complete processing task.

MSB—Most Significant Bit. The bit in a string of bits representing the highest value.

NET—Short for NETWORK and describes a number of computers connected to each other via phone or high speed links. A net is convenient for exchanging common information in the form of "mail" as well as for data exchange.

NMI—Nonmaskable Interrupt. A hardware interrupt which cannot be disabled by software. It is generally the highest priority interrupt.

Object Code—Output from a compiler or assembler which is itself executable machine code (or is suitable for processing to produce executable machine code).

Operand—In a computer, a datum which is processed by the CPU. It is referenced by the address part of an instruction.

Operating System—A collection of integrated service routines used by the computer to control the sequence of programs. The operating system consists of software which controls the execution of computer programs and which may provide storage assignment, input/output control, scheduling, data management, accounting, debugging, editing, and related services. Their sophistication varies from small monitor systems, like those used on boards, to the large, complex systems used on main frames.

Operating System Mode—In this mode, the CPU can execute all instructions in the instruction set, access all bits in the Processor Status Register, and access any memory location available to the processor.

Operator—In the description of an instruction, it is the action to be performed on operands.

Page Fault—A hardware generated trap used to tell the operating system to bring the missing page in from secondary storage.

Page Swap—The exchange of a page of software in secondary storage with another page located in main memory. The operating system supervises this operation, which is executed by the CPU and involves external devices such as disk and DMA controllers.

Page Table—A 1K-byte area in main memory containing 256 entries which describe the location and attributes of all pointer tables, i.e., a list of pointer table addresses.

Peripheral—A device which is part of the computer system and operates under the supervision of the CPU. Peripheral devices are often physically separated from the CPU.

Pascal—A high level language designed originally to teach structured programming. It has become popular in the software community and has been expanded to be a versatile language in industry.

Physical Address—The address presented to main memory, either by the CPU or MMU.

Pointer Table—A 512-byte page located either in main memory or secondary storage containing 128 entries. Each entry describes an individual page of the software program. Each page of the software program may reside in main memory or in secondary storage.

Pop—To read a datum from the top of a stack.

PORT—To port an operating system is to cause that particular operating system to operate with a defined hardware package. GENIX is the NSC version of UNIX which has been ported to SYS32. The operating system for other Series 32000 based systems will differ in some degree from SYS32 and the NSC GENIX binary will not operate. It is now necessary to modify GENIX to fit the situation caused by the new hardware. The GENIX SOURCE is used because this is the program that is most readily understood by the programmer. The source is changed, compiled, and linked to get a new binary for that particular machine.

Primitive Data Type—A data type which can be directly manipulated by the hardware. With Series 32000, these are integers, floating-point numbers, Booleans, BCD digits, and bit fields.

Procedure—A subprogram which performs a particular function required by a module, i.e., by a larger program; an ordered set of instructions that have a general or frequent use.

Process—A task.

Program Base—Module Descriptor entry which points to the first instruction in the module being described.

Program Counter—CPU register which specifies the logical address of the currently executing instruction.

Protection—The process of restricting a software program's access to certain portions of memory using hardware mechanisms. Typically done at the operating system and page level.

PSR—Processor Status Register. A 16-bit register on Series 32000 CPU's which contains bits used by the software to make decisions and determine program flow.

Push—to write a datum to the top of a stack.

Quad word—Four words, i.e., 64 bits.

Queue—A First-In-First-Out data storage area, in which the data may be removed at a rate different from that at which it was stored.

Real Time—The actual time in human terms, related to a process. In a UNIX system, real time is total elapsed time, CPU time is the percent of time a process is actually in the CPU. Sys time is the time spent in system mode, and user time is the time spent in user mode.

Glossary (Continued)

Real Time Operating Systems—An operating system which operates with a known and predictable response time limit, so that it can control a physical event.

Record—A structured data type with multiple elements, each of which may be of a different data type, e.g., strings, arrays, bytes, etc.

Register—A temporary storage location, usually in the CPU, which holds digital data.

Relative Address—The number that specifies the difference between the base address and the absolute address.

Relocatable—In reference to software programs, this is code which can be loaded into any location in main memory without affecting the operation of the program.

Return Address—The address to which a subroutine call, interrupt or trap subroutine will return after it is finished executing.

Routine—A procedure.

Royalty—Royalty is money paid to the inventor for each item of product sold. A good analogy to use is the music business. Any time a song is used, the songwriter is paid a royalty. Think of UNIX as a song and GENIX or SYSTEM V as special arrangements. For each shipment of GENIX or SYSTEM V, the customer pays a royalty to NSC who, in turn, pays a royalty to AT&T.

SB—In the Series 32000 Static Base Register. Points to the start of the static data area for the currently executing module.

Secondary Storage—This is generally slow-access, nonvolatile memory such as a hard-disk which is used to store the pages of software programs not currently needed by the CPU.

Segmented Address Space—Term used to describe the division of allocatable memory space into blocks of segments of variable size.

Setup Time—The minimum amount of time that data must be present at an input to ensure data acceptance when the device is clocked.

Slave Processor—A processor which cooperates with the main microprocessor in executing certain instructions from the instruction stream. A slave processor generally accelerates certain functions which increases overall system throughput. Examples of slave processors are the FPU and MMU of Series 32000.

Software—Programs or data structures that execute instructions or cause instructions to be executed and that will cause the computer to do work.

Software License—NSC does not sell software. Rather, we license the right to use our software. A software license is required for all Series 32000 software. We use the license to protect NSC's interests and to assist in honoring our commitment to AT&T. The license is also the vehicle which we use to track customers so that updates can be issued in a timely manner.

Software Q/A—It is the charter of the Quality Assurance people to ensure that when a software product reaches the customer that it is "bug" free. In the real world, it is impossible to test every combination of functions, so some bugs do get through. The Q/A engineer develops test programs which rigorously test the product prior to its introduction to the market place.

SP1—In the Series 32000, User Stack Pointer. Points to the top of the User Stack and is selected for all stack operations while in User Mode.

SP0—In the Series 32000, Interrupt Stack Pointer. Points to the top of the interrupt stack. It is used by the operating system whenever an interrupt or trap occurs.

Stack—A one-dimensional data structure in which values are entered and removed one datum at a time from a location called the Top-of-Stack. To the programmer, it appears as a block of memory and a variable called the Stack Pointer (which points to the top of the stack).

Stack Pointer—CPU register which points to the top of a stack.

Static Base Register—A 32-bit CPU register which points to the beginning of the static data area for the currently executing module.

String—An array of integers, all of the same length. The integers may be bytes, words, or double words. The integers may be interpreted in various ways (see ASCII).

Subroutine—A self-contained program which is part of a procedure.

Symmetry—A computer architecture is said to be symmetrical when any instruction can specify any operand length (byte, word or double word) and make use of any address-data register or memory location while using any addressing mode.

Synchronous—Refers to two or more things made to happen in a system at the same time, by means of a common clock signal.

Tag—A label appended to some data entry used in a look-up process whereby the desired datum can be identified by its tag.

Task—The highest-level subdivision of a user software program. The largest program entity that a computer's hardware directly deals with.

TCU—Timing Control Unit. A device used to provide system clocks, bus control signals and bus cycle extension capability for Series 32000.

Trap—An internally generated interrupt request caused as a direct and immediate result of the encounter of an event.

T-State—One clock period. If the system clock frequency is 10 MHz, one T-State will take 100 ns to complete. Operations internal and external to the CPU are synchronized to the beginning and middle of the T-States. There are four T-States in a normal Series 32000 CPU bus cycle.

Glossary (Continued)

UNIX™—An operating system developed at Bell Laboratories in the early 1970s. Software programs that run under UNIX are written in the high-level language C, making them highly portable. UNIX systems do not distinguish user programs from operating system programs in either capability or usage, and they allow users to route the output of one program directly into the input of another. This operating is unique and is becoming very popular in the microcomputer world.

USENET—A net to which UNIX systems in the United States connect. Some systems in Europe and Australia also use this net for the purpose of passing information.

User—A software program. The total set of tasks (instructions) that accomplish a desired result. Tasks are managed by the operating system.

User Mode—Machine state in which the executing procedure has limited use of the instruction set and limited access to memory and the PSR.

uucp—Software which allows UNIX computers to pass information to other UNIX systems.

Variable—A parameter that can assume any of a given set of values.

Vector—Byte provided by the ICU (Interrupt Control Unit) which tells the CPU where within the Descriptor table the descriptor is located for the interrupt it has just requested.

Virtual Address—Address generated by the user to the available address space which is translated by the computer and operating system to a physical address of available memory.

Virtual Memory—The storage space that may be regarded as addressable main storage by the system. The operating system maps Virtual addresses into physical (main memory) addresses. The size of virtual memory is limited by the method of memory management employed and by the amount of secondary storage available, not by the actual number of main storage locations, so that the user does not have to worry about real memory size or allocation.

VMS—This is the operating system designed by Digital Equipment Corporation for their VAX series of computers. The original Series 32000 software was developed on a VAX which was being controlled by the VMS Operating System.

Wait-State—An additional clock period added to a CPU memory cycle which gives an external memory device additional time to provide the CPU with data. Also used by bus arbitration circuitry to hold the CPU in an idle state until access to a shared resource is gained.

Winchester—Small, hard-disk media commonly found in personal computers.

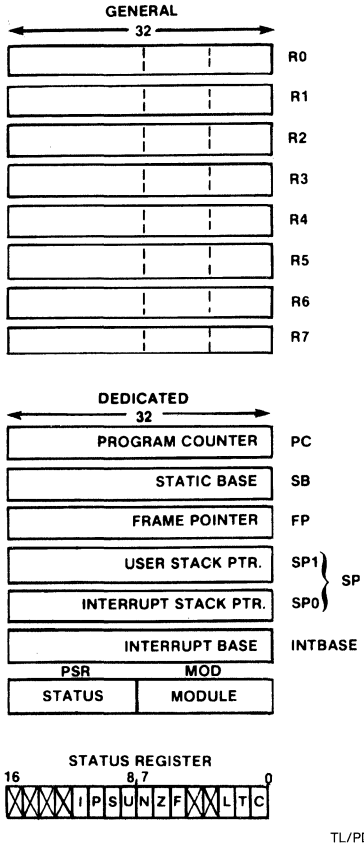
Word—A character string or bit string considered as the primary data entity. For historical reasons, a word is a group of 16 bits in Series 32000 systems.

Series 32000 Programming Reference Guide

Series 32000 family of microprocessors were carefully designed and optimized to operate in environments which demand large scale computing capabilities but microprocessor size and price.

This section does not explicitly describe the use of the NS32008 CPU. It is instruction set compatible with the CPUs described in these sections. The only differences relate to the external data bus width, the instruction execution timing and the fact that the NS32008 will not function with NS32082 MMU.

NS32332 Central Processing Unit Programming Model



TL/PD/1K17-1

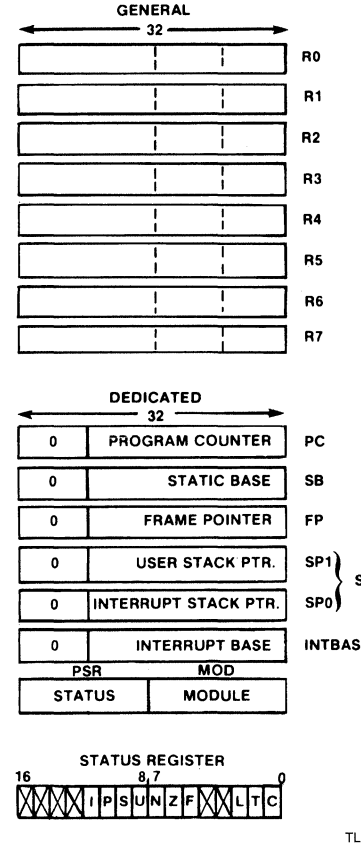
USER FLAG

C: Carry Flag
T: Trace Flag
L: Low Flag
F: General Condition Flag
Z: Zero Flag
N: Negative Flag

SUPERVISOR FLAGS

U: User Mode Flag
S: Stack Flag
P: Trace Trap Pending Flag
I: Interrupt Enable Flag

NS32032/NS32016 Central Processing Unit Programming Model



TL/PD/1K17-2

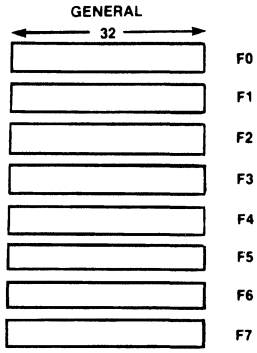
USER FLAG

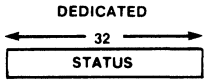
C: Carry Flag
T: Trace Flag
L: Low Flag
F: General Condition Flag
Z: Zero Flag
N: Negative Flag

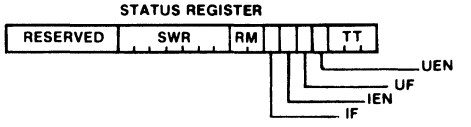
SUPERVISOR FLAGS

U: User Mode Flag
S: Stack Flag
P: Trace Trap Pending Flag
I: Interrupt Enable Flag

NS32081 Floating Point Unit Programming Model





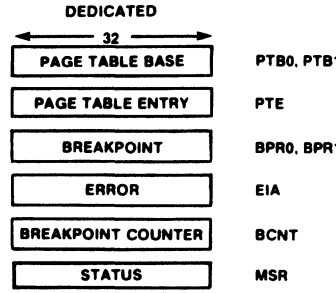


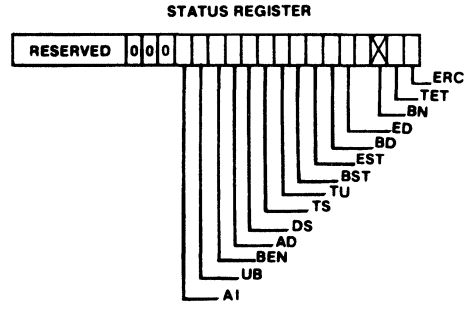
TL/PD/1K17-3

TT: Trap Type
- 000 No Trap Request
- 001 Underflow
- 010 Overflow
- 011 Division by Zero
- 100 Illegal Instruction
- 101 Invalid Operation
- 110 Inexact Result
- 111 (Reserved for future use.)

UEN: Underflow Trap Enable
UF: Underflow Flag
IEN: Inexact Result Trap Enable
IF: Inexact Result Flag
RM: Rounding Mode
- 00 Round to nearest value
- 01 Round toward zero
- 10 Round toward positive infinity
- 11 Round toward negative infinity

NS32082 Memory Management Unit Programming Model





TL/PD/1K17-4

ERC: Error Class Flag
TET: Translation Error Trace Flag
BN: Breakpoint Number Bit
ED: Error Data Direction Bit
BD: Breakpoint Direction Bit
EST: Error Status Flag
BST: Breakpoint Status Flag
TU: Translate User Flag
TS: Translate Supervisor Flag
DS: Dual Space Bit
AO: Access Override Bit
BEN: Breakpoint Enable Bit
UB: User Break Bit
AI: Abort or Interrupt Bit

Memory Management Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
LMR	*mmureg,* short mmureg	*src* read.D	Load MMU Register *mmureg:* = *src*	—	UND* ILL**	14	0010
SMR	*mmureg,* short mmureg	*dest* write.D	Store MMU Register *dest:* = *mmureg*	—	UND* ILL**	14	0011
RDVAL	*src* addr		Validate Address for Reading § If ADDRESS*(src)* in User mode may be read, F: = 0 else F: = 1	F	UND* ILL** ABT***	14	0000
WRVAL	*dest* addr		Validate Address for Writing § If ADDRESS*(dest)* in User mode may be written to, then F: = 0 else F: = 1	F	UND* ILL** ABT***	14	0001
MOVSU*i*	*src,* addr	*dest* addr	Move Value from Supervisor to User Space § *dest:* = *src* (*src* is in supervisor space; *dest* in user space)	—	UND* ILL**	8	110 reg = 001
MOVUS*i*	*src,* addr	*dest* addr	Move Value from User to Supervisor Space § *dest:* = *src* (*src* is in user space; *dest* in supervisor space)	—	UND* ILL**	8	110 reg = 011

* TRAP (UND) if M bit in CFG is 0.

** TRAP (ILL) if U flag in PSR is 1.

***TRAP (ABT) if level 1 page table address invalid.

Floating-Point Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
MOV*f*	*src*, read.*f*	*dest* write.*f*	Move Floating-Point *dest: = src*	—	UND*	11	0001
MOVLF	*src*, read.L	*dest* write.F	Move Long Floating to Floating *dest.F: = src.L*	— FSR:TT UF IF	UND* FPU	9	010
MOVFL	*src*, read.F	*dest* write.L	Move Floating to Long Floating *dest.L: = src.F*	— FSR:TT	UND* FPU	9	011
MOV*if*	*src*, read.*i*	*dest* write.*f*	Move Integer to Floating-Point *dest.i = src.f*	— FSR:TT IF	UND* FPU	9	000
ROUND*fi*	*src*, read.*f*	*dest* write.*i*	Round Floating-Point to Integer (round to even) *dest.i: = src.f* If overflow, then TRAP(FPU) (*src.f* rounded to nearest integer, or to nearest even integer if a tie)	— FSR:TT IF	UND* FPU	9	100
TRUNC*fi*	*src*, read.*f*	*dest* write.*i*	Truncate Floating-Point to Integer *dest.i: = src.f* If overflow, then TRAP(FPU) (*src.f* rounded to zero)	— FSR:TT IF	UND* FPU	9	101
FLOOR*fi*	*src*, read.*f*	*dest* write.*i*	Floor Floating-Point to Integer *dest.i: = src.f* If overflow, then TRAP(FPU) (rounded *src.f* toward negative infinity)	— FSR:TT IF	UND* FPU	9	111
ADD*f*	*src*, read.*f*	*dest* rmw.*f*	Add Floating-Point *dest: = dest + src*	— FSR:TT UF IF	UND* FPU	11	0000
SUB*f*	*src*, read.*f*	*dest* rmw.*f*	Subtract Floating-Point *dest: = dest − src*	— FSR:TT UF IF	UND* FPU	11	0100
MUL*f*	*src*, read.*f*	*dest* rmw.*f*	Multiply Floating-Point *dest: = dest * src*	— FSR:TT UF IF	UND* FPU	11	1100
DIV*f*	*src*, read.*f*	*dest* rmw.*f*	Divide Floating-Point if *src* = 0, then TRAP(FPU) else *dest: = dest / src*	— FSR:TT UF IF	UND* FPU	11	1000
CMP*f*	*src1*, read.*f*	*src2* read.*f*	Compare Floating-Point Z: = 1 if *src2* = *src1;* else Z: = 0 N: = 1 if *src2* < *src1;* else N: = 0 L: = 0 (always)	Z N L FSR:TT	UND* FPU	11	0010

Floating-Point Instructions (Continued)

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
NEG*f*	*src*, read.*f*	*dest* write.*f*	Negate Floating-Point *dest:* = 0 − *src* (*src* sign bit complemented)	— FSR:TT	UND* FPU	11	0101
ABS*f*	*src*, read.*f*	*dest* rmw.*f*	Absolute Value of Floating-Point if *src* < 0, *dest:* = 0 − *src* if *src* > 0 *dest:* = *src*	— FSR:TT	UND* FPU	11	1101
LFSR	*src* read.D		Load FSR FSR: = *src*	— FSR:all	UND*	9	001
SFSR	*dest* write.D		Store FSR *dest:* = FSR	—	UND*	9	110

*TRAP (UND) if f bit in CFG is 0.

Quick Integer Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
MOVQ*i*	*src*, quick quick	*dest* write.*i* *dest:* = *src*	Move Quick Integer (*src* sign-extended to *dest* length)	—	—	2	101
CMPQ*i*	*src1*, quick quick	*src2* read.*i* N	Compare Quick Integer Z: = 1 if *src2* = src1; Z: = 0 otherwise N: = 1 if *src2* < *src1;* N: = 0 otherwise (signed operands) L: = 1 if *src2* < *src1;* L: = 0 otherwise (unsigned operands) (*src1* sign-extended to *src2* length)	Z L	—	2	001
ADDQ*i*	*src*, quick quick	*dest* rmw.*i*	Add Quick Integer *dest:* = *dest* + *src* C: = 1 on carry; C: = 0 on no carry F: = 1 on overflow; F: = 0 on no overflow; (*src* sign-extended to *dest* length)	C	—	2	000

Extended Integer Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
MEI*i*	*src* read.*i*	*dest* rmw.*2i*	Multiply Extended Integer (*dest:* = *src* * (*dest* mod 2**i) (unsigned operands) (low-order half of *dest*)	—	—	7	1001
DEI*i*	*src*, read.*i*	*dest* rmw.*2i*	Divide Extended Integer *dest:* = (*dest* div *src*)* 2**i + *dest* mod *src* (unsigned operands)	—	DVZ	7	1011

Boolean Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
NOT*i*	*src* read.*i*	*dest* write.*i*	NOT *dest: src* XOR 1	—	—	6	1001
S(cond)*i*		*dest* write.*i*	Save Condition Code as a Boolean if cond, then *dest:* = 1	—	—	2	011

Block Instructions

Syntax				Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
MOVM*i*	*block1,* addr	*block2,* addr	*length* disp cons4	Move Multiple *block2:* = *block1*	—	—	7	0000
CMPM*i*	*block1,* addr	*block2,* addr	*length* disp cons4	Compare Multiple Z: = 1 if *block1* = *block2;* else Z: = 0 N: = 1 if *block1* > *block2;* else N: = 0 (signed integers) L: = 1 if *block1* > *block2;* else L: = 0 (unsigned integers)	Z N L	—	7	0001

Packed Decimal Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
ADDP*i*	*src,* read.*i*	*dest* rmw.*i*	Add Packed Decimal *dest:* = *dest* + *src* + C C: = 1 on carry; C: = 0 on no carry F: = 0	C F	—	6	1111
SUBP*i*	*src,* read.*i*	*dest* rmw.*i*	Subtract Packed Decimal *dest:* = *dest* − *src* − C C: = 1 on borrow; C: = 0 on no borrow F: = 0	C F	—	6	1011

Array Instructions

Syntax				Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
INDEX*i*	*accum,* reg reg	*length,* read.*i*	*index* read.*i*	Calculate Array Index *accum:* = (*length* + 1) **accum* + *index*	—	—	8	100
CHECK*i*	*dest,* reg reg	*bounds,* addr	*src* read.*i*	Check Array Index if *bounds(upper)* > = *src* > = *bounds(lower)* then, *dest:* = *src* − *bounds (lower)* F: = 0 else; *dest:* = undefined F: = 1	F	—	8	011

Bit Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
TBIT*i*	*offset,* read.*i*	*base* regaddr	Test Bit F: = BIT*(base, offset)*	F	—	4	1101
SBIT|I|*i*	*offset,* read.*i*	*base* regaddr	Set Bit F: = BIT*(base, offset)* BIT*(base, offset):* = 0	F	—	6	0110 |0111|
IBIT*i*	*offset,* read.*i*	*base* regaddr	Invert Bit F: = BIT*(base, offset)* BIT*(base, offset):* = 1	F	—	6	0010 |0011|
CBIT|I|*i*	*offset,* read.*i*	*base* regaddr	Clear Bit F: = BIT*(base, offset)* BIT*(base, offset)* : = NOT|BIT*(base, offset)* |	F	—	6	1110
CVTP	*offset,* reg reg	*base,dest* addr write.D	Convert to Bit Pointer *dest:* = (8* ADDR*(base)* + *offset*) mod 2**32	—	—	8	001

FFS*i*	*base,* read.*i*	*offset* *rmw.B*	Find First Set Bit
			
			If (*offset* < 0 or *offset* ≥ length in bits of *base*),
			then operation is undefined
			else
			j: = *offset*
			while (j < length of *base* and BIT*(base,j)* = 0
			do j: = j + 1
			if j = length of *base* then
			F: = 1; *offset:* = 0, else
			F: = 0; *offset:* = j
				F	—	8	101

Bit Field Instructions

Syntax					Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
EXT*i*	*offset,* reg reg	*base,* regaddr	*dest,* write.*i*	*length* *disp*	Extract Field *dest:* = FIELD *(base, offset, length)*	—	—	8	000
EXTS*i*	*base,* regaddr	*dest,* write.*i*	*offset, length* imm cons3, cons5		Extract Field Short *dest:* = FIELD *(base, offset, length)*	—	—	7	0011
INS*i*	*offset,* reg reg	*src,* read.*i*	*base,* regaddr	*length* disp disp	Insert Field FIELD*(base, offset, length)* : = *src*	—	—	8	010
INSS*i*	*src,* read.*i*	*base,* regaddr	*offset, length* imm cons3, cons5		Insert Field Short FIELD *(base offset length)* : = *src*	—	—	7	0010

String Instructions

Syntax		Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
MOVS*i*	[B[,]][U|W]	Move String *string2* = *string1* F: = 1 if until/while condition is met; F: = 0 otherwise	F	—	5	0000
MOVST	[B[,]][U|W]	Move String with Translation *string2* = *translate-string* F: = 1 if until/while condition is met; F: = 0 otherwise	F	—	5	0000 i = 00
CMPS*i*	[B[,]][U|W]	Compare String F: = 1 if until/while condition is met; F: = 0 otherwise Z: = 1 if *string1* = *string2* and F: = 0; else Z: = 0 N: = 1 if *string2* < *string1* and F: = 0; else N: = 0 L: = 1 if *string2* < *string1* and F: = 0; else L: = 0	Z N L F	—	5	0001
CMPST	[B[,]][U|W]	Compare String with Translation F: = 1 if until/while condition is met; F: = 0 otherwise Z: = 0 if translate-string ≠ *string2* and F: = 0; Z: = 0 otherwise N: = 1 if translate-string > *string2* and F: = 0; N: = 0 otherwise L: = 1 if translate-string > *string2* and F: = 0; L: = 0 otherwise	Z N L F	—	5	0001 i = 00
SKPS*i*	[B[,]][U|W]	Skip String F: = 1 if until/while condition is met; F: = 0 otherwise	F	—	5	0011
SKPST	[B[,]][U|W]	Skip String with Translation F: = 1 if until/while condition is met; F: = 0 otherwise	F	—	5	0011 i = 00

Note: B, W, and U are optional. U and W are mutually exclusive. The comma is required whenever B and either U or W are specified.

Processor Control Instructions

Syntax				Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
DIRECT AND CONDITIONAL JUMPING								
JUMP	*dest* addr			Jump PC: = ADDR*(dest)*	—	—	3	0100
B*(cond)*	*disp* short	*disp* disp label		Conditional Branch if *cond,* then PC: = PC + *disp*	—	—	0	1010
BR	*disp* disp label			Unconditional Branch PC: = PC + *disp*	—	—	0	1110 1010
CASE*i*	*index* read.*i*			Case Branch PC: = PC + *index* (signed *index*)	—	—	3	1110
ACB*i*	*inc,* quick quick	*index,* rmw.*i*	*disp* *disp* label	Add, Compare, and Branch *index:* = *index* + *inc* if *index* < > 0, then PC: = PC + *disp*	—	—	2	100
SUBROUTINE AND PROCEDURES								
JSR	*dest* addr			Jump to Subroutine PUSHD(return address) PC: = ADDR*(dest)*	—	—	3	1100
BSR	*disp* disp label			Branch to Subroutine PUSHD(return address) PC: = PC + *(disp)*	—	—	1	0000
RET	*constant* disp disp			Return from Subroutine POPD(PC) SP: = SP + *constant*	—	—	1	0001
CXP	*constant* disp external			Call External Procedure SP: = SP − 2 PUSHW(MOD) PUSHD(return address) temp: = DOUBLEWORD (DOUBLEWORD(MOD + 4) + 4* *constant*) MOD: = WORD(temp) SB: = DOUBLEWORD (MOD + 0) PC: = DOUBLEWORD (MOD + 8) + WORD(temp + 2)	—	—	1	0010

Processor Control Jumping Block Instructions (Continued)

Syntax				Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
JBROUTINE AND PROCEDURES (Continued)								
CXPD	*desc* addr			Call External Procedure with Descriptor SP: = SP − 2 PUSHW(MOD) PUSHD(return address) MOD: = WORD*(descriptor)* SB: = DOUBLEWORD (MOD + 0) PC: = DOUBLEWORD (MOD + 8) + WORD(*descriptor* + 2)	—	—	3	0000
RXP	*constant* disp disp			Return from External Procedure POPD(PC) POPW(MOD) SB: = DOUBLEWORD (MOD + 0) SP: = SP + *constant* + 2	—	—	1	0011
ERVICE RETURN								
RETT	*constant* disp			Return from Trap § if U = 1, then TRAP(ILL) else POPD(PC) POPW(MOD) POPW(PSR) SB: = DOUBLEWORD (MOD) SP: = SP + *constant*	all	ILL	1	0100
RETI				Return from Interrupt § if U = 1, then TRAP(ILL) else POPD(PC) POPW(MOD) POPW(PSR) SB: = DOUBLEWORD (MOD)	all	ILL	1	0101

Integer Instructions

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
ARITHMETIC INSTRUCTIONS							
ADD*i*	*src*, read.*i*	*dest* rmw.*i*	Add *dest* = *dest* + *src* C: = 1 on carry; C: = 0 on no carry F: = 1 on overflow; F: = 0 on no overflow	C F	—	4	0000
ADDC*i*	*src*, read.*i*	*dest* rmw.*i*	Add with Carry *dest* = *dest* + *src* + C C: = 1 on carry; C: = 0 on no carry F: = 1 on overflow; F: = 0 on no overflow	C F	—	4	0100
CMP*i*	*src1*, read.*i*	*src2* read.*i*	Compare Z: = 1 if *src1* = *src2*; Z: = 0 otherwise N: = 1 if *src1* > *src2*; *N: = 0 otherwise* (signed operands) L: = 1 if *src1* > *src2*; L: = 0 otherwise (unsigned operands)	Z N L	—	4	0001
SUB*i*	*src*, read.*i*	*dest* rmw.*i*	Subtract *dest* = *dest* − *src* C: = 1 on borrow; C: = 0 on no borrow F: = 1 on overflow; F: = 0 on no overflow	C F	—	4	1000
SUBC*i*	*src*, read.*i*	*dest* rmw.*i*	Subtract with Borrow *dest* = *dest* − (*src* + C) C: = 1 on borrow; C: = 0 on no borrow F: = 1 on overflow; F: = 0 on no overflow	C F	—	4	1100
NEG*i*	*src*, read.*i*	*dest* write.*i*	Negate *dest* = 0 − *src* C: = 1 on carry; C: = 0 on no carry F: = 1 on overflow; F: = 0 on no overflow	C F	—	6	1000

Integer Instructions (Continued)

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
RITHMETIC INSTRUCTIONS (Continued)							

ABS*i*	*src*,
	read.*i*
		dest
		write.*i*
			Absolute Value
			
			if *src* < 0, then
			dest: = 0 − *src;*
			F: = 1 on overflow
			F: = 0 on no overflow
			else
			dest: = *src;* F: = 0
				F	—	6	1100

MUL*i*	*src*, read.*i*	*dest* rmw.*i*	Multiply *dest:* = *src* dest*	—	—	7	1000
DIV*i*	*src*, read.*i*	*dest* rmw.*i*	Divide if *src* = 0, then TRAP(DVZ) else *dest:* = *dest* DIV *src* (signed division; *dest* DIV *src* rounded toward negative infinity)	—	DVZ	7	1111
MOD*i*	*src*, read.*i*	*dest* rmw.*i*	Modulus if *src* = 0, then TRAP(DVZ) else *dest:* = *dest* − *src** (*dest* DIV *src*) (signed division; *dest* DIV *src* rounded toward negative infinity)	—	DVZ	7	1110
QUO*i*	*src*, read.*i*	*dest* rmw.*i*	Quotient if *src* = 0, then TRAP(DVZ) else *dest:* = *dest/src* (signed division; *dest/src* round toward zero)	—	DVZ	7	1100
REM*i*	*src*, read.*i*	*dest* rmw.*i*	Remainder if *src* = 0, then TRAP(DVZ) else *dest:* = *dest* − *src** *(dest/src)* (signed division; *dest/src* rounded toward zero)	—	DVZ	7	1101
MOVE INSTRUCTIONS							
MOV*i*	*src*, read.*i*	*dest* write.*i*	Move *dest:* = *src*	—	—	4	0101
MOVXBW	*src*, read.B	*dest* write.W	Move Sign-Extending Byte to Word *dest* (low-order byte): = *src* *dest* (high-order byte): = SIGN*(src)*	—	—	7	0100

Integer Instructions (Continued)

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
MOVE INSTRUCTIONS (Continued)							
MOVXBD	*src*, read.B	*dest* write.D	Move Sign-Extending Byte to Double-Word *dest* (low-order byte): = *src* *dest* (high-order byte): = SIGN*(src)*	—	—	7	0111
MOVXWD	*src*, read.W	*dest* write.D	Move Sign-Extending Word to Double-Word *dest* (low-order byte): = *src* *dest* (high-order byte): = SIGN*(src)*	—	—	7	0111
MOVZBW	*src*, read.B	*dest* write.W	Move Zero-Extending Byte to Word *dest* (low-order byte): = *src* *dest* (high-order bits): = 0	—	—	7	0101
MOVZBD	*src*, read.B	*dest* write.D	Move Zero-Extending Byte to Double-Word *dest* (low-order byte): = *src* *dest* (high-order bits): = 0	—	—	7	0110
MOVZWD	*src*, read.W	*dest* write.D	Move Zero-Extending Word to Double-Word *dest* (low-order word): = *src* *dest* (high-order bits): = 0	—	—	7	0110
ADDR	*src*, addr	*dest* write.D	Compute Effective Address *dest:* = ADDR*(src)*	—	—	4	1001
LXPD	EXT*(n)* addr external	*dest* write.D	Load External Procedure Descriptor *dest:* = Double-Word (Double-Word (mod + 4) + 4**n*)	—	—	4	1001 i = II
SHIFT INSTRUCTIONS							

ASH*i*	*count*, read.B	*dest* rmw.*i*	Arithmetic Shift (Left or Right)
			
			if *count* < 0, then
			dest: = *dest* shifted right by | *count* | bits, emptied bit positions filled from original sign bit.
			else
			dest: = *dest* shifted left by | *count* | bits, emptied bit positions filled with zero.
				—	—	6	0001

LSH*i*	*count*, read.B*i*	*dest* rmw.*i*	Logical Shift (Left or Right)
			
			if *count* < 0, then
			dest: = *dest* shifted right by | *count* | bits, emptied bit positions filled with zero.
			else
			dest: = *dest* shifted left by | *count* | bits, emptied bit positions filled with zero.
				—	—	6	0101

Integer Instructions (Continued)

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
SHIFT INSTRUCTIONS (Continued)							

ROT*i*	*count,* read.B*i*	*dest* rmw.*i*	Rotate (Left or Right)
			
			if *count* < 0, then
			dest: = *dest* shifted right by | *count* | bits, end-around.
			else
			dest: = *dest* shifted left by | *count* | bits, end around.
				—	—	6	0000

LOGICAL INSTRUCTIONS							
AND*i*	*src,* read.*i*	*dest* rmw.*i*	Logical AND *dest:* = *dest* AND *src*	—	—	4	1010
OR*i*	*src,* read.*i*	*dest* rmw.*i*	Logical OR *dest:* = *dest* OR *src*	—	—	4	0110
BIC*i*	*src,* read.*i*	*dest* rmw.*i*	Bit Clear *dest:* = *dest* AND NOT*(src)*	—	—	4	0010
XOR*i*	*src,* read.*i*	*dest* rmw.*i*	Exclusive Or *dest:* = *dest* XOR *src*	—	—	4	1110
COM*i*	*src,* read.*i*	*dest* write.*i*	Complement *dest:* = NOT*(src)*	—	—	6	1101

Processor Service Instructions

Syntax		Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
REGISTER/STACK MANIPULATION						
ADJSP*i*	*src* read.*i*	Adjust Stack Pointer SP: = SP − *src* (*src* is signed) S bit specifies current SP	—	—	3	1010
BICPSR(B|W)	*src* read.(B|W)	Bit Clear in PSR if § W length if BICPSRW and U = 1, then TRAP(ILL) else, PSR: = PSR AND NOT *(src)*	all	ILL	3	0010
BISPSR(B|W)	*src* read.(B|W)*i*	Bit Set in PSR § if W *length* if BISPSRW and U = 1, then TRAP(ILL) else, PSR: = PSR OR *src*	all	ILL	3	0110
SAVE	*reglist* imm reglist	Save General Purpose Registers for each register *Rn* in *reglist,* PUSHD *(Rn)* in reverse numerical order.	—	—	1	0110

Processor Service Instructions (Continued)

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
REGISTER/STACK MANIPULATION (Continued)							
RESTORE	*reglist* imm reglist		Restore General Purpose Registers for each register *Rn* in *reglist,* POPD*(Rn)* in reverse numerical order.	—	—	1	0111
ENTER	*reglist,* imm reglist	*constant* disp disp	Enter New Context PUSHD(FP) FP: = SP SP: = SP − *constant* for each register *Rn* in *reglist,* PUSHD*(Rn)* in numerical order.	—	—	1	1000
EXIT	*reglist* imm reglist		Exit Context for each register *Rn* in *reglist,* POPD*(Rn)* in reverse numerical order. SP: = FP POPD(FP)	—	—	1	1001
LPR*i*	*procreg.* short procreg	*src* read.*i*	Load Processor Register § If PSR or INTBASE If U = 1 and ((*procreg* = PSR) or (*procreg* = INTBASE)) then TRAP(ILL) else *procreg:* = *src* (S bit specifies current SP) (all flags affect if *procreg* = PSR or UPSR)	all	ILL	2	110
SPR*i*	*procreg.* short procreg	*dest* write.*i*	Store Processor Register § If PSR or INTBASE*i* If U = 1 and ((*procreg* = PSR) or (*procreg* = INTBASE)) then TRAP(ILL) else *dest:* = *procreg* (S bit specifies current SP)	—	ILL	2	010
SETCFG	*cfglist* short procreg		Set Configuration Register § If U = 1, then TRAP(ILL) else CFG: = short	—	—	5	0010

Processor Service Instructions (Continued)

Syntax			Operations	Flags Affected	Traps Taken	Instr. Format	Op Field
EXCEPTIONS							
BPT			Breakpoint Trap TRAP(BPT)	—	BPT	1	1111
SVC			Supervisor Call Trap TRAP(SVC)	—	SVC	1	1110
FLAG			Flag Trap – FLG If F = 1, then TRAP (FLG)	—	—	1	1101
MISCELLANEOUS							
NOP			No Operation PC: = PC + 1	—	—	1	1010
WAIT			Wait for Interrupt PC: = PC + 1 Wait until next interrupt	—	—	1	1011

Traps

Trap Name

DVZ = Divide by Zero Trap
a zero divisor in a Divide, Modulus, Quotient, Remainder, or Divide Extended Integer instruction.

ILL = Illegal Instruction Trap
a Privileged instruction when U = 1

UND = Undefined Instruction Trap
a Memory Management instruction when CFG M = 0, or a Floating-Point instruction when CFG F = 0, or any undefined operation codes.

FPU = Floating-Point Error Trap
a Floating-Point instruction on:
Underflow, Overflow, Invalid Division, Illegal Instruction, Reserved Operand, Inexact Result

SVC = Supervisor Trap
a Supervisor Call instruction.

FLG = Flag Trap
a Flag instruction when F = 1.

BPT = Breakpoint Trap
a Breakpoint instruction.

ABT = Instruction Abort Trap
a Page fault.

Operand Classes

The general operand access classes are:

read - the operand is read

write - the operand is written

rmw - the operand is read, modified, and written

addr - the address of the memory location designated by the operand, is calculated. Whether or not the addressed is accessed depands upon the instruction.

regaddr - the operand designates either a memory location or a general register which is in turn used as a base for a bit address calculation.

quick - 4-bit constant is read

reg - double-word from register is read

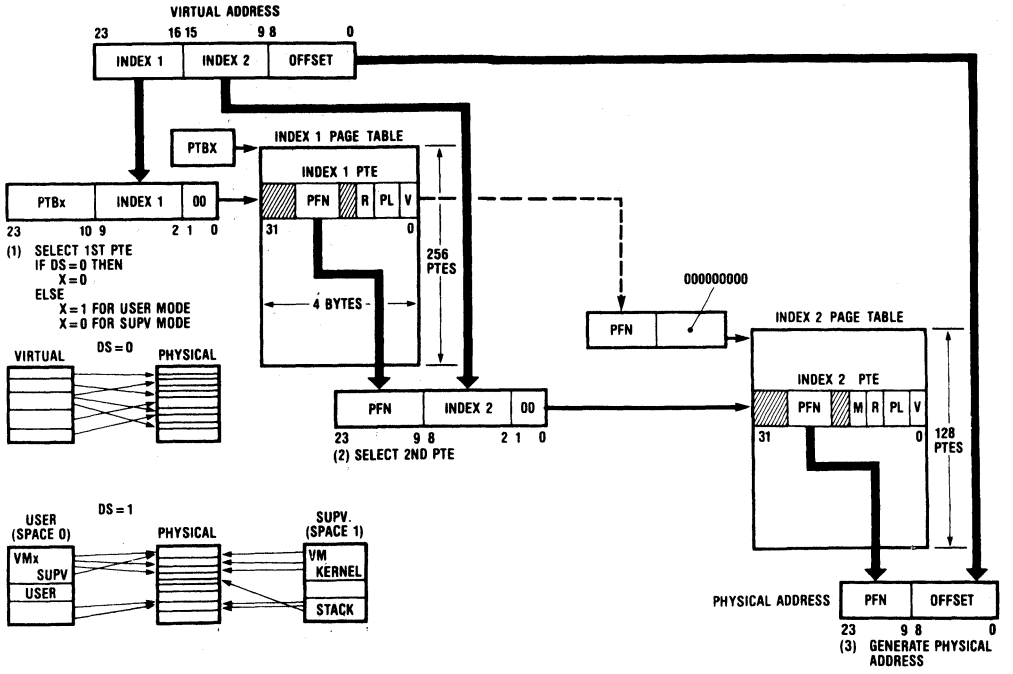
short - 4-bit condition code is read

imm - a) 8-bit register mask is read
b) concatenated 5-bit and 3-bit constants are read

disp - 1-, 2-, 4-byte displacement is read

NS32082 Memory Management

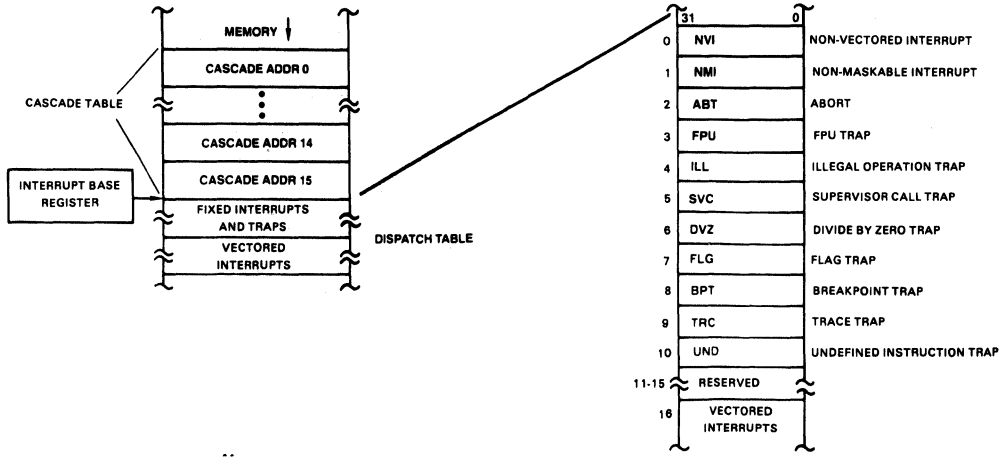
Virtual to Physical Address Translation



TL/PD/1K17-5

Series 32000 Central Processing Units

Interrupt Dispatch and Cascade Tables



TL/PD/1K17-6

Series 32000 Addressing Modes

Encoding	Mode	Assembler	Effective Address
REGISTER			
00000	Register 0	R0 or F0	None: Operand is in the specified register
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
REGISTER RELATIVE			
01000	Register 0 Relative	disp(R0)	Disp + Register
01001	Register 1 Relative	disp(R1)	
01010	Register 2 Relative	disp(R2)	
01011	Register 3 Relative	disp(R3)	
01100	Register 4 Relative	disp(R4)	
01101	Register 5 Relative	disp(R5)	
01110	Register 6 Relative	disp(R6)	
01111	Register 7 Relative	disp(R7)	
MEMORY RELATIVE			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
RESERVED			
10011	(Reserved for Future Use)		
IMMEDIATE			
10100	Immediate	value	None: Operand is input from instruction queue.
ABSOLUTE			
10101	Absolute	@disp	Disp.
EXTERNAL			
10110	External	EXT(disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp 1.
TOP OF STACK			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
MEMORY SPACE			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
SCALED INDEX			
11100	Index, bytes	mode[Rn:B]	EA(mode) + Rn
11101	Index, words	mode[Rn:W]	EA(mode) + 2 × Rn
11110	Index, double words	mode[Rn:D]	EA(mode) + 4 × Rn
11111	Index, quad words	mode[Rn:Q]	EA(mode) + 8 × Rn 'mode' and 'n' are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

Instruction Formats

NOTATIONS

i = Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f = Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c = Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op = Operation Code
- Valid encodings shown with each format.

gen, gen 1, gen 2 = General Addressing Mode Field

reglist = General Purpose Register Number

cond = Condition Code Field
- 0000 = EQual: Z = 1
- 0001 = Not Equal: Z = 0
- 0010 = Carry Set: C = 1
- 0011 = Carry Clear: C = 0
- 0100 = Higher: L = 1
- 0101 = Lower or Same: L = 0
- 0110 = Greater Than: N = 1
- 0111 = Less or Equal: N = 0
- 1000 = Flag Set: F = 1
- 1001 = Flag Clear: F = 0
- 1010 = LOwer: L = 0 and Z = 0
- 1011 = Higher or Same: L = 1 or Z = 1
- 1100 = Less Than: N = 0 and Z = 0
- 1101 = Greater or Equal: N = 1 or Z = 1
- 1110 = (Unconditionally True)
- 1111 = (Unconditionally False)

short = Short Immediate value. May contain quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.

procreg: CPU Dedicated Register, in LPR, SPR.
- 0000 = US
- 0001 – 0111 = (Reserved)
- 1000 = FP
- 1001 = SP
- 1010 = SB
- 1011 = (Reserved)
- 1100 = (Reserved)
- 1101 = PSR
- 1110 = INTBASE
- 1111 = MOD

Options: in String Instructions

U/W	B	T

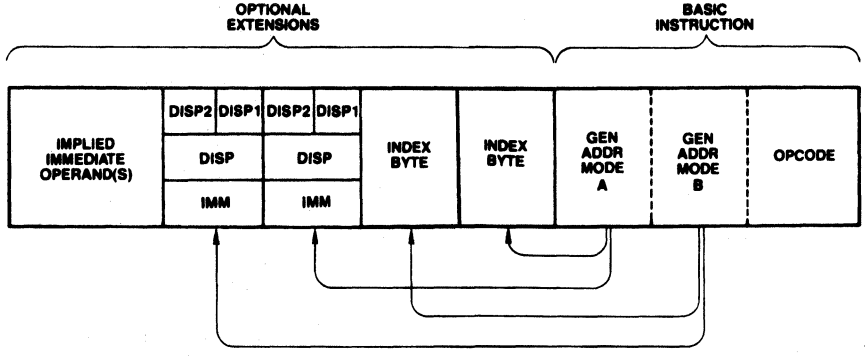
- T = Translated
- B = Backward
- U/W = 00: None
  - 01: While Match
  - 11: Until Match

Configuration bits, in cfglist (SETCFG):

C	M	F	I

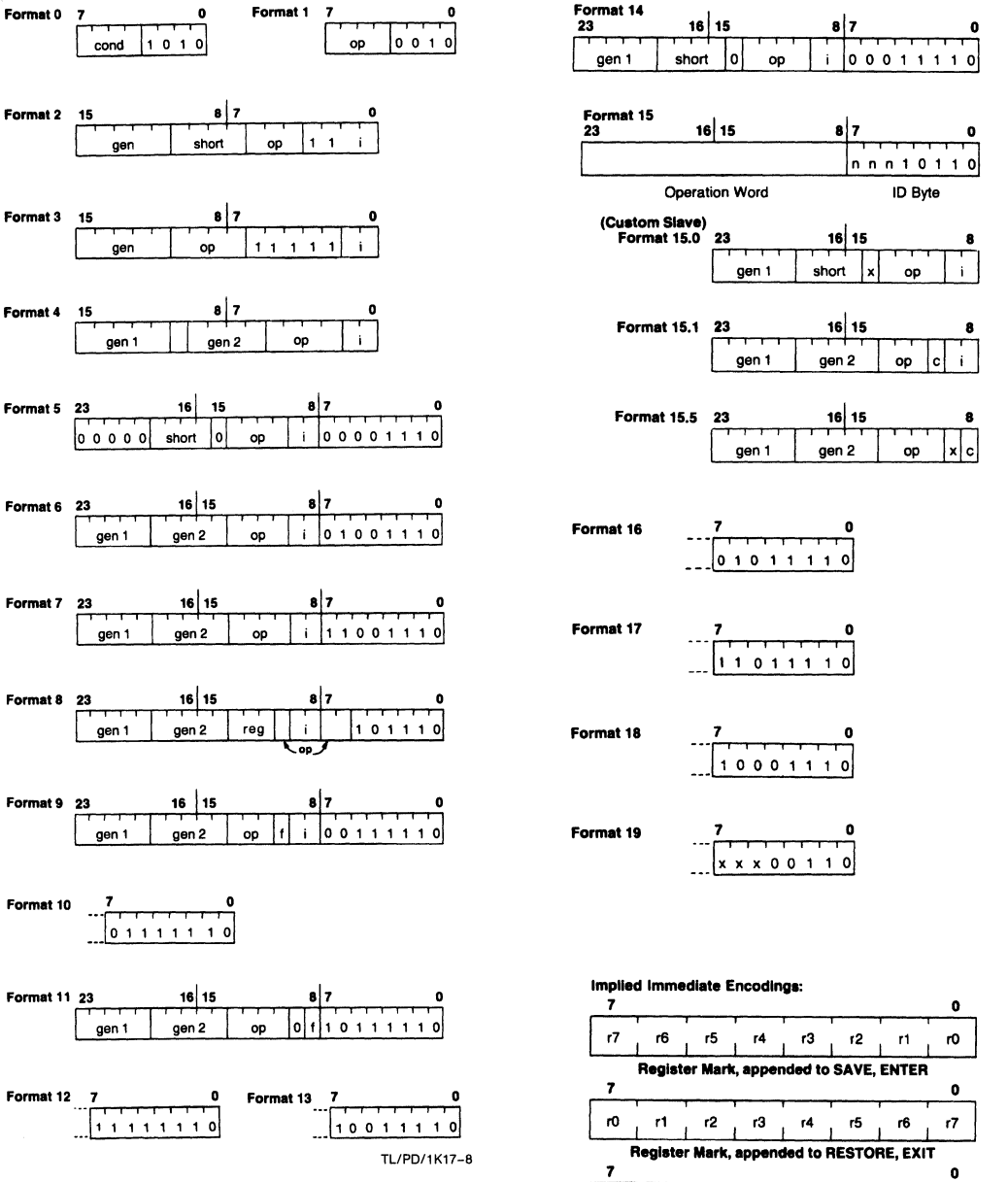
mmureg: MMU Register number, in LMR, SMR
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = PF0
- 0101 = PF1
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = SC
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA

General Instruction Format



TL/PD/1K17-7

General Instruction Format (Continued)



TL/PD/1K17-8

TL/PD/1K17-9

Series 32000 Instruction Execution Times

ASSUMPTIONS

The entire instruction, with all displacements and immediate operands, is assumed to be present in the instruction queue when needed.

Interference from instruction prefetches, which is very dependent upon the preceding instruction(s), is ignored. This assumption will tend to affect the timing estimate in an optimistic direction.

It is assumed that all memory operand transfers are completed before the next instruction begins execution. In the case of an operand of access class rmw in memory, this is pessimistic, as the Write transfer occurs in parallel with the execution of the next instruction.

It is assumed that there is no overlap between the fetch of an operand and the following sequences of microcode. This is pessimistic, as the fetch of Operand A will generally occur in parallel with the effective address calculation of Operand B, and the fetch of Operand B will occur in parallel with the execution phase of the instruction.

Where possible, the values of operands are taken into consideration when they affect instruction timing, and a range of times is given. Where this is not done, the worst case is assumed.

DEFINITIONS

TEA — The time required to calculate an operand's Effective Address. For a Register or Immediate operand, this includes the fetch of that operand.

TMMU — The extra clock cycle required for translation of memory addresses if a NS32082 MMU is present.

TOPB — The time needed to read or write a memory byte.

TOPW — The time needed to read or write a memory word.

TOPD — The time needed to read or write a memory double-word.

TOPi — The time needed to read or write a memory operand, where the operand size is given by the operation length of the instruction. It is always equivalent to either TOPB, TOPW or TOPD.

TCY — Internal processing overhead, in clock cycles.

L — Internal processing whose duration depends on the operation length. The number of clock cycles is derived by multiplying this value by the number of bytes in the operation length.

EQUATIONS

TMMU — If a NS32082 MMU is present then TMMU = 1
else TMMU = 0

TOPB — If operand is in a register or is immediate then TOPB = 0
else TOPB = 3 + TMMU

TOPW — If operand is in a register or is immediate then TOPW = 0
else if word-aligned (even address) then TOPW = 3 + TMMU
else TOPW = 7 + 2 * TMMU (for NS32016)
or TOPW = 3 + TMMU (for NS32032 and NS32332)

TOPD — If operand is in a register or is immediate then TOPD = 0
else if word-aligned (even address) then TODP = 7 + 2 * TMMU (for NS32016) or TOPD = 3 + TMMU (for NS32032 and NS32332)
else TOPD = 11 + 3 * TMMU (for NS32016)
or TOPD = 7 + 2 * TMMU (for NS32032 and NS32332)

TOPi — If operand is in a register or is immediate then TOPi = 0
else if i = byte then TOPi = TOPB
else if i = word then TOPi = TOPW
else (i = double-word) then TOPi = TOPD

TCY — TCY = 1

L — If i (operation length) = byte then L = 1
else if i = word then L = 2
else (i = double-word) L = 4

TEA — If REGISTER addressing then TEA = 2
if IMMEDIATE or ABSOLUTE addressing then TEA = 4
if REGISTER RELATIVE or MEMORY SPACE addressing then TEA = 5
if MEMORY RELATIVE addressing then TEA = 7 + TOPD
if TOP OF STACK addressing then
if access class = write then TEA = 4
if access class = read then TEA = 2
else TEA = 3
if EXTERNAL addressing then TEA = 11 + 2 * TOPD (for NS32016) or TEA = 11 + TOPD (for NS32032 and NS32332)
if SCALED INDEXED addressing then TEA = TI1 + TI2
where TI1 depends on scale factor:
if byte indexing TI1 = 5
if word indexing TI1 = 7
if double-word indexing TI1 = 8
if quad-word indexing TI1 = 10
and TI2 = TEA of the basemode except:
if basemode is REGISTER then TI2 = 5
if basemode is TOP OF STACK then TI2=4

CALCULATION OF TOTAL EXECUTION TIME (TEX)

TEX is obtained by performing the following steps:

1. Find the desired instruction in the table.
2. Calculate the values of TEA, TOPB, etc. using the numbers in the table and the equations given on the preceding page.
3. The result derived by adding together these values is the execution time (TEX) in clock cycles.

NOTES ON TABLE USE

Values in the TEA column indicate the number of effective addresses to be calculated. If the value in this column is less than the number of general operands in the instruction, this is because one or both operands are in registers and that instruction has an optimized form which eliminates TEA for such operands.

In the L column, multiply the entry by the operation length in bytes (1, 2 or 4).

In the TCY column, special notations sometimes appear:

n1–n2 means n1 minimum, n2 maximum

n1%n2 means that the instruction flushes the instruction queue after n1 clock cycles and nonsequentially fetches the next instruction. The value n2, indicating the total number of clock cycles in internally executing the instruction (including n1), is not generally useful. The most accurate technique for determining such timing depends on the size and alignment of the Basic Instruction portion of the next instruction, plus Index Bytes. If this portion can be read in one memory cycle, then the execution time is n1 + 10 (including the memory cycle). If more memory cycles are required, the value is n1 + 5 + 4*m, where m is the number of memory cycles required.

In the Notes column, notations held within angle brackets <> indicate alternatives in the form of the instruction which affect the execution time. A table entry which is affected by the form of the instruction may have multiple values, separated by slashes, corresponding to the alternatives. The notations are:

<M>	Memory form
<R>	Register form
<MM>	Memory-to-Memory form
<RM>	Register-to-Memory form
<MR>	Memory-to-Register form
<RR>	Register-to-Register form
x	either Register to Memory

EXAMPLE OF TABLE USAGE

Calculate TEX for the instruction:

CMPW R0, TOS

Operand A is in a register; Operand B is in memory. This means that we must use the table values corresponding to the <xM> case as given in the Notes column (<xM> meaning "anything to memory").

Only the TEA, TOPi and TCY columns have values assigned for the CMPi instruction. Therefore, they are the only ones that need to be calculated to find TEX. The blank columns are irrelevant to this instruction.

The TEA column contains 2 for the <xM> case. This means that effective address times have to be calculated for both operands. (For the <MR> case, the Register operand would have required no TEA time, therefore only the Memory operand TEA would have been necessary.) From the equations:

TEA (Register mode) = 2,

TEA (Top of Stack mode, read access class) = 2,

Total TEA = 2 + 2 = 4.

The TOPi column represents potential operand transfers to or from memory. For a Compare instruction, each operand is read once, for a total of two operand transfers.

TOPi (Word, Register) = 0,

TOPi (Word, TOS) = TOPW = 3 (assuming aligned, no MMU)

Total TOPi = 3

TCY is the time required for internal operation within the CPU. The TCY value for this case is 3.

TEX = TEA + TOPi + TCY = 4 + 3 + 3 = 10 machine cycles.

If the CPU is running at 10 MHz then a machine cycle (clock cycle) is 100 ns. Therefore, this instruction would have 10 X 100 ns, or 1.0 microseconds, to execute.

Basic and Memory Management Instructions NS32016 and NS32032 CPUs

Mnemonic	TEA	TOPB	TOPW	TOPD	TOPi	TCY	L	Notes
ABSi	2	—	—	—	2	9/8	—	src < 0 / src > = 0
ACBi	1	—	—	—	2	16/15%20	—	<M>, no branch / branch
ACBi	—	—	—	—	—	18/17%22	—	<R>, no branch / branch
ADDi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM> / <MR> / <RR>
ADDCi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM> / <MR> / <RR>
ADDPi	2	—	—	—	3	16/18	—	no carry / carry
ADDQi	1/0	—	—	—	2/0	6/4	—	<M> / <R>
ADDR	2/1	—	—	1/0	—	2/3	—	<xM> / <xR>
ADJSPi	1	—	—	—	1	6	—	
ANDi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM> / <MR> / <RR>
ASHi	2	1	—	—	2	14-45	—	
Bcond	—	—	—	—	—	7/6%10	—	no branch / branch
BICi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM> / <MR> / <RR>
BICPSRB	1	1	—	—	—	18%22	—	
BICPSRW	1	—	1	—	—	30%34	—	
BISPSRB	1	1	—	—	—	18%22	—	
BISPSRW	1	—	1	—	—	30%34	—	
BPT	—	—	4	3	—	40	—	

Basic and Memory Management Instructions NS32016 and NS32032 CPUs (Continued)

Mnemonic	TEA	TOPB	TOPW	TOPD	TOPi	TCY	L	Notes
BR	1	—	—	—	1	4%9	—	
BSR	—	—	—	1	—	6%16	—	
CASEi	1	—	—	—	1	4%9	—	
CBITi	2/1	2/0	—	—	1	15/7	—	<xM>/<xR>
CBITIi	2/1	2/0	—	—	1	15/7	—	<xM>/<xR>
CHECKi	2	—	—	—	3	7/10/11	—	high / low / ok
CMPi	2/1/0	—	—	—	2/1/0	3	—	<xM>/<MR>/<RR>
CMPMi	2	—	—	—	2 * n	9 * n + 24	—	n = # of elements in block
CMPQi	1/0	—	—	—	1/0	3	—	<M>/<R>
CMPSi	—	—	—	—	2 * n	35 * n + 53	—	n = # of elements, not Translated
CMPST	—	n	—	—	2 * n	38 * n + 53	—	Translated
COMi	2	—	—	—	2	7	—	
CVTP	2	—	—	1	—	7	—	
CXP	—	—	3	4	—	16%21	—	
CXPD	1	—	3	3	—	13%18	—	
DEIi	2/1	—	—	—	5/1	38/31	16	<xM>/<xR>
DIA	—	—	—	—	—	3%7	—	
DIVi	2	—	—	—	3	58–68	16	
ENTER	—	—	—	n + 1	—	4 * n + 18	—	n = # of general registers saved
EXIT	—	—	—	n + 1	—	5 * n + 17	—	n = # of general registers restored
EXTi	2	—	—	1	1	19–29	—	field in memory
EXTi	2	—	—	—	1	17–51	—	field in register
EXTSi	2	—	—	1	1	26–36	—	
FFSi	2	2	—	—	1	24–28	24	
FLAG	—	—	0/4	0/3	—	6/44	—	no trap / trap
IBITi	2/1	2/0	—	—	1	17/9	—	<xM>/<xR>
INDEXi	2	—	—	—	2	25	16	
INSi	2	—	—	2	1	29–39	—	field in memory
INSi	1	—	—	—	1	28–96	—	field in register
INSSi	2	—	—	2	1	39–49	—	
JSR	1	—	—	1	1	5%15	—	
JUMP	1	—	—	—	—	2%6	—	
LMR	1	—	—	—	1	30%34	—	
LPRi	1	—	—	—	1	19–33	—	
LSHi	2	1	—	—	2	14–45	—	
MEIi	2	—	—	—	4	23	16	
MODi	2	—	—	—	3	54–73	16	
MOVi	2/1/0	—	—	—	2/1/0	1/3/3	—	<xM>/<MR>/<RR>
MOVMi	2	—	—	—	2 * n	3 * n + 20	—	n = # of elements in block
MOVQi	1/0	—	—	—	1/0	2/3	—	<M>/<R>

Basic and Memory Management Instructions NS32016 and NS32032 CPUs (Continued)

Mnemonic	TEA	TOPB	TOPW	TOPD	TOPI	TCY	L	Notes
MOVSi	—	—	—	—	2 * n	13 * n + 18	—	n = # of elements no options
MOVSi	—	—	—	—	2 * n	24 * n + 54	—	B, W and/or U option in effect
MOVST	—	n	—	—	2 * n	27 * n + 54	—	Translated
MOVSUi	2	—	—	—	2	33%37	—	
MOVUSi	2	—	—	—	2	33%37	—	
MOVXBD	2	1	—	1	—	6	—	
MOVXBW	2	1	1	—	—	6	—	
MOVXWD	2	—	1	1	—	6	—	
MOVZBD	2	1	—	1	—	5	—	
MOVZBW	2	1	1	—	—	5	—	
MOVZWD	2	—	1	1	—	5	—	
MULi	2	—	—	—	3	15	16	
NEGi	2	—	—	—	2	5	—	
NOP	—	—	—	—	—	3	—	
NOTi	2	—	—	—	2	5	—	
ORi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM>/<MR>/<RR>
QUOi	2	—	—	—	3	49–55	16	
RDVAL	1	1	—	—	—	21	—	
REMi	2	—	—	—	3	57–62	16	
RESTORE	—	—	—	n	—	5 * n + 12	—	n = # of general registers restored
RET	—	—	—	1	—	2%8	—	
RETI	—	1	3	3	—	39%45	—	
RETT	—	—	2	2	—	35%41	—	
ROTi	2	1	—	—	2	14–45	—	
RXP	—	—	1	2	—	2%6	—	
Scondi	1	—	—	—	1	9/10	—	False / True
SAVE	—	—	—	n	—	4 * n + 13	—	n = # of general registers saved
SBITi	2/1	2/0	—	—	1	15/7	—	<xM>/<xR>
SBITIi	2/1	2/0	—	—	1	15/7	—	<xM>/<xR>
SETCFG	—	—	—	—	—	15	—	
SKPSi	—	—	—	—	n	27 * n + 51	—	n = # of elements, not Translated
SKPST	—	n	—	—	n	30 * n + 51	—	Translated
SMR	1	—	—	—	1	25	—	
SPRi	1	—	—	—	1	21–27	—	
SUBi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM>/<MR>/<RR>
SUBCi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM>/<MR>/<RR>
SUBPi	2	—	—	—	3	16/18	—	no carry / carry
SVC	—	—	4	3	—	40	—	
TBITi	2/1	1/0	—	—	1	14/4	—	<xM>/<xR>
WAIT	—	—	—	—	—	6–?	—	? = until an interrupt/reset
WRVAL	1	1	—	—	—	21	—	
XORi	2/1/0	—	—	—	3/1/0	3/4/4	—	<xM>/<MR>/<RR>

NS32081 FLOATING-POINT EXECUTION TIMES

The following section gives execution timing information for Floating-Point instructions. Some additional timing definitions are used, as given below.

f - The floating-point operation length.

Standard Floating (32 bits): f = 1

Long Floating (64 bits): f = 2

Tf - The time required to transfer 32 bits of a floating-point value to or from the NS32081 Floating-Point Unit.

Tf = 4 always.

Ti - The time required to transfer an integer value to or from the NS32081 Floating-Point Unit.

Byte: Ti = 2

Word: Ti = 2

Double-Word: T1 = 4

Floating-Point Instruction Execution Times

Instruction	Case	TEA	TOPD	TOPi	Ti	Tf	TCY
MOVf	<MM>	2	2f	—	—	2f	23
	<RR>	—	—	—	—	—	27
	<MR>	1	f	—	—	f	23
	<RM>	—	f	—	—	f	27
ADDf, SUBf	<MM>	2	3f	—	—	3f	70
	<RR>	—	—	—	—	—	74
	<MR>	1	f	—	—	f	70
	<RM>	1	2f	—	—	2f	70
MULf	<MM>	2	3f	—	—	3f	30 + 14f
	<RR>	—	—	—	—	—	34 + 14f
	<MR>	1	f	—	—	f	30 + 14f
	<RM>	1	2f	—	—	2f	30 + 14f
DIVf	<MM>	2	3f	—	—	3f	55 + 30f
	<RR>	—	—	—	—	—	59 + 30f
	<MR>	1	f	—	—	f	55 + 30f
	<RM>	1	2f	—	—	2f	55 + 30f
ABSf, NEGf	<MM>	1	2f	—	—	2f	20
	<RR>	—	—	—	—	—	24
	<MR>	1	f	—	—	f	20
	<RM>	—	f	—	—	f	24
CMPf	<MM>	2	2f	—	—	2f	45
	<RR>	—	—	—	—	—	49
	<MR>	1	f	—	—	f	45
	<RM>	1	f	—	—	f	45
MOVLF	<MM>	1	3	—	—	3	23
	<RR>	—	—	—	—	—	27
	<MR>	1	2	—	—	2	23
	<RM>	—	1	—	—	1	27
MOVFL	<MM>	1	3	—	—	3	22
	<RR>	—	—	—	—	—	26
	<MR>	1	1	—	—	1	22
	<RM>	—	2	—	—	2	26
MOVif	<MM>	1	f	1	1	f	53
	<MR>	1	—	1	1	—	53
ROUNDfi, TRUNCfi, FLOORfi	<MM>	1	f	1	1	f	53
	<RM>	—	-	1	1	—	66
SFSR	<M>	—	1	—	—	1	13
LFSR	<M>	1	1	—	—	1	18

Basic and Memory Management Instructions NS32332 CPU

TEA Table

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPi	TCY	L
TEA: Absolute	—	—	—	—	—	—	2	
TEA: External	—	—	—	—	2	—	6	
TEA: Memory relative	—	—	—	—	1	—	4	
TEA: Register	— —	— —	— —	— —	— —	— —	0 2, Class Address or mode is Index.	
TEA: Register relative/ Memory space	—	—	—	—	—	—	2	
TEA: TOS push	—	—	—	—	—	—	3	
TEA: TOS address /Modify /Index	—	—	—	—	—	—	2	
TEA: Index (B, W, D, Q)	—	—	—	—	1	—	2	

TGET Table

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPi	TCY	L
TGET: Absolute	—	—	—	—	—	1	2	
TGET: External	—	—	—	—	2	1	6	
TGET: Memory relative	—	—	—	—	1	1	4	
TGET: Register	—	—	—	—	—	—	0	
TGET: Register relative/ Memory space	—	—	—	—	—	1	2	
TGET: TOS pop	—	—	—	—	—	1	1	
TGET: TOS address /Modify /Index	—	—	—	—	—	1	2	
TGET: Index (B, W, D, Q)	—	—	—	—	—	1	2	
TGET: Immediate	— —	— —	— —	— —	— —	— —	3, First operand. 2, 2nd operand.	

NS32332 Instruction Execution Times Table

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPI	TCY			L
ABSi	1	1	—	—	—	1	6		<M>	
	1	1	—	—	—	1	9	Src < 0	<R>	
	1	1	—	—	—	1	8	Src > 0	<R>	
ACB1	—	1	—	—	—	1	7	No branch	<M>	
	—	1	—	—	—	1	7%8	Branch	<M>	
	—	1	—	—	—	1	9	No branch	<R>	
	—	1	—	—	—	1	7%10	Branch	<R>	
ADDi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	
ADDCi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	
ADDPi	—	2	—	—	—	1	21	No carry		
	—	2	—	—	—	1	20	Carry		
ADDQi	—	1	—	—	—	1	3		<M>	
	—	—	—	—	—	—	4		<R>	
ADDR	2	—	—	—	—	1	1		<M>	
	2	—	—	—	—	—	3		<R>	
ADJSPi	—	1	—	—	—	—	8			
ANDi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	
ASHi							N = Number of shifts			
	—	2	—	—	—	1	7	No Shift	<M>	
	—	1	—	—	—	—	8	No Shift	<R>	
	—	2	—	—	—	1	11 + N	Count < 0	<M>	
	—	2	—	—	—	—	12 + N	Count < 0	<R>	
	—	2	—	—	—	1	12 + N	Count > 0	<M>	
	—	2	—	—	—	—	13 + N	Count > 0	<R>	
Bcond	—	—	—	—	—	—	7	No branch		
	—	—	—	—	—	—	5%8	branch		
BICi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	
BICPSRB	—	1	—	—	—	—	14	No PSR (U)		
	—	1	—	—	—	—	13%16	PSR (U)		
BICPSRW	—	1	—	—	—	—	14	No PSR (U)		
	—	1	—	—	—	—	13%16	PSR (U)		
BISPSRB	—	1	—	—	—	—	14	No PSR (U)		
	—	1	—	—	—	—	13%16	PSR (U)		
BISPSRW	—	1	—	—	—	—	14	No PSR (U)		
	—	1	—	—	—	—	13%16	PSR (U)		
BPT	—	—	—	4	3	—	31			
BR	—	—	—	—	—	—	2%7			
BSR	—	—	—	—	1	—	5%7			
CASEi	—	—	—	—	—	—	6%9			
CBITi	1	1	2	—	—	—	17		<XM>	
	—	1	—	—	—	—	7		<XR>	
CBITIi	1	1	3	—	—	—	18		<XM>	
	—	1	—	—	—	—	7		<XR>	
CHECKi	1	1	—	—	—	1	10		High	
	1	1	—	—	—	2	19		Low	
	1	1	—	—	—	2	20		O.K.	

NS32332 Instruction Execution Times Table (Continued)

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPI	TCY	L
CMPi	— —	2 2	— —	— —	— —	— —	3 <XM>, <XR> 5 <XI>	
CMPMi	2 2 2	— — —	— — —	— — —	— — —	2n 2n 2n	8 * n + 19 i = B n = number of compares 8 * n + 22 i = W 8 * n + 23 i = D	
CMPQi	—	1	—	—	—	—	3	
CMPSi	— —	— —	— —	— —	— —	2n —	37 * n + (69–71) n = number of elements 13, no elements	
CMPST	— —	— —	n —	— —	— —	2n —	42 * n + (70–72) n = number of elements 13, no elements	
COMi	1 —	1 1	— —	— —	— —	1 —	6 <XM> 8 <XR>	
CVTP	2 2	— —	— —	— —	1 —	— —	4 <M> 6 <R>	
CXP	—	—	—	3	4	—	16 (Till flush Queue 9 cyc. + 2 TOPW + 2 TOPD).	
CXPD	1	—	—	3	3	—	14 (Till flush Queue 7 cyc. + 2 TOPW + 1 TOPD).	
DEIi DEIi	— —	2 1	— —	— —	— —	3 —	16 * i + (27–28), <XM> i = 1, B i = 2, W i = 4, D 16 * i + (36–37), <XR>	
DIA	—	—	—	—	—	—	2%5	
DIVi	— —	2 1	— —	— —	— —	1 —	49–56 <XM> 50–57 <XR>	
ENTER	— —	— —	— —	— —	1 n + 1	— —	9 , no operand n = number of register saved n + 14; n > 0	
EXIT	— —	— —	— —	— —	1 n + 1	— —	5 , no operand n = number of register saved 3n + 6; n > 0	
EXTi EXTi	2 1	— —	— —	— —	1 —	1 1	Field in memory: 18, (offset MOD 8) = 0 20 + (offset MOD 8), (offset MOD 8) > 0 Field in register: 13, (offset MOD 32) = 0 15 + (offset MOD 32), (offset MOD 32) > 0	

NS32332 Instruction Execution Times Table (Continued)

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPi	TCY	L
EXTSi	1	1	—	—	—	1	21 (offset MOD 8) = 0 23 + (offset MOD 8), (offset MOD 8) > 0	
FFSi	—	2	—	—	—	1	<XM> "1" not found = = = = = = = = = = = 17, offset = 0; 20 + offset, offset > 0; "1" found = = = = = = = = = = = i = operand size (1–4) 24 → 24 + i * 8, offset = 0 27 + offset → 24 + i * 8, offset > 0	
FFSi	—	1	—	—	—	—	<XR> "1" not found = = = = = = = = = = = 18, offset = 0; 21 + offset, offset > 0; "1" found = = = = = = = = = = = 25 → 25 + i * 8, offset = 0 28 + offset → 28 + i * 8, offset > 0	
FLAG	— —	— —	— —	— 4	— 3	- —	6, no trap. 35, trap.	
IBITi	1 —	1 1	2 —	— —	— —	- —	19 <XM> 9 <XR>	
INDEXi	—	2	—	—	—	—	83	
INSi	1	1	—	—	2	—	Field in memory, 29, (offset MOD 8) = 0 35 + 2 * (offset MOD 8), (offset MOD 8) > 0	
INSi	—	1	—	—	—	—	Field in register, 23, (offset MOD 32) = 0 29 + 2 * (offset MOD 32), (offset MOD 32) > 0	
INSSi	—	2	—	—	1	—	Field in memory, 29, (offset MOD 8) = 0 35 + 2 * (offset MOD 8), (offset MOD 8) > 0	

NS32332 Instruction Execution Times Table (Continued)

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPi	TCY	L
INSSi	—	1	—	—	—	—	Field in register, 31, (offset MOD 8) = 0 37 + 2 * (offset MOD 8), (offset MOD 8) > 0	
JSR	1	—	—	—	1	—	4%6	
JUMP	1	—	—	—	—	—	1%4	
LMR	—	1	—	—	—	—	5%9	
LPRi	—	1	—	—	—	—	10 ,UPSR 17%20 ,PSR 18 ,no PSR	
LSHi							N = Number of shifts	
	—	2	—	—	—	1	7 No Shift <M>	
	—	1	—	—	—	—	8 No Shift <R>	
	—	2	—	—	—	1	11 + N Count < 0 <M>	
	—	2	—	—	—	—	12 + N Count < 0 <R>	
	—	2	—	—	—	1	12 + N Count > 0 <M>	
	—	2	—	—	—	—	13 + N Count > 0 <R>	
MEIi	—	2	—	—	—	2	17, <XM>	16
	—	1	—	—	—	—	20, <XR>	16
MODi	—	2	—	—	—	1	<XM> 45–49, remainder = 0; 57–63, remainder<>0;	16
MODi	—	1	—	—	—	—	<XR> 46–50, remainder = 0; 58–64, remainder<>0;	16
MOVi	1	1	—	—	—	1	1, <XM>	
	—	1	—	—	—	—	3, <XR>	
MOVMi	2	—	—	—	—	2n	n = number of elements in block. 19 + 2 * n, i = Byte. 22 + 2 * n, i = Word. 23 + 2 * n, i = D. W.	
MOVQi	1	—	—	—	—	1	1, <M>	
	—	—	—	—	—	—	3, <R>	
MOVSi	—	—	—	—	—	2n	13 * n + 49, no option n = number of elements 19 * n + (84–86)	
	—	—	—	—	—	2n	B, W and/or U option in effect.	
MOVST	—	—	n	—	—	2n	24 * n + 85–87) B, W and/or U option in effect.	
MOVSUi	2	—	—	—	—	2	19	
MOVUSi	2	—	—	—	—	2	19	
MOVXBD	1	1	—	—	1	—	7 <XM>	
	—	1	—	—	—	—	5 <XR>	
MOVXBW	1	1	—	1	—	—	7 <XM>	
	—	1	—	—	—	—	5 <XR>	

NS32332 Instruction Execution Times Table (Continued)

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPI	TCY		L
MOVXWD	1	1	—	—	1	—	7	<XM>	
	—	1	—	—	—	—	5	<XR>	
MOVZBD	1	1	—	—	1	—	6	<XM>	
	—	1	—	—	—	—	4	<XR>	
MOVZBW	1	1	—	1	—	—	6	<XM>	
	—	1	—	—	—	—	4	<XR>	
MOVZWD	1	1	—	—	1	—	6	<XM>	
	—	1	—	—	—	—	4	<XR>	
MULi	—	2	—	—	—	1	11	<XM>	16
	—	1	—	—	—	—	12	<XR>	16
NEGi	1	1	—	—	—	1	4	<XM>	
	—	1	—	—	—	—	6	<XR>	
NOP	—	—	—	—	—	—	3		
NOTi	1	1	—	—	—	1	4	<XM>	
	—	1	—	—	—	—	6	<XR>	
ORi	—	2	—	—	—	1	3	<XM>	
	—	1	—	—	—	—	4	<XR>	
QUOi	—	2	—	—	—	1	41–46	<XM>	16
	—	1	—	—	—	—	42–47	<XR>	16
RDVAL	1	—	1	—	—	—	9		
REMi	—	2	—	—	—	1	46–51	<XM>	16
	—	1	—	—	—	—	47–52	<XR>	16
RESTORE	—	—	—	—	—	—	5	, no operand	
								n = number of register restored.	
	—	—	—	—	n	—		3 * n + 6; n > 0	
RET	—	—	—	—	1	—	5%5		
RETI	—	—	1	2	2	—		32, no slave (Till flush Queue 24 cyc. + 1 TOPB + 2 TOPW + 1 TOPD)	
RETI	—	—	2	2	3	—		32, with slave (Till flush Queue 24 cyc. + 2 TOPB + 2 TOPW + 2 TOPD)	
RETT	—	—	—	2	2	—		22 (Till flush Queue 14 cyc. + 2 TOPW + 1 TOPD)	
ROTi								N = Number of shifts	
	—	2	—	—	—	1	7	No Shift	<M>
	—	1	—	—	—	—	8	No Shift	<R>
	—	2	—	—	—	1	11 + N	Count < 0	<M>
	—	2	—	—	—	—	12 + N	Count < 0	<R>
	—	2	—	—	—	1	12 + N	Count > 0	<M>
	—	2	—	—	—	—	13 + N	Count > 0	<R>
RXP	—	—	—	1	2	—	6	(Till flush Queue 5 cyc.)	

NS32332 Instruction Execution Times Table (Continued)

Mnemonic	TEA	TGET	TOPB	TOPW	TOPD	TOPI	TCY			L
Scondi	1	—	—	—	—	1	4		<M>	
	—	—	—	—	—	—	7	, true	<R>	
							6	, false	<R>	
SAVE	—	—	—	—	—	—	5	, no operand		
								n = number of register saved.		
	—	—	—	—	n	—		n + 6; n > 0		
SBITi	1	1	2	—	—	—	17		<XM>	
	—	1	—	—	—	—	7		<XR>	
SBITIi	1	1	3	—	—	—	18		<XM>	
	—	1	—	—	—	—	7		<XR>	
SETCFG	—	—	—	—	—	—	5			
SKPSi	—	—	—	—	—	n		29 * n + (66–68) B, W and/or U option in effect.		
SKPST	—	—	n	—	—	n		34 * n + (67–69) B, W and/or U option in effect.		
SMR	1	—	—	—	—	1	7		<M>	
	—	—	—	—	—	—	9		<R>	
SPRi	1	—	—	—	—	1	7	, no UPSR;	<M>	
	1	—	—	—	—	1	10	, UPSR;	<M>	
	—	—	—	—	—	—	9	, no UPSR;	<R>	
	—	—	—	—	—	—	12	, UPSR;	<R>	
SUBi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	
SUBCi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	
SUBPi	—	2	—	—	—	1	21	No carry		
	—	2	—	—	—	1	20	Carry		
SVC	—	—	—	4	3	—	31			
TBITi	1	1	1	—	—	—	14		<M>	
	—	—	—	—	—	—	4		<R>	
WAIT	—	—	—	—	—	—	4–?	? = until on interrupt/Reset		
WRVAL	1	—	1	—	—	—	9			
XORi	—	2	—	—	—	1	3		<XM>	
	—	1	—	—	—	—	4		<XR>	

Series 32000® Quick Reference Timing

Listed below are the most frequently used timing specifications for the Series 32000 components. This listing is intended for quick reference only, and is not all-inclusive. Detailed timing specifications and diagrams are located in the "AC Electrical Characteristics" section of each datasheet in this volume.

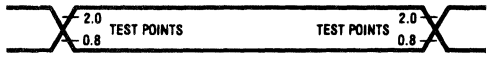


FIGURE 1

All the timing specifications given in this section refer to 0.8V or 2.0V on the input and output signals as illustrated in *Figure 1*, unless specifically stated otherwise.

ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge
T.E. — trailing edge F.E. — falling edge

NS32032 Output Signals: Internal Propagation Delays

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
			Min	Max	Min	Max	Min	Max	
t_{ALv}	Address Bits 0–23 Valid	after R.E., PHI1 T1		65		55		40	ns
t_{ALh}	Address Bits 0–23 Hold	after R.E., PHI1 Tmmu or T2	5		5		5		ns
t_{Dv}	Data Valid (Write Cycle)	after R.E., PHI1 T2		70		60		50	ns
t_{Dh}	Data Hold (Write Cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	Address Bits 0–23 Setup	before $\overline{ADS}$ T.E.	25		25		25		ns
t_{ALADSh}	Address Bits 0–23 Hold	after $\overline{ADS}$ T.E.	25		20		15		ns
t_{ALf}	Address Bits 0–23 Floating (No MMU)	after R.E., PHI1 T2		25		25		25	ns
t_{ADf}	Data Bits D24–D31 Floating (No MMU)	after R.E., PHI1 T2		25		25		25	ns
t_{ALMf}	Address Bits 0–23 Floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
t_{ADMf}	Data Bits 21–31 Floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
t_{BEv}	$\overline{BEn}$ Signals Valid	after R.E., PHI2 T4		70		60		45	ns
t_{BEh}	$\overline{BEn}$ Signals Hold	after R.E., PHI2 T4 or Ti	0		0		0		ns
t_{STv}	Status (ST0–ST3) Valid	after R.E., PHI1 T4 (before T1, see note)		65		55		45	ns
t_{STh}	Status (ST0–ST3) Hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	$\overline{DDIN}$ Signal Valid	after R.E., PHI1 T1		83		62		50	ns
t_{DDINh}	$\overline{DDIN}$ Signal Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	$\overline{ADS}$ Signal Active (Low)	after R.E., PHI1 T1		55		45		35	ns
t_{ADSia}	$\overline{ADS}$ Signal Inactive	after R.E., PHI2 T1		60		55		45	ns
t_{ADSw}	$\overline{ADS}$ Pulse Width	at 0.8V (both edges)	50		40		30		ns
t_{DSa}	$\overline{DS}$ Signal Active (Low)	after R.E., PHI1 T2		70		60		45	ns
t_{DSia}	$\overline{DS}$ Signal Inactive	after R.E., PHI1 T4		50		50		40	ns
t_{ALf}	AD0–AD23 Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns

NS32032 Output Signals: Internal Propagation Delays (Continued)

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
			Min	Max	Min	Max	Min	Max	
t_{ADf}	D24–D31 Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{DSf}	$\overline{DS}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{ADSf}	$\overline{ADS}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{BEf}	$\overline{BEn}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	$\overline{DDIN}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HLDAa}	$\overline{HLDA}$ Signal Active (Low)	after R.E., PHI1 Ti		100		90		75	ns
t_{HLDAia}	$\overline{HLDA}$ Signal Inactive	after R.E., PHI1 Ti		100		90		75	ns
t_{DSr}	$\overline{DS}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{ADSr}	$\overline{ADS}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{BEr}	$\overline{BEn}$ Signals Return from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINr}	$\overline{DDIN}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	$\overline{DDIN}$ Signal Floating (Caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		80		65		50	ns
t_{DDINr}	$\overline{DDIN}$ Signal Returns from Floating (Caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		75		65		50	ns
t_{SPCa}	$\overline{SPC}$ Output Active (Low)	after R.E., PHI1 T1		50		45		35	ns
t_{SPCia}	$\overline{SPC}$ Output Inactive	after R.E., PHI1 T4		50		45		35	ns
t_{SPCnf}	$\overline{SPC}$ Output Nonforcing	after R.E., PHI2 T4		40		25		10	ns
t_{Dv}	Data Valid (Slave Processor Write)	after R.E., PHI1 T1		80		65		50	ns
t_{Dh}	Data Hold (Slave Processor Write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	$\overline{PFS}$ Pulse Width	at 0.8V (both edges)	70		70		70		ns
t_{PFSa}	$\overline{PFS}$ Pulse Active (Low)	after R.E., PHI2		70		60		50	ns
t_{PFSia}	$\overline{PFS}$ Pulse Inactive	after R.E., PHI2		70		60		50	ns
t_{ILOs}	$\overline{ILO}$ Signal Setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns
t_{ILOh}	$\overline{ILO}$ Signal Hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t_{ILOa}	$\overline{ILO}$ Signal Active (Low)	after R.E., PHI1		70		70		70	ns
t_{ILOia}	$\overline{ILO}$ Signal Inactive	after R.E., PHI1		70		70		70	ns

NS32032 Output Signals: Internal Propagation Delays (Continued)

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
			Min	Max	Min	Max	Min	Max	
t_{USv}	U/$\overline{S}$ Signal Valid	after R.E., PHI1 T4		70		60		45	ns
t_{USh}	U/$\overline{S}$ Signal Hold	after R.E., PHI1 T4	10		10		10		ns
t_{NSPF}	Nonsequential Fetch to next $\overline{PFS}$ Clock Cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	$\overline{PFS}$ Clock Cycle to next Non-Sequential Fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	Last Operand Transfer of an Instruction to next $\overline{PFS}$ Clock Cycle	before R.E., PHI1 T1 of first of first bus cycle of transfer	0		0		0		t_{Cp}

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

NS32032 Input Signal Requirements

Name	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PWR}	Power Stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	Data in Setup (Read Cycle)	before F.E., PHI2 T3	20		15		10		ns
t_{DIh}	Data in Hold (Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	$\overline{HOLD}$ Active (Low) Setup Time (See Note)	before F.E., PHI2 TX1	25		25		25		ns
t_{HLDia}	$\overline{HOLD}$ Inactive Setup Time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	$\overline{HOLD}$ Hold Time	after R.E., PHI1 TX2	0		0		0		ns
t_{FLTa}	$\overline{FLT}$ Active (Low) Setup Time	before F.E., PHI2 Tmmu	25		25		25		ns
t_{FLTia}	$\overline{FLT}$ Inactive Setup Time	before F.E., PHI2 T2	25		25		25		ns
t_{RDYs}	RDY Setup Time	before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	RDY Hold Time	after F.E., PHI1 T3	0		0		0		ns
t_{ABTs}	$\overline{ABT}$ Setup Time ($\overline{FLT}$ Inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t_{ABTs}	$\overline{ABT}$ Setup Time ($\overline{FLT}$ Active)	before F.E., PHI2 T_f	30		25		20		ns
t_{ABTh}	$\overline{ABT}$ Hold Time	after R.E., PHI1	0		0		0		ns
t_{RSTs}	$\overline{RST}$ Setup Time	before F.E., PHI1	20		20		15		ns
t_{RSTw}	$\overline{RST}$ Pulse Width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	$\overline{INT}$ Setup Time	before F.E., PHI1	25		25		25		ns

NS32032 Input Signal Requirements (Continued)

Name	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
			Min	Max	Min	Max	Min	Max	
t_{NMIw}	$\overline{NMI}$ Pulse Width	at 0.8V (both edges)	70		70		70		ns
t_{DIs}	Data Setup (Slave Read Cycle)	before F.E., PHI2 T1	30		25		20		ns
t_{DIh}	Data Hold (Slave Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	$\overline{SPC}$ Pulse Delay from Slave	after R.E., PHI2 T4	40		35		25		ns
t_{SPCs}	$\overline{SPC}$ Setup Time	before F.E., PHI1	35		30		25		ns
t_{SPCw}	$\overline{SPC}$ Pulse Width from Slave Processor (Async Input)	at 0.8V (both edges)	30		25		20		ns
t_{ATs}	$\overline{AT}/\overline{SPC}$ Setup for Address Translation Strap	before R.E., PHI1 of cycle during which $\overline{RST}$ Pulse Is Removed	1		1		1		t_{Cp}
t_{ATh}	$\overline{AT}/\overline{SPC}$ Hold for Address Translation Strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		t_{Cp}

Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

NS32032 Clocking Requirements

Name	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
			Min	Max	Min	Max	Min	Max	
t_{CLr}	PHI1, PHI2 Rise Time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	PHI1, PHI2 Fall Time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	Clock Period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	PHI1, PHI2 Pulse Width	At 2.0V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 14 ns		0.5 t_{Cp} − 12 ns		0.5 t_{Cp} − 10 ns		ns
$t_{CLh(1,2)}$	PHI1, PHI2 High Time	At V_{CC} − 0.9V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 18 ns		0.5 t_{Cp} − 17 ns		0.5 t_{Cp} − 15 ns		ns
$t_{nOVL(1,2)}$	Non-Overlap Time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}	Non-Overlap Asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	at 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}	PHI1, PHI2 Asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	at 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

NS32016 Output Signals: Internal Propagation Delays

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
			Min	Max	Min	Max	Min	Max	
t_{ALv}	Address Bits 0–15 Valid	after R.E., PHI1 T1		65		55		40	ns
t_{ALh}	Address Bits 0–15 Hold	after R.E., PHI1 Tmmu or T2	5		5		5		ns
t_{Dv}	Data Valid (Write Cycle)	after R.E., PHI1 T2		70		60		50	ns
t_{Dh}	Data Hold (Write Cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{AHv}	Address Bits 16–23 Valid	after R.E., PHI1 T1		65		55		40	ns
t_{AHh}	Address Bits 16–23 Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	Address Bits 0–15 Setup	before $\overline{ADS}$ T.E.	25		25		25		ns
t_{AHADSs}	Address Bits 16–23 Set Up	before $\overline{ADS}$ T.E.	25		25		25		ns
t_{ALADSh}	Address Bits 0–15 Hold	after $\overline{ADS}$ T.E.	20		20		15		ns
t_{AHADSh}	Address Bits 16–23 Hold	after $\overline{ADS}$ T.E.	20		20		15		ns
t_{ALf}	Address Bits 0–15 Floating	after R.E., PHI1 T2 (no MMU)		25		25		25	ns
t_{ALMf}	Address Bits 0–15 Floating	after R.E., PHI1 Tmmu (with MMU)		25		25		25	ns
t_{AHMf}	Address Bits 16–23 Floating	after R.E., PHI1 Tmmu (with MMU)		25		25		25	ns
t_{HBEv}	$\overline{HBE}$ Signal Valid	after R.E., PHI1 T1		70		60		50	ns
t_{HBEh}	$\overline{HBE}$ Signal Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{STv}	Status (ST0–ST3) Valid	after R.E., PHI1 T4 (before T1, see note)		65		55		45	ns
t_{STh}	Status (ST0–ST3) Hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	$\overline{DDIN}$ Signal Valid	after R.E., PHI1 T1		83		62		50	ns
t_{ILOia}	$\overline{ILO}$ Signal Inactive	after R.E., PHI1		55		45		35	ns
t_{USv}	U/$\overline{S}$ Signal Valid	after R.E., PHI1 T4		55		45		35	ns
t_{USh}	U/$\overline{S}$ Signal Hold	after R.E., PHI1 T4	10		10		10		ns
t_{NSPF}	Nonsequential Fetch to next $\overline{PFS}$ Clock Cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	$\overline{PFS}$ Clock Cycle to next Non-Sequential Fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	Last Operand Transfer of an Instruction to next $\overline{PFS}$ Clock Cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t_{Cp}

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

NS32016 Input Signal Requirements

Name	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PWR}	Power Stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	Data in Setup (Read Cycle)	before F.E., PHI2 T3	25		20		15		ns
t_{DIh}	Data in Hold (Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	$\overline{HOLD}$ Active (Low) Setup Time (See Note)	before F.E., PHI2 TX1	25		25		25		ns

NS32016 Input Signal Requirements (Continued)

Name	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
			Min	Max	Min	Max	Min	Max	
t_{HLDia}	$\overline{HOLD}$ Inactive Setup Time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	$\overline{HOLD}$ Hold Time	after R.E., PHI1 TX2	0		0		0		ns
t_{FLTa}	$\overline{FLT}$ Active (Low) Setup Time	before F.E., PHI2 Tmmu	25		25		25		ns
t_{FLTia}	$\overline{FLT}$ Inactive Setup Time	before F.E., PHI2 T2	25		25		25		ns
t_{RDYs}	RDY Setup Time	before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	RDY Hold Time	after F.E., PHI1 T3	0		0		0		ns
t_{ABTs}	$\overline{ABT}$ Setup Time ($\overline{FLT}$ inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t_{ABTs}	$\overline{ABT}$ Setup Time ($\overline{FLT}$ active)	before F.E., PHI2 T_f	30		25		20		ns
t_{ABTh}	$\overline{ABT}$ Hold Time	after R.E., PHI1	0		0		0		ns
t_{RSTs}	$\overline{RST}$ Setup Time	before F.E., PHI1	20		20		15		ns
t_{RSTw}	$\overline{RST}$ Pulse Width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	$\overline{INT}$ Setup Time	before F.E., PHI1	25		25		25		ns
t_{NMIw}	$\overline{NMI}$ Pulse Width	at 0.8V (both edges)	80		75		70		ns
t_{DIs}	Data Setup (Slave Read Cycle)	before F.E., PHI2 T1	30		25		20		ns
t_{DIh}	Data Hold (Slave Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	$\overline{SPC}$ Pulse Delay from Slave	after R.E., PHI2 T4	40		35		25		ns
t_{SPCs}	$\overline{SPC}$ Setup Time	before F.E., PHI1	35		30		25		ns
t_{SPCw}	$\overline{SPC}$ Pulse Width from Slave Processor (Async. Input)	at 0.8V (both edges)	30		25		20		ns
t_{ATh}	$\overline{AT}/\overline{SPC}$ Hold for Address Translation Strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		t_{Cp}
t_{ATs}	$\overline{AT}/\overline{SPC}$ Setup for Address Translation Strap	before R.E., PHI1 of Cycle During which $\overline{RST}$ pulse is removed	1		1		1		t_{Cp}

Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

NS32016 Clocking Requirements

Name	Description	Reference/ Conditions	NS32016-6		NS32016-8		NS32016-10		Units
			Min	Max	Min	Max	Min	Max	
t_{CLr}	PHI1, PHI2 Rise Time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	PHI1, PHI2 Fall Time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	Clock period	R.E., PHI1, PHI2 to Next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	PHI1, PHI2 Pulse Width	At 2.0V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 14 ns		0.5 t_{Cp} − 12 ns		0.5 t_{Cp} − 10 ns		ns
$t_{CLh(1,2)}$	PHI1, PHI2 High Time	at V_{CC} − 0.9V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 18 ns		0.5 t_{Cp} − 17 ns		0.5 t_{Cp} − 15 ns		ns
$t_{nOVL(1,2)}$	Non-Overlap Time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}	Non-Overlap Asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}	PHI1, PHI2 Asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

NS32008 Output Signals: Internal Propagation Delays

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32008-6		NS32008-8		NS32008-10		Units
			Min	Max	Min	Max	Min	Max	
t_{ALv}	Address Bits 0–7 Valid	after R.E., PHI1 T1		80		65		50	ns
t_{ALh}	Address Bits 0–7 Hold	after R.E., PHI1 T2	5		5		5		ns
t_{Dv}	Data Valid (Write Cycle)	after R.E., PHI1 T2		80		65		50	ns
t_{Dh}	Data Hold (Write Cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{AHv}	Address Bits 8–23 Valid	after R.E., PHI1 T1		95		75		50	ns
t_{AHh}	Address Bits 8–23 Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ALADSs}	Address Bits 0–7 Set Up to $\overline{ADS}$ T.E.	before $\overline{ADS}$ reaches 2.0V	25		25		25		ns
t_{AHADSs}	Address Bits 8–23 Set Up to $\overline{ADS}$ T.E.	before $\overline{ADS}$ reaches 2.0V	25		25		25		ns
t_{ALADSh}	Address Bits 0–7 Hold from $\overline{ADS}$ T.E.	after $\overline{ADS}$ reaches 2.0V	15		15		15		ns
t_{ALf}	Address Bits 0–7 Floating	after R.E., PHI1 T2		25		25		25	ns
t_{STv}	Status (ST0–ST3) Valid	after R.E., PHI1 T4 (before T1, see note)		90		70		45	ns
t_{STh}	Status (ST0–ST3) Hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
t_{DDINv}	$\overline{DDIN}$ Signal Valid	after R.E., PHI1 T1		83		62		50	ns
t_{DDINh}	$\overline{DDIN}$ Signal Hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{ADSa}	$\overline{ADS}$ Signal Active (Low)	after R.E., PHI1 T1		55		45		35	ns
t_{ADSia}	$\overline{ADS}$ Signal Inactive	after R.E., PHI2 T1		60		55		45	ns
t_{ADSw}	$\overline{ADS}$ Pulse Width	at 0.8V (both edges)	50		40		30		ns
t_{DSa}	$\overline{DS}$ Signal Active (Low)	after R.E., PHI1 T2		70		60		45	ns
t_{DSia}	$\overline{DS}$ Signal Inactive	after R.E., PHI1 T4	10	60	10	50	10	40	ns
t_{ALf}	AD0–AD7 Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{AHf}	A8–A23 Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 T1		100		65		25	ns
t_{ADSf}	$\overline{ADS}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINf}	$\overline{DDIN}$ Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{HLDAa}	$\overline{HLDA}$ Signal Active (Low)	after R.E., PHI1 Ti		100		90		75	ns
t_{HLDAia}	$\overline{HLDA}$ Signal Inactive	after R.E., PHI1 Ti		100		90		75	ns
t_{ADSr}	$\overline{ADS}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{DDINr}	$\overline{DDIN}$ Signal Returns from Floating (Caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		100		80		55	ns
t_{SPCa}	$\overline{SPC}$ Output Active (Low)	after R.E., PHI1 T1		50		45		35	ns
t_{SPCia}	$\overline{SPC}$ Output Inactive	after R.E., PHI1 T4		50		45		35	ns
t_{SPCnf}	$\overline{SPC}$ Output Nonforcing	after R.E., PHI2 T4		40		25		10	ns
t_{Dv}	Data Valid (Slave Processor Write)	after R.E., PHI1 T1		80		65		50	ns
t_{Dh}	Data Hold (Slave Processor Write)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{PFSw}	$\overline{PFS}$ Pulse Width	at 0.8V (both edges)	70		70		70		ns
t_{PFSa}	$\overline{PFS}$ Pulse Active (Low)	after R.E., PHI2		70		60		50	ns
t_{PFSia}	$\overline{PFS}$ Pulse Inactive	after R.E., PHI2		70		60		50	ns

NS32008 Output Signals: Internal Propagation Delays (Continued)

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
			Min	Max	Min	Max	Min	Max	
t_{ILOs}	$\overline{ILO}$ Signal Setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns
t_{ILOh}	$\overline{ILO}$ Signal Hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t_{ILOa}	$\overline{ILO}$ Signal Active (Low)	after R.E., PHI1		70		65		55	ns
t_{ILOia}	$\overline{ILO}$ Signal Inactive	after R.E., PHI1		70		65		55	ns
t_{USv}	U/$\overline{S}$ Signal Valid	after R.E., PHI1 T4		70		60		45	ns
t_{USh}	U/$\overline{S}$ Signal Hold	after R.E., PHI1 T1	10		10		10		ns
t_{NSPF}	Nonsequential Fetch to Next $\overline{PFS}$ Clock Cycle	after R.E., PHI1 T1	4		4		4		t_{Cp}
t_{PFNS}	$\overline{PFS}$ Clock Cycle to Next Non-Sequential Fetch	before R.E., PHI1 T1	4		4		4		t_{Cp}
t_{LXPF}	Last Operand Transfer of an Instruction to Next $\overline{PFS}$ Clock Cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t_{Cp}

Note 1: Timing parameters for components with an "S" suffix are not guaranteed compatible with an NS32081 Slave Processor at a clock rate greater than 4 MHz.

Note 2: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . .Ti, T4, T1. . .". If the CPU was not idling, the sequence will be: ". . .T4, T1. . .".

NS32008 Input Signal Requirements

Name	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PWR}	Power Stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		50		50		μs
t_{DIs}	Data in Setup (Read Cycle)	before F.E., PHI2 T3	20		15		10		ns
t_{DIh}	Data in Hold (Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{HLDa}	$\overline{HOLD}$ Active (Low) Setup Time (See Note)	before F.E., PHI2 TX1	25		25		25		ns
t_{HLDia}	$\overline{HOLD}$ Inactive Setup Time	before F.E., PHI2 Ti	25		25		25		ns
t_{HLDh}	$\overline{HOLD}$ Hold Time	after R.E., PHI1 TX2	0		0		0		ns
t_{RDYs}	RDY Setup Time	before F.E., PHI2 T2 or T3	25		25		25		ns
t_{RDYh}	RDY Hold Time	after F.E., PHI1 T3	0		0		0		ns
t_{RSTs}	$\overline{RST}$ Setup Time	before F.E., PHI1	20		15		10		ns
t_{RSTw}	$\overline{RST}$ Pulse Width	at 0.8V (both edges)	64		64		64		t_{Cp}
t_{INTs}	$\overline{INT}$ Setup Time	before T.E., PHI1	20		20		20		ns
t_{NMIw}	$\overline{NMI}$ Pulse Width	at 0.8V (both edges)	70		70		70		ns
t_{DIs}	Data Setup (Slave Read Cycle)	before F.E., PHI2 T1	20		15		10		ns
t_{DIh}	Data Hold (Slave Read Cycle)	after R.E., PHI1 T4	10		10		10		ns
t_{SPCd}	$\overline{SPC}$ Pulse Delay from Slave	after R.E., PHI2 T4	17		13		10		ns
t_{SPCs}	$\overline{SPC}$ Setup Time	Before F.E., PHI1	42		32		25		ns
t_{SPCw}	SPC Pulse Width from Slave Processor (Async. Input)	at 0.8V (both edges)	30		25		20		ns

Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, and the state of the RDY input.

NS32008 Clocking Requirements

Name	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
			Min	Max	Min	Max	Min	Max	
t_{CLr}	PHI1, PHI2 Rise Time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	PHI1, PHI2 Fall Time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	Clock Period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	PHI1, PHI2 Pulse Width	at 2.0V on PHI1, PHI2 (both edges)	0.5 t_{Cp} − 14 ns		0.5 t_{Cp} − 12 ns		0.5 t_{Cp} − 10 ns		ns
$t_{CLh(1,2)}$	PHI1, PHI2 High Time	at V_{CC} − 0.9V on PHI1, PHI2 (both edges)	0.5t_{Cp} − 18 ns		0.5t_{Cp} − 17 ns		0.5t_{Cp} − 15 ns		ns
$t_{nOVL(1,2)}$	Non-Overlap Time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI1, PHI2	0	7	0	7	0	7	ns
t_{nOVLas}	Non-Overlap Asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	at 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}	PHI1, PHI2 Asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	at 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

NS32201 Clock Signals

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{Cp}	Clock Period	PHI1 R.E. to Next PHI1 R.E.	160		120		100		ns
t_{CLh}	Clock High Time	At V_{CC} − 0.9V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp} − 17 ns	0.5 t_{Cp} − 7 ns	0.5 t_{Cp} − 16 ns	0.5 t_{Cp} − 7 ns	0.5 t_{Cp} − 15 ns	0.5 t_{Cp} − 7 ns	ns
t_{CLl}	Clock Low Time	At 0.8V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp}	0.5 t_{Cp} + 12 ns	0.5 t_{Cp}	0.5 t_{Cp} + 11 ns	0.5 t_{Cp}	0.5t_{CP} + 10 ns	ns
$t_{CLw(1,2)}$	Clock Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp} − 14 ns	0.5 t_{Cp} − 4 ns	0.5 t_{Cp} − 12 ns	0.5 t_{Cp} − 4 ns	0.5 t_{Cp} − 10 ns	0.5 t_{Cp} − 4 ns	ns
t_{CLwas}	PHI1, PHI2 Asymmetry (t_{CLW1}) − $t_{CLW(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns
t_{CLR}	Clock Rise Time	0.8V to V_{CC} − 0.9V on PHI1, PHI2 R.E.		9		8		7	ns
t_{CLF}	Clock Fall Time	V_{CC} − 0.9V to 0.8V on PHI1, PHI2 F.E.		7		7		7	ns
$t_{nOVL(1,2)}$	Clock Nonoverlap Time	0.8V on PHI1, PHI2 F.E. to 0.8V on PHI2, PHI1 R.E.	0	5	0	5	0	5	ns
t_{nOVLas}	Non-Overlap Asymmetry ($t_{nOVL(1)}$ − $t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{XIR}	XIN Rise Time	1.5V to V_{CC} − 1.5V on XIN R.E.		15		11		9	ns
t_{XIF}	XIN Fall Time	V_{CC} − 1.5V to 1.5V on XIN F.E.		15		11		9	ns
t_{Xh}	XIN High Time (External Input)	2.5V on XIN R.E. to 2.5V on XIN F.E.	25		20		16		ns

NS32201 Timing Tables

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{XI}	XIN Low Time (External Input)	2.5V on XIN F.E. to 2.5V on XIN R.E.	25		20		16		ns
t_{XFr}	XIN to FCLK R.E. Delay	2.5V on XIN R.E. to FCLK R.E.	15	29	15	28	15	27	ns
t_{XFf}	XIN to FCLK F.E. Delay	2.5V on XIN F.E. to FCLK F.E.	15	29	15	28	15	27	ns
t_{XCr}	XIN to CTTL R.E. Delay	2.5V on XIN R.E. to CTTL R.E.	24	40	24	39	24	35	ns
t_{XPr}	XIN to PHI1 R.E. Delay	2.5V on XIN R.E. to PHI1 R.E.	21	40	21	37	21	32	ns
t_{FCR}	FCLK Rise Time	0.8V to 2.0V on FCLK R.E.		12		9		7	ns
t_{FCF}	FCLK Fall Time	2.0V to 0.8V on FCLK F.E.		12		9		7	ns
t_{FCr}	FCLK to CTTL R.E. Delay	FCLK R.E. to CTTL R.E.	5	17	5	16	5	15	ns
t_{FCf}	FCLK to CTTL F.E. Delay	FCLK R.E. to CTTL F.E.	5	17	5	16	5	15	ns
t_{FPr}	FCLK to PHI1 R.E. Delay	FCLK R.E. to PHI1 R.E.	2	17	2	13	2	10	ns
t_{FPf}	FCLK to PHI1 F.E. Delay	FCLK R.E. to PHI1 F.E.	−4	8	−4	6	−4	4	ns
t_{Fw}	FCLK High Time with Crystal	At 2.0V on FCLK (Both Edges)	0.25 t_{Cp} − 7 ns	0.25 t_{Cp} + 7 ns	0.25 t_{Cp} − 6 ns	0.25 t_{Cp} + 6 ns	0.25 t_{Cp} − 5 ns	0.25 t_{Cp} + 5 ns	ns
t_{PCf}	PHI2 R.E.to CTTL F.E. Delay	PHI2 R.E. to CTTL F.E.	−8	12	−7	11	−6	10	ns
t_{CTw}	CTTL High Time	At 2.0V on CTTL (Both Edges)	0.5 t_{Cp} − 8 ns	0.5 t_{Cp} + 8 ns	0.5 t_{Cp} − 8 ns	0.5 t_{Cp} + 8 ns	0.5 t_{Cp} − 7 ns	0.5 t_{Cp} + 7 ns	ns

NS32201 CTTL Timing CL = 50 pF

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PCr}	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	−2	7	−2	6	−2	5	ns
t_{CTR}	CTTL Rise Time	0.8V to 2.0V on CTTL R.E.		6		6		5	ns
t_{CTF}	CTTL Fall Time	2.0V to 0.8V on CTTL F.E.		5		5		4	ns

Note 1: PHI1 and PHI2 are interchangeable for the following parameters: t_{Cp}, t_{CLh}, t_{CLl}, t_{CLw}, t_{CLR}, t_{CLF}, t_{nOVL}, t_{XPr}, t_{FPr}, t_{FPf}.

NS32201 CTTL Timing CL = 100 pF

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PCr}	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	−2	8	−2	7	−2	6	ns
t_{CTR}	CTTL Rise Time	0.8V to 2.0V on CTTL R.E.		8		8		7	ns
t_{CTF}	CTTL Fall Time	2.0V to 0.8V on CTTL F.E.		6		6		5	ns

NS32201 Control Inputs ($\overline{RST1}$, $\overline{RSTO}$, $\overline{ADS}$, $\overline{DDIN}$) Timing

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{RSTr}	$\overline{RSTO}$ R.E. Delay	After PHI1 R.E.		25		20		15	ns
t_{RSTs}	$\overline{RST1}$ Setup Time	Before PHI1 R.E.	20		20		20		ns
t_{ADs}	$\overline{ADS}$ Setup Time	Before PHI1 R.E.	30		28		25		ns
t_{ADw}	$\overline{ADS}$ Pulse Width	$\overline{ADS}$ L.E. to $\overline{ADS}$ T.E.	25		25		25		ns
t_{DDs}	$\overline{DDIN}$ Setup Time	Before PHI1 R.E.	10		10		10		ns
t_{DDh}	$\overline{DDIN}$ Hold Time	After PHI1 R.E.	15		15		15		ns

NS32201 Control Outputs ($\overline{TSD}$, $\overline{RD}$, $\overline{WR}$, $\overline{DBE}$ & $\overline{RWEN}$/SYNC) Timing

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{Tf}	$\overline{TSO}$ L.E. Delay	After PHI1 R.E.		12		11		10	ns
t_{Tr}	$\overline{TSO}$ T.E. Delay	After PHI1 R.E.		20		18		15	ns
$t_{RWf(F)}$	$\overline{RD}/\overline{WR}$ L.E. Delay (Fast Cycle)	After PHI1 R.E.		50		40		30	ns
$t_{RWf(S)}$	$\overline{RD}/\overline{WR}$ L.E. Delay (Peripheral Cycle)	After PHI1 R.E.		30		23		15	ns
t_{RWr}	$\overline{RD}/\overline{WR}$ T.E. Delay	After PHI1 R.E.		25		23		20	ns
$t_{DBf(W)}$	$\overline{DBE}$ L.E. Delay (Write Cycle)	After PHI1 R.E.		35		30		24	ns
$t_{DBf(R)}$	$\overline{DBE}$ L.E. Delay (Read Cycle)	After PHI2 R.E.		30		23		15	ns
t_{DBr}	$\overline{DBE}$ T.E. Delay	After PHI2 R.E.		20		20		20	ns
t_{pLZ}	$\overline{RD}$,$\overline{WR}$ Low Level to TRI-STATE	After $\overline{RWEN}$/SYNC R.E.		20		20		20	ns
t_{pHZ}	$\overline{RD}$,$\overline{WR}$ High Level to TRI-STATE	After $\overline{RWEN}$/SYNC R.E.		20		20		20	ns
t_{pZL}	$\overline{RD}$,$\overline{WR}$ TRI-STATE to Low Level	After $\overline{RWEN}$/SYNC F.E.		25		23		20	ns
t_{pZH}	$\overline{RD}$,$\overline{WR}$ TRI-STATE to High Level	After $\overline{RWEN}$/SYNC F.E.		25		23		20	ns

NS32201 Wait States & Cycle Hold ($\overline{CWAIT}$, $\overline{WAIT}$n, $\overline{PER}$ & $\overline{RDY}$) Timing

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
$t_{CWs(H)}$	$\overline{CWAIT}$ Setup Time (Cycle Hold)	Before PHI1 R.E.	35		30		25		ns
$t_{CWh(H)}$	$\overline{CWAIT}$ Hold Time (Cycle Hold)	After PHI1 R.E.	0		0		0		ns
$t_{CWs(W)}$	$\overline{CWAIT}$ Setup Time (Wait States)	Before PHI2 R.E.	13		12		10		ns
$t_{CWh(W)}$	$\overline{CWAIT}$ Hold Time (Wait States)	After PHI2 R.E.	20		14		8		ns
t_{Ws}	$\overline{WAIT}$n Setup Time	Before PHI2 R.E.	5		5		5		ns
t_{Wh}	$\overline{WAIT}$n Hold Time	After PHI2 R.E.	25		20		15		ns
t_{Ps}	$\overline{PER}$ Setup Time	Before PHI1 R.E.	0		0		0		ns
t_{Ph}	$\overline{PER}$ Hold Time	After PHI1 R.E.	30		25		20		ns
t_{Rd}	$\overline{RDY}$ Delay	After PHI2 R.E.		30		26		23	ns

NS32201 Synchronization (SYNC) Timing

Name	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
			Min	Max	Min	Max	Min	Max	
t_{Sys}	SYNC Setup Time	Before FCLK R.E.	20		19		18		ns
t_{Syh}	SYNC Hold Time	After FCLK R.E.	3		2		0		ns
t_{CS}	CTTL/SYNC Inversion Delay	CTTL (master) to $\overline{RWEN}$/SYNC (slave)		25		20		15	ns

NS32081 Output Signal Propagation Delays

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/ Conditions	NS32081-6		NS32081-8		NS32081-10		Units
			Min	Max	Min	Max	Min	Max	
t_{Dv}	Data Valid	After $\overline{SPC}$ L.E.		65		55		45	ns
t_{Df}	D_0–D_{15} Floating	After $\overline{SPC}$ T.E.		50		50		50	ns
t_{SPCFw}	$\overline{SPC}$ Pulse Width from FPU	At 0.8V (Both Edges)	$t_{CLKp} - 50$	$t_{CLKp} + 50$	$t_{CLKp} - 50$	$t_{CLKp} + 50$	$t_{CLKp} - 50$	$t_{CLKp} + 50$	ns
t_{SPCFl}	$\overline{SPC}$ Output Active	After CLK R.E.		90		70		55	ns
t_{SPCFh}	$\overline{SPC}$ Output Inactive	After CLK R.E.		90		70		55	ns
t_{SPCFnf}	$\overline{SPC}$ Output Nonforcing	After CLK F.E.		75		55		45	ns

NS32081 Input Signal Requirements

Name	Description	Reference/ Conditions	NS32081-6		NS32081-8		NS32081-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PWR}	Power Stable to $\overline{RST}$ R.E.	After V_{CC} Reaches 4.5V	50		50		50		μs
t_{RSTw}	$\overline{RST}$ Pulse Width	At 0.8V (Both Edges)	64		64		64		t_{CLKp}
t_{Ss}	Status (ST0–ST1) Setup	Before $\overline{SPC}$ L.E.	75		60		50		ns
t_{Sh}	Status (ST0–ST1) Hold	After $\overline{SPC}$ L.E.	90		70		40		ns
t_{Ds}	D0–D15 Setup Time	Before $\overline{SPC}$ T.E.	75		55		40		ns
t_{Dh}	D0–D15 Hold Time	After $\overline{SPC}$ T.E.	80		65		50		ns
t_{SPCw}	SPC Pulse Width from CPU	At 0.8V (Both Edges)	100		85		70		ns
t_{SPCs}	$\overline{SPC}$ Input Active	Before CLK R.E.	50		45		40		ns
t_{SPCh}	$\overline{SPC}$ Input Inactive	After CLK R.E.	0		0		0		ns
t_{RSTs}	$\overline{RST}$ Setup	Before CLK F.E.	10		10		10		ns
t_{RSTh}	$\overline{RST}$ R.E. Delay	After CLK R.E.	0		0		0		ns

NS32081 Clocking Requirements

Name	Description	Reference/ Conditions	NS32081-6		NS32081-8		NS32081-10		Units
			Min	Max	Min	Max	Min	Max	
t_{CLKh}	Clock High Time	At 2.0V (Both Edges)	60	1000	50	1000	42	1000	ns
t_{CLKl}	Clock Low Time	At 0.8V (Both Edges)	60	1000	50	1000	42	1000	ns
t_{CLKp}	Clock Period	CLK R.E. to Next CLK R.E.	160	2000	125	2000	100	2000	ns

NS32082 Output Signals: Internal Propagation Delays

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
			Min	Max	Min	Max	Min	Max	
t_{ALv}	Address Bits 0–15 Valid	After R.E., PHI1 Tmmu or T1		60		55		40	ns
t_{ALh}	Address Bits 0–15 Hold	After R.E., PHI1 T2	5		5		5		ns
t_{AHh}	Address Bits 16–24 Valid	After R.E., PHI1 Tmmu or T1		65		55		40	ns
t_{AHh}	Address Bits 16–24 Hold	After R.E., PHI1 T2	5		5		5		ns
t_{ALPAVs}	Address Bits 0–15 Set Up	Before $\overline{PAV}$ T.E.	25		25		25		ns
t_{AHPAVs}	Address Bits 16–24 Set Up	Before $\overline{PAV}$ T.E.	25		25		25		ns
t_{ALPAVh}	Address Bits 0–15 Hold	After $\overline{PAV}$ T.E.	20		20		15		ns
t_{AHPAVh}	Address Bits 16–24 Hold	After $\overline{PAV}$ T.E.	20		20		15		ns
t_{ALf}	AD0–AD15 Floating	After R.E., PHI1 T2		25		25		25	ns
t_{AHf}	A16–A24 Floating	After R.E., PHI1 T2 or T1		25		25		25	ns
t_{ALz}	AD0–AD15 Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 Ti		45		35		25	ns
t_{AHZ}	A16–A24 Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 Ti		45		35		25	ns
t_{ALr}	AD0–AD15 Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 T1		75		65		50	ns
t_{AHr}	A16–A24 Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 T1		80		65		50	ns
t_{Dv}	Data Valid (Memory Write)	After R.E., PHI1 T2		80		65		50	ns
t_{Dh}	Data Hold (Memory Write)	After R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{Dv}	Data Valid (Slave Processor Read)	After R.E., PHI1 T1		80		65		50	ns
t_{Dh}	Data Hold (Slave Processor Read)	After R.E., PHI1 next T1 or Ti	0		0		0		ns
t_{Df}	Data Bits Floating (Slave Processor Read)	After R.E., PHI1 T1 or Ti		10		10		10	ns
t_{DDINv}	$\overline{DDIN}$ Signal Valid	After R.E., PHI1 T1 or Tmmu		80		65		50	ns
t_{DDINh}	$\overline{DDIN}$ Signal Hold	After R.E., PHI1 T1 or Ti	0		0		0		ns
t_{DDINf}	$\overline{DDIN}$ Signal Floating	After R.E., PHI1 T2						25	ns
t_{DDINz}	$\overline{DDIN}$ Signal Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 Ti		85		70		50	ns
t_{DDINr}	$\overline{DDIN}$ Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI1 T1 or Ti		85		70		50	ns
t_{DDINAf}	$\overline{DDIN}$ Floating after Abort ($\overline{FLT}$ = 0)	After R.E., PHI2 T2		25		25		25	ns
t_{PAVa}	$\overline{PAV}$ Signal Active	After R.E., PHI1 Tmmu or T1		55		45		35	ns
t_{PAVia}	$\overline{PAV}$ Signal Inactive	After R.E., PHI2 Tmmu or T1		60		55		45	ns

NS32082 Output Signals: Internal Propagation Delays (Continued)

Maximum times assume capacitive loading of 100 pF

Name	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
			Min	Max	Min	Max	Min	Max	
t_{PAVw}	$\overline{PAV}$ Pulse Width	At 0.8V (Both Edges)	50		40		30		ns
t_{PAVdz}	$\overline{PAV}$ Floating Delay	After $\overline{HLDAI}$ F.E.		40		30		25	ns
t_{PAVdr}	$\overline{PAV}$ Return from Floating	After $\overline{HLDAI}$ R.E.		40		30		25	ns
t_{PAVz}	$\overline{PAV}$ Floating (Caused by $\overline{HOLD}$)	After R.E., PHI2 T4		50		40		30	ns
t_{PAVr}	$\overline{PAV}$ Return from Floating (Caused by $\overline{HOLD}$)	After R.E., PHI2 Ti		50		40		30	ns
t_{FLTa}	$\overline{FLT}$ Signal Active	After R.E., PHI1 Tmmu		90		75		60	ns
t_{FLTia}	$\overline{FLT}$ Signal Inactive	After R.E., PHI1 Tmmu, T_f or T2		55		45		35	ns
t_{ABTa}	Abort Signal Active	After R.E., PHI1 Tmmu or T1		90		70		55	ns
t_{ABTia}	Abort Signal Inactive	After R.E., PHI1 T2		80		65		50	ns
t_{ABTw}	Abort Pulse Width	At 0.8V (Both Edges)	110		90		70		ns
t_{INTa}	$\overline{INT}$ Signal Active	After R.E., PHI1 Tmmu or T_f						55	ns
t_{INTia}	$\overline{INT}$ Signal Inactive	After R.E., PHI1 T2		80		65		50	ns
t_{INTw}	$\overline{INT}$ Pulse Width	At 0.8V (Both Edges)	110		90		70		ns
t_{SPCa}	$\overline{SPC}$ Signal Active	After R.E., PHI1 T1		55		45		35	ns
t_{SPCia}	$\overline{SPC}$ Signal Inactive	After R.E., PHI1 T4		55		45		35	ns
t_{SPCf}	$\overline{SPC}$ Signal Floating	After F.E., PHI1 T4		35		30		25	ns
t_{SPCw}	$\overline{SPC}$ Pulse Width	At 0.8V (Both Edges)	110		90		70		ns
t_{HLD0da}	$\overline{HLDA0}$ Assertion Delay	After $\overline{HLDAI}$ F.E.		60		55		50	ns
$t_{HLD0dia}$	$\overline{HLDA0}$ Deassertion Delay	After $\overline{HLDAI}$ R.E.		60		55		50	ns
t_{HLD0a}	$\overline{HLDA0}$ Signal Active	After R.E., PHI1 Ti		50		40		30	ns
t_{HLD0ia}	$\overline{HLDA0}$ Signal Inactive	After R.E, PHI1 Ti		50		40		30	ns
t_{ATa}	$\overline{AT}/\overline{SPC}$ Signal Active	After R.E., PHI1		50		40		35	ns
t_{ATia}	$\overline{AT}/\overline{SPC}$ Signal Inactive	After R.E., PHI1		50		40		35	ns
t_{ATf}	$\overline{AT}/\overline{SPC}$ Signal Floating	After F.E., PHI1		35		30		25	ns
t_{RST0a}	$\overline{RST}/\overline{ABT}$ Asserted (Low)	After R.E. PHI1		40		35		30	ns
t_{RST0ia}	$\overline{RST}/\overline{ABT}$ Deasserted (High)	After R.E. PHI1 Ti		40		35		30	ns

NS32082 Input Signal Requirements

Name	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
			Min	Max	Min	Max	Min	Max	
t_{DIs}	Data In Set Up (Memory Read)	Before F.E., PHI2 T3	25		20		15		ns
t_{DIh}	Data In Hold (Memory Read)	After R.E., PHI1 T4	0		0		0		ns
t_{DIs}	Data In Set Up (Slave Processor Write)	Before F.E., PHI2 T1	30		25		20		ns
t_{DIh}	Data In Hold (Slave Processor Write)	After R.E., PHI1 T4	0		0		0		ns

NS32082 Input Signal Requirements (Continued)

Name	Description	Reference/Conditions	NS32082-6		NS32082-8		NS32082-10		Units
			Min	Max	Min	Max	Min	Max	
t_{RDYs}	RDY Signal Set Up	Before F.E., PHI2 T2 or T3	25		20		15		ns
t_{RDYh}	RDY Signal Hold	After F.E., PHI1 T3	0		0		0		ns
t_{USs}	U/$\overline{S}$ Signal Set Up	Before F.E., PHI2 T4 or Ti	45		40		35		ns
t_{USh}	U/$\overline{S}$ Signal Hold	After R.E., PHI1 Next T4	0		0		0		ns
t_{STs}	Status Signals Set Up	Before F.E., PHI2 T4 or Ti	70		60		45		ns
t_{STh}	Status Signals Hold	After R.E., PHI1 Next T4	0		0		0		ns
t_{SPCs}	$\overline{SPC}$ Input Set Up	Before F.E., PHI2 T1	70		55		45		ns
t_{SPCh}	$\overline{SPC}$ Input Hold	After R.E., PHI1 T4	0		0		0		ns
t_{HLDs}	$\overline{HOLD}$ Signal Set Up	Before F.E., PHI2 T4 or Ti	25		25		25		ns
t_{HLDh}	$\overline{HOLD}$ Signal Hold	After F.E., PHI2 T4 or Ti	0		0		0		ns
t_{HLDIs}	$\overline{HLDAI}$ Signal Set Up	Before F.E., PHI2 Ti	30		25		20		ns
t_{HLDIh}	$\overline{HLDAI}$ Signal Hold	After F.E., PHI2 Ti	0		0		0		ns
t_{HBFs}	A24/$\overline{HBF}$ Signal Set Up	Before F.E., PHI2	20		15		10		ns
t_{HBFh}	A24/$\overline{HBF}$ Signal Hold	After F.E., PHI2	0		0		0		ns
t_{RSTIs}	Reset Input Set Up	Before F.E., PHI1	20		20		20		ns
t_{PWR}	Power Stable to $\overline{RSTI}$ R.E.	After V_{CC} Reaches 4.5V	50		50		50		μs
t_{RSTIw}	$\overline{RSTI}$ Pulse Width	At 0.8V (Both Edges)	64		64		64		t_{Cp}

NS32082 Clocking Requirements

Name	Description	Reference/ Conditions	NS32082-6		NS32082-8		NS32082-10		Units
			Min	Max	Min	Max	Min	Max	
t_{CLr}	PHI1, PHI2 Rise Time	0.8V to V_{CC} − 0.9V on R.E., PHI1, PHI2		9		8		7	ns
t_{CLf}	PHI1, PHI2 Fall Time	V_{CC} − 0.9V to 0.8V on F.E., PHI1, PHI2		9		8		7	ns
t_{Cp}	Clock Period	R.E., PHI1, PHI2 to Next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	PHI1, PHI2 Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp} − 14 ns		0.5 t_{Cp} − 12 ns		0.5 t_{Cp} − 10 ns		ns
$t_{CLh(1,2)}$	PHI1, PHI2 High Time	At V_{CC} − 0.9V on PHI1, PHI2 (Both Edges)	0.5 t_{Cp} − 18 ns		0.5 t_{Cp} − 17 ns		0.5 t_{Cp} − 15 ns		ns
$t_{nOVL(1,2)}$	Non-overlap Time	0.8V on F.E. PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	0	7	0	7	0	7	ns
t_{nOVLas}	Non-overlap Asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−4	4	−4	4	−4	4	ns
t_{CLwas}	PHI1, PHI2 Asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	At 2.0V on PHI1, PHI2	−5	5	−5	5	−5	5	ns

NS32202 Timing Specification Standard

Timing Tables

Symbol	Description	Reference/Conditions	NS32202-6		NS32202-8		NS32202-10		Units
			Min	Max	Min	Max	Min	Max	
READ CYCLE									
t_{AhRDia}	Address Hold Time	After $\overline{RD}$ T.E.	80		80		80		ns
t_{AsRDa}	Address Setup Time	Before $\overline{RD}$ L.E.	50		50		40		ns
$t_{CShRDia}$	$\overline{CS}$ Hold Time	After $\overline{RD}$ T.E.	80		80		80		ns
t_{CSsRDa}	$\overline{CS}$ Setup Time	Before $\overline{RD}$ L.E.	50		50		30		ns
t_{DhRDia}	Data Hold Time	After $\overline{RD}$ T.E.	0	50	0	50	0	50	ns
t_{RDaDv}	Data Valid	After $\overline{RD}$ L.E.		200		175		150	ns
t_{RDw}	$\overline{RD}$ Pulse Width	At 0.8V (Both Edges)	220		190		160		ns
t_{SsRDa}	ST1 Setup Time	Before $\overline{RD}$ L.E.	50		50		50		ns
t_{ShRDia}	ST1 Hold Time	After $\overline{RD}$ T.E.	−30		−30		−30		ns
WRITE CYCLE									
t_{AhWRia}	Address Hold Time	After $\overline{WR}$ T.E.	80		80		80		ns
t_{AsWRa}	Address Setup Time	Before $\overline{WR}$ L.E.	50		50		50		ns
$t_{CShWRia}$	$\overline{CS}$ Hold Time	After $\overline{WR}$ T.E.	80		80		80		ns
t_{CSsWRa}	$\overline{CS}$ Setup Time	Before $\overline{WR}$ L.E.	50		50		50		ns
t_{DhWRia}	Data Hold Time	After $\overline{WR}$ T.E.	60		60		50		ns
t_{DsWRia}	Data Setup Time	Before $\overline{WR}$ T.E.	150		125		100		ns
t_{WRiaPf}	Port Output Floating	After $\overline{WR}$ T.E. (To PDIR)		200		200		200	ns
t_{WRiaPv}	Port Output Valid	After $\overline{WR}$ T.E.		200		200		200	ns
t_{WRw}	$\overline{WR}$ Pulse Width	At 0.8V (Both Edges)	220		190		160		ns
OTHER TIMINGS									
t_{COUTl}	Internal Sampling Clock Low Time	At 0.8V (Both Edges)	50		50		50		ns
t_{COUTp}	Internal Sampling Clock Period		400		400		400		ns
t_{SCINh}	External Sampling Clock High Time	At 2.0V (Both Edges)	100		100		100		ns
t_{SCINl}	External Sampling Clock Low Time	At 0.8V (Both Edges)	100		100		100		ns
t_{SCINp}	External Sampling Clock Period		800		800		800		ns
t_{Ch}	External Clock High Time (Without Prescaler)	At 2.0V (Both Edges)	160		130		100		ns
t_{Chp}	External Clock High Time (With Prescaler)	At 2.0V (Both Edges)	80		65		50		ns
t_{Cl}	External Clock Low Time (Without Prescaler)	At 0.8V (Both Edges)	160		130		100		ns
t_{Clp}	External Clock Low Time (With Prescaler)	At 0.8V (Both Edges)	80		65		50		ns
t_{Cy}	External Clock Period (Without Prescaler)		650		500		400		ns
t_{Cyp}	External Clock Period (With Prescaler)		160		130		100		ns
t_{GCOUTl}	Counter Output Transition Delay	After CLK F.E.		300		300		300	ns

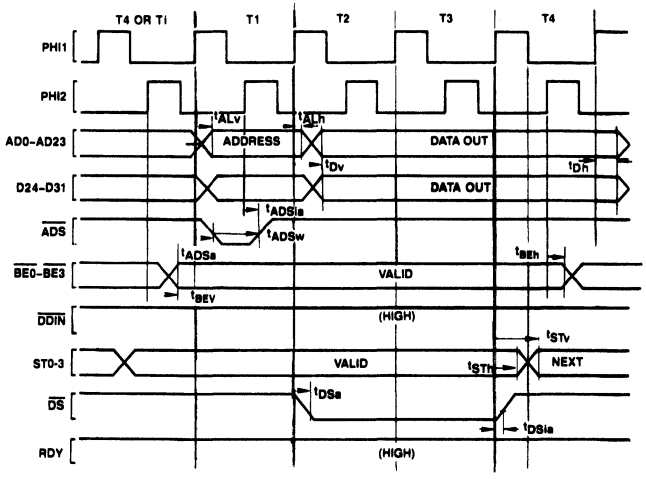
NS32202 Timing Specification Standard (Continued)

OTHER TIMINGS (Continued)

Symbol	Description	Reference/Conditions	NS32202-6		NS32202-8		NS32202-10		Units
			Min	Max	Min	Max	Min	Max	
t_{COUTw}	Counter Output Pulse Width in Pulsed Form	At 0.8V (Both Edges)	50		50		50		ns
t_{ACKIR}	Interrupt Request Delay	After Previous Interrupt Acknowledge	500		500		500		ns
t_{IRId}	$\overline{INT}$ Output Delay	After Interrupt Request Active		800		800		800	ns
t_{IRw}	Interrupt Request Pulse Width in Edge Trigger		50		50		50		ns
t_{RSTw}	$\overline{RST}$ Pulse Width	At 0.8V (Both Edges)	400		400		400		ns

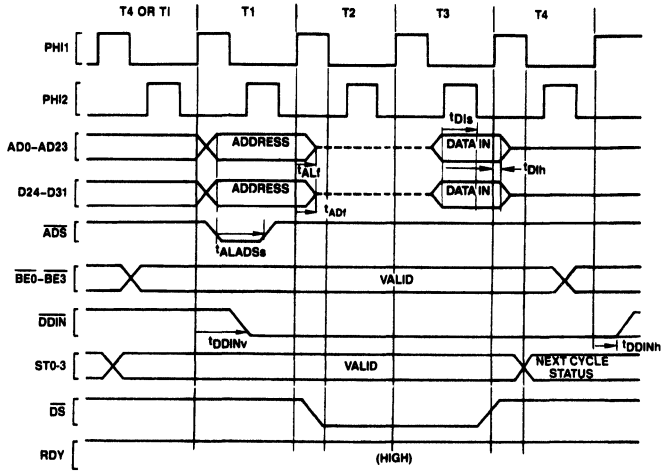
NS32032 Read and Write Cycle Timing

Write Cycle



B11K20-2

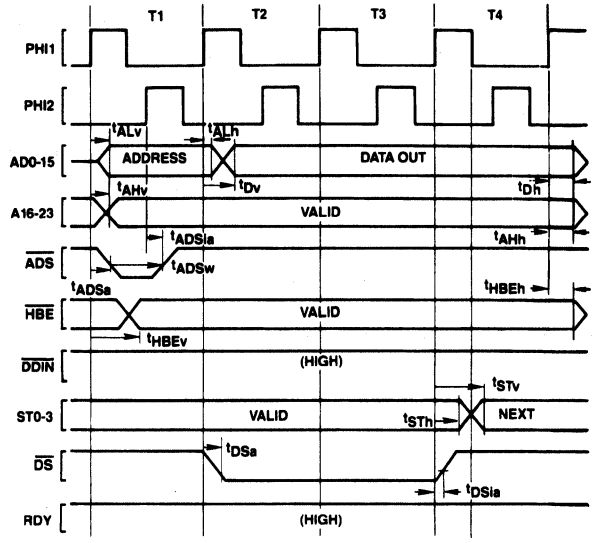
Read Cycle



B11K20-3

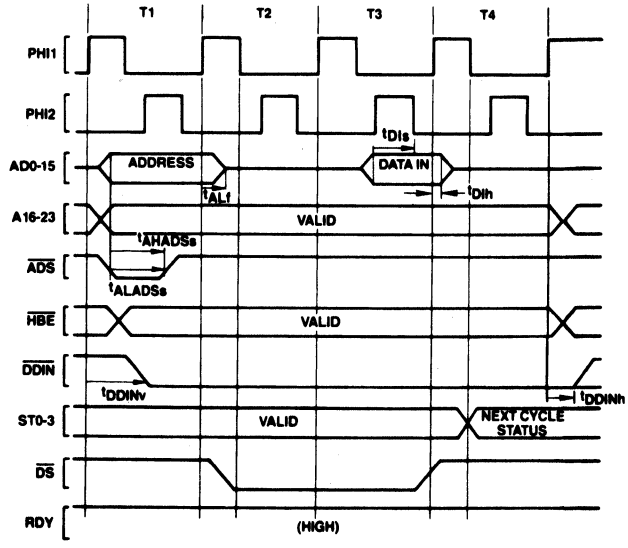
NS32016 Read and Write Cycle Timing

Write Cycle



B11K20-4

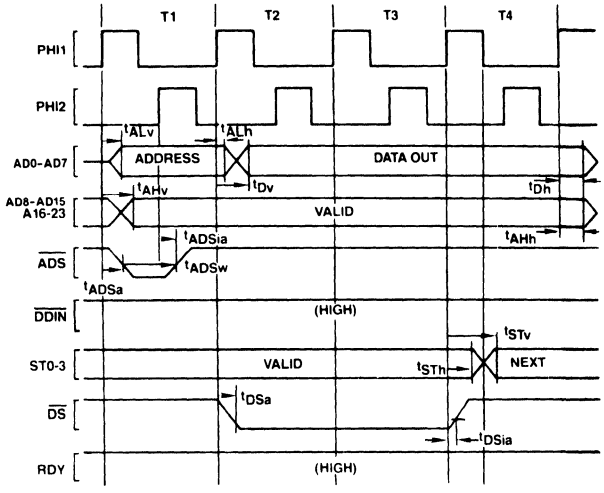
Read Cycle



B11K20-5

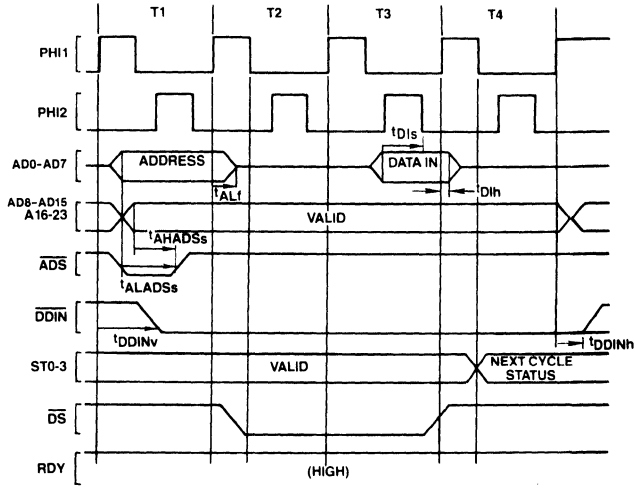
NS32008 Read and Write Cycle Timing

Write Cycle



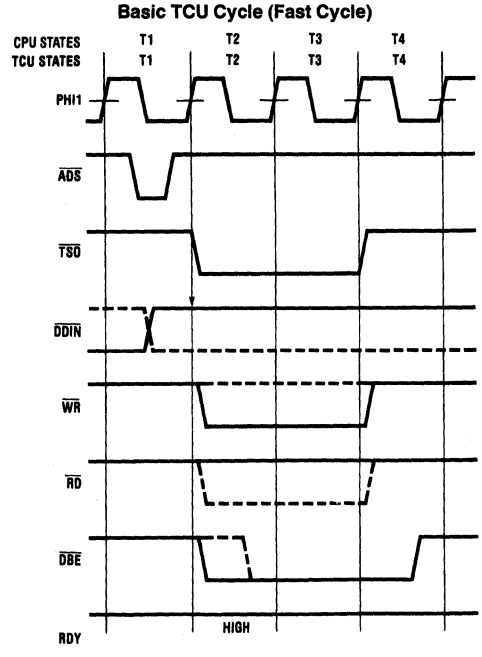
B11K20-6

Read Cycle



B11K20-7

NS32201 Basic TCU Timing



B11K20-8

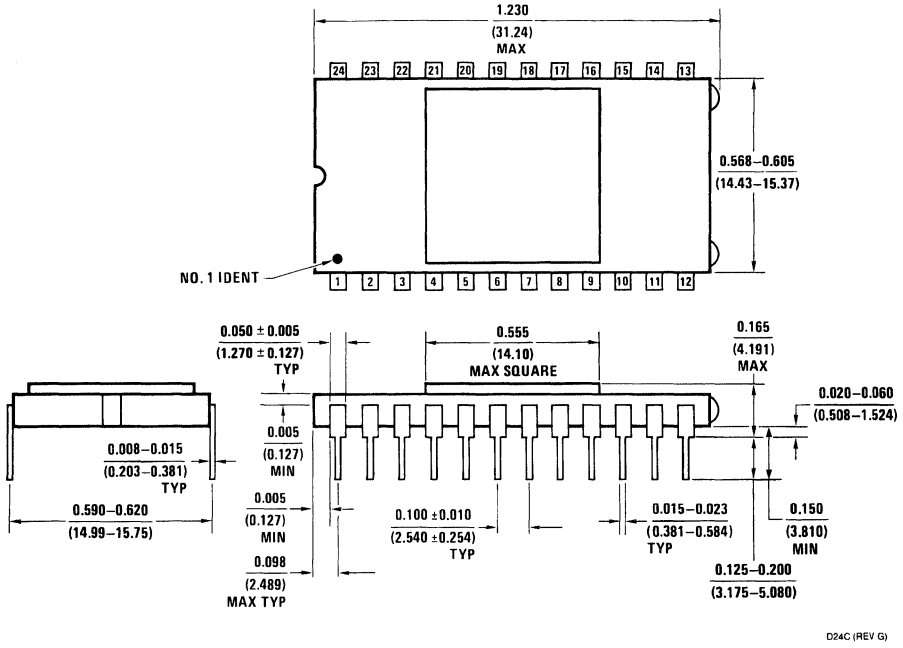
Note 1: The CPU and TCU view some timing states (T-states) differently. For clarity, references to T-states will sometimes be followed by (TCU) or (CPU). (CPU) also implies (MMU).

Note 2: Arrows indicate when the TCU samples the input.

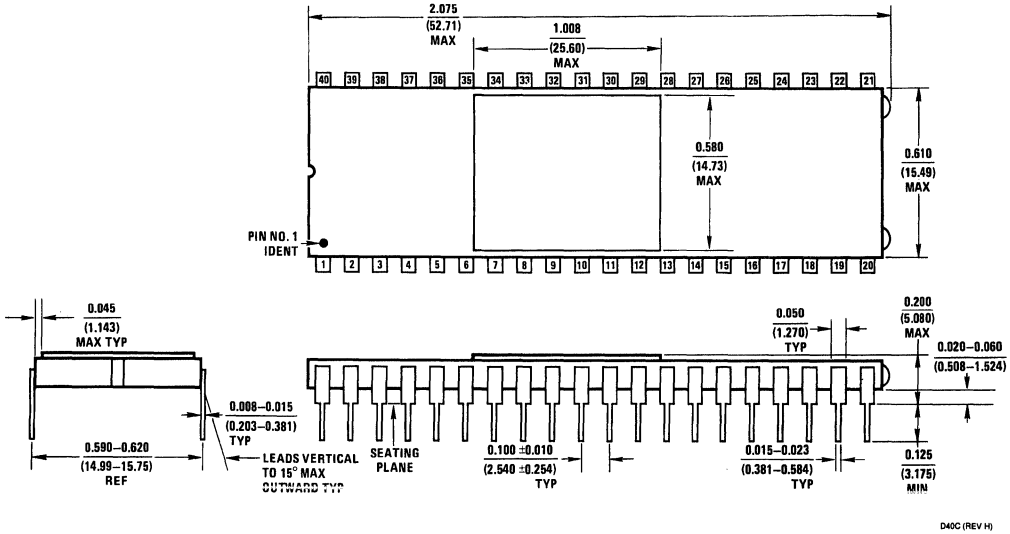
Note 3: $\overline{\text{RWEN}}$ is assumed low ($\overline{\text{RD}}$ and WR enabled) unless specified differently.

Note 4: For clarity, T-states for both the TCU and CPU are shown above the diagrams. (See Note 1.)

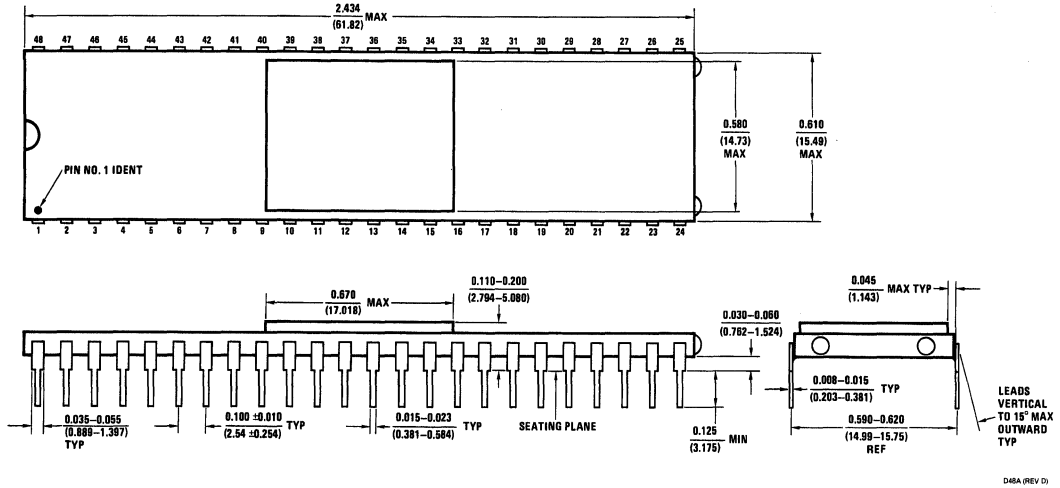
All dimensions are in inches (millimeters)



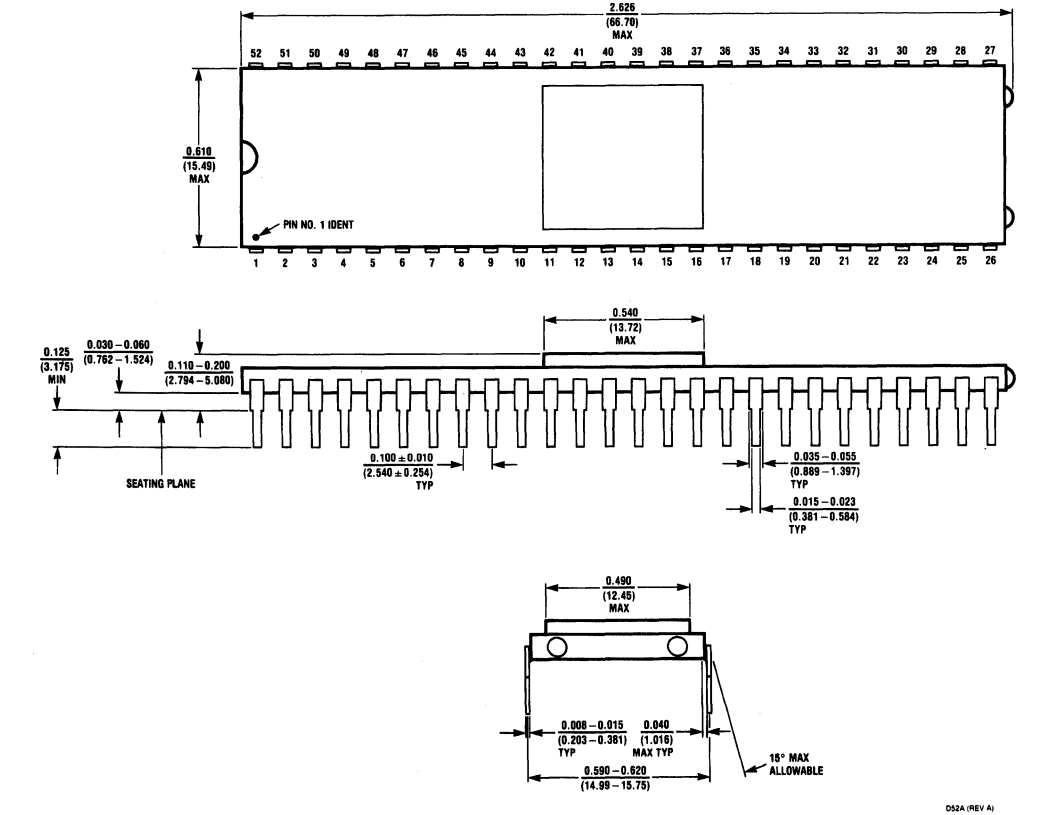
NS Package D24C



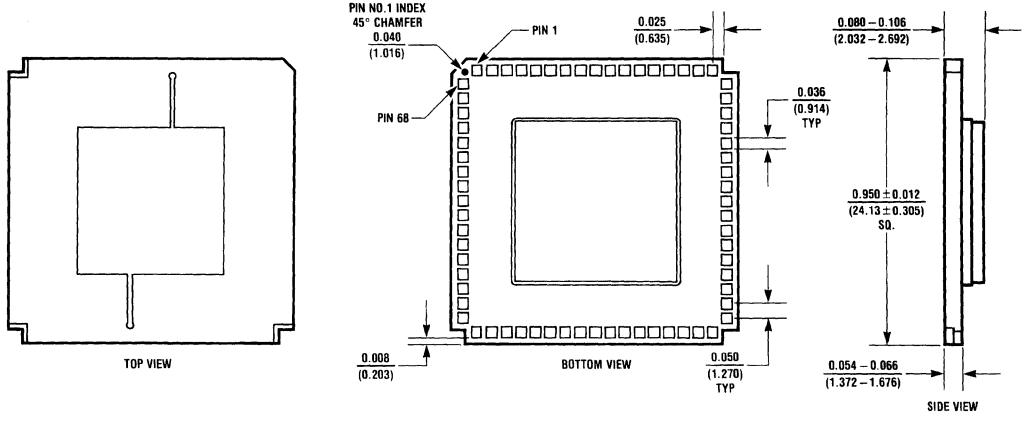
NS Package D40C



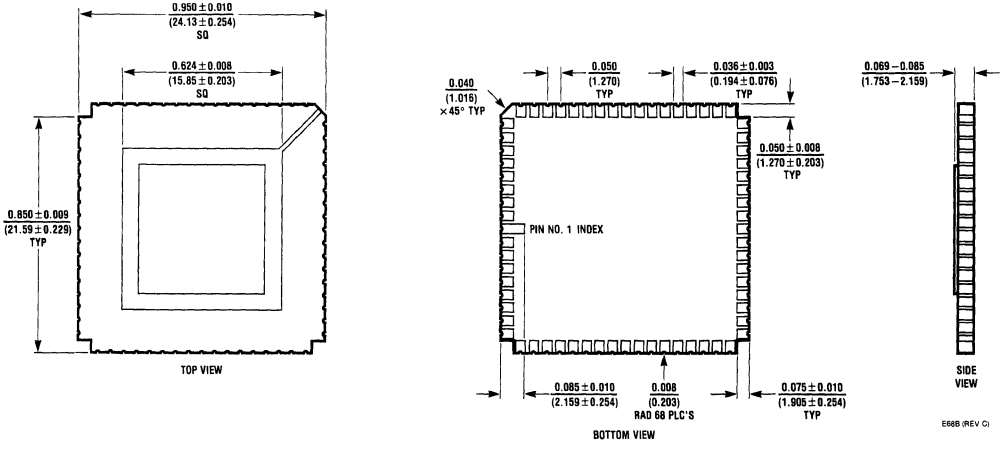
NS Package D48A



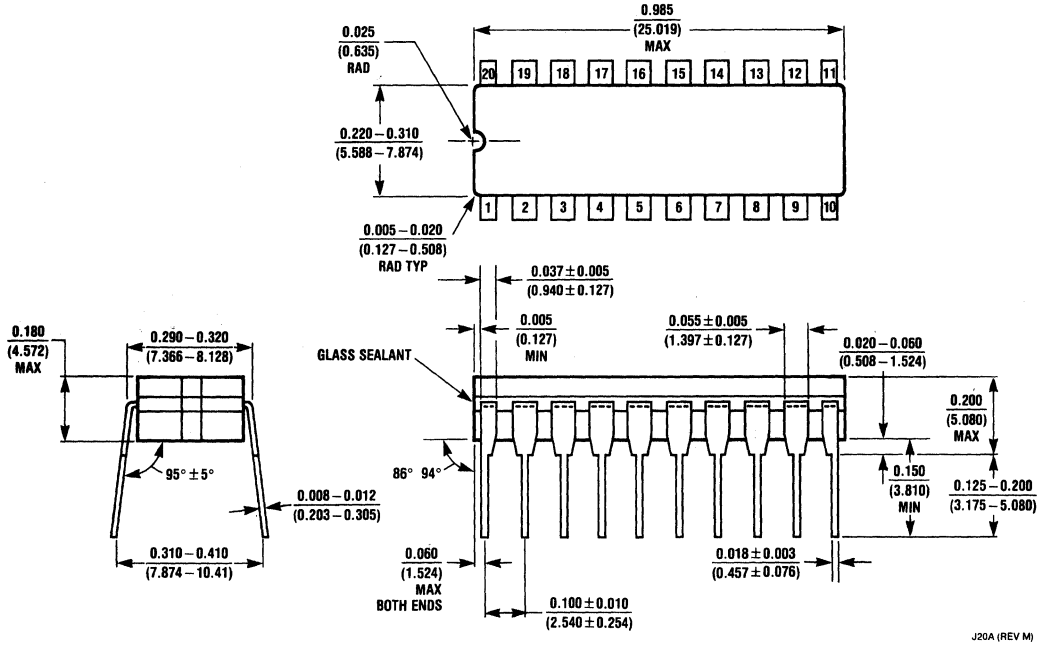
NS Package D52A



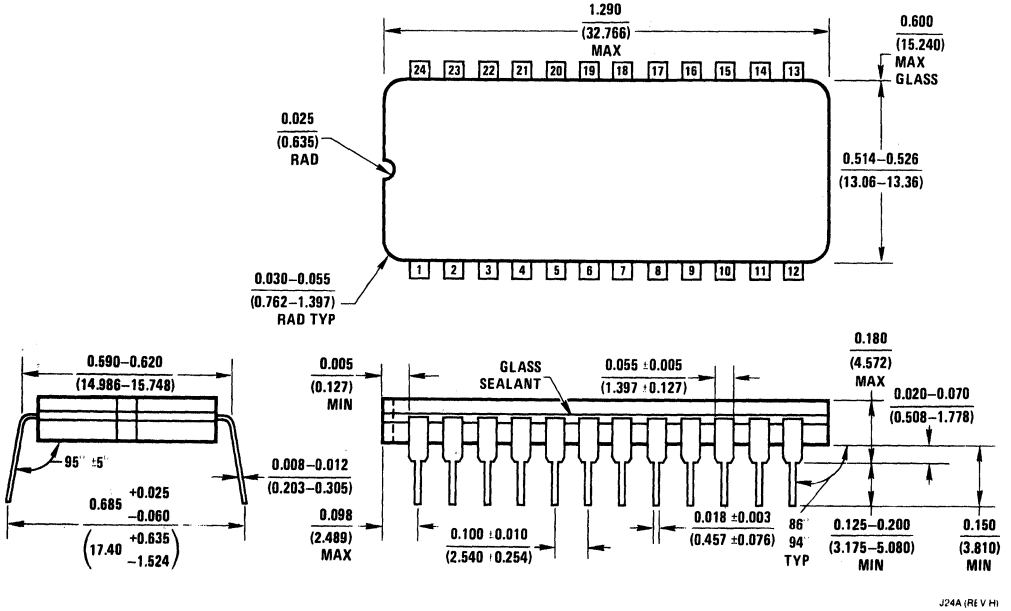
NS Package E68A



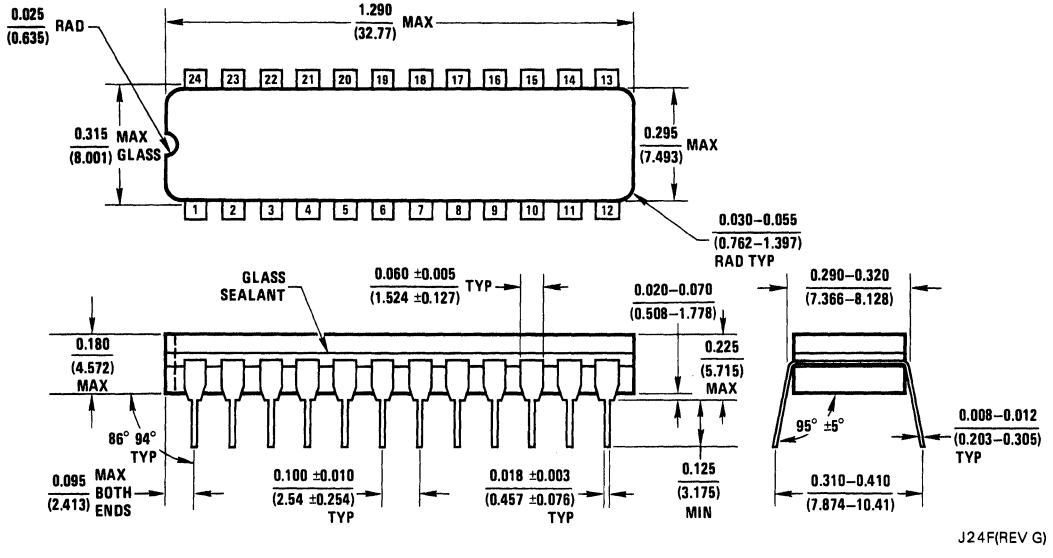
NS Package E68B



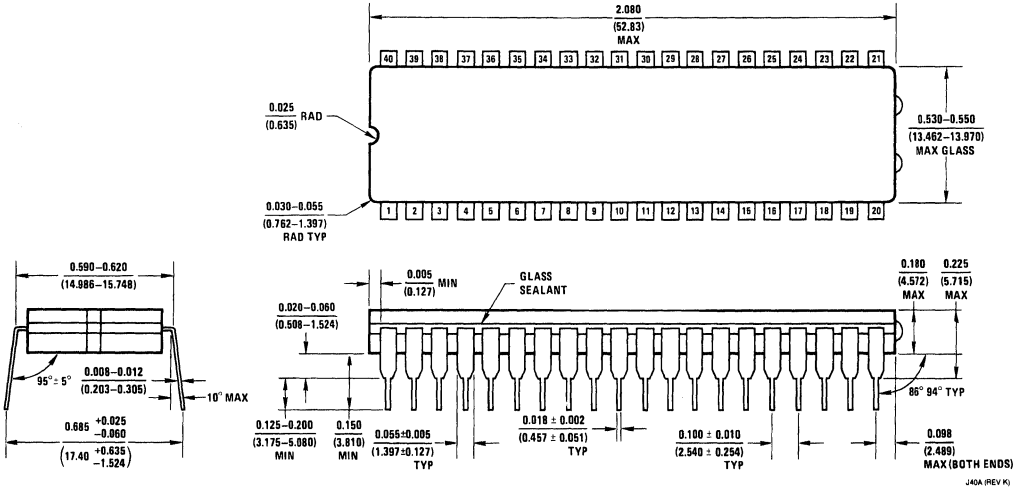
NS Package J20A



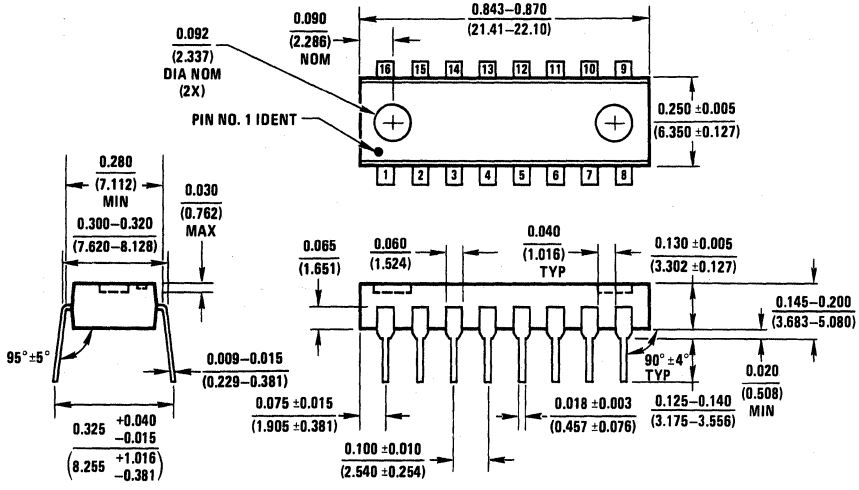
NS Package J24A



NS Package J24F

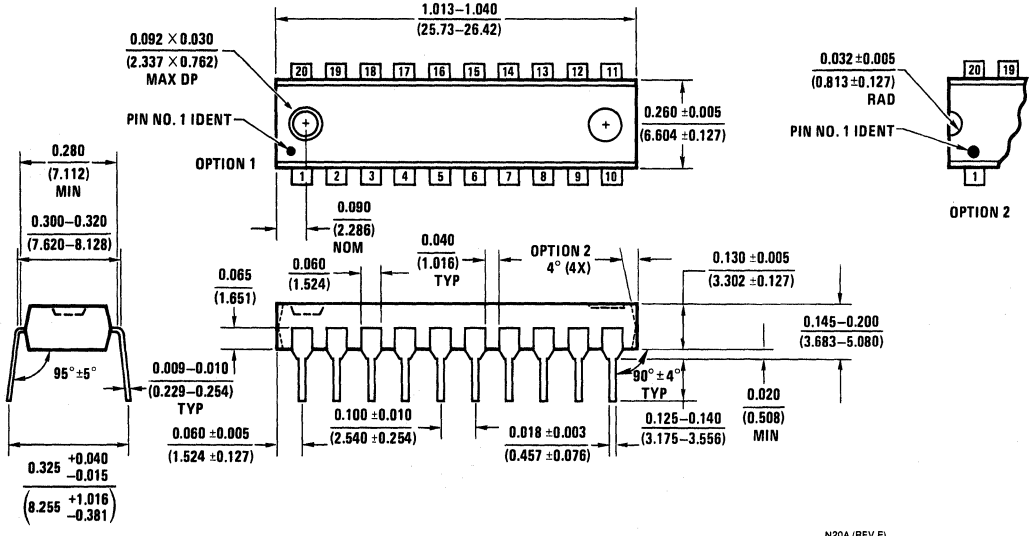


NS Package J40A



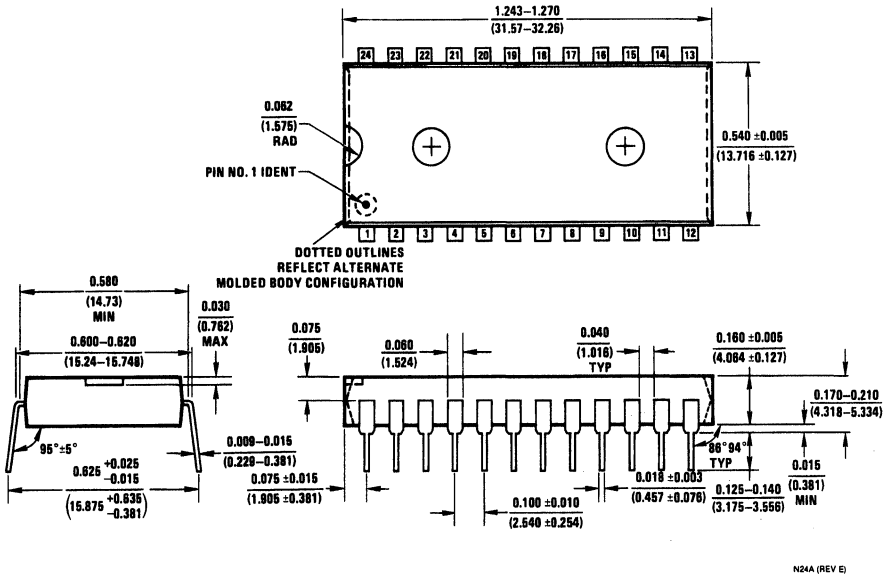
N16A (REV E)

NS Package N16A

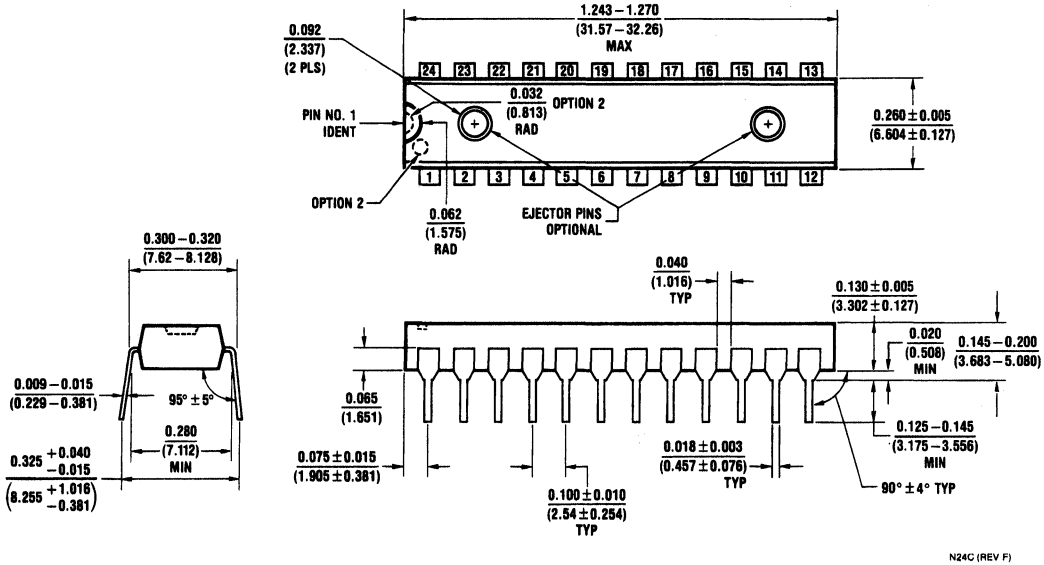


N20A (REV F)

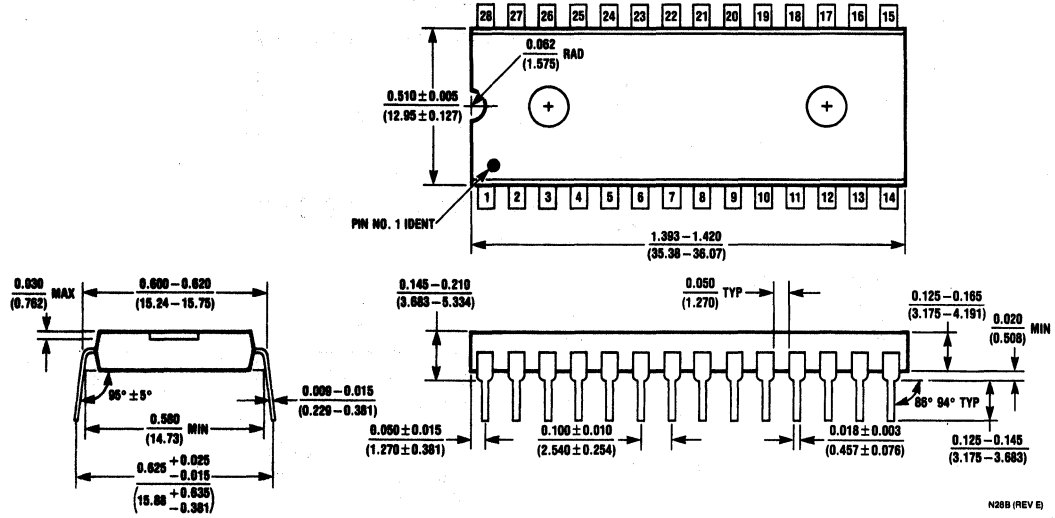
NS Package N20A



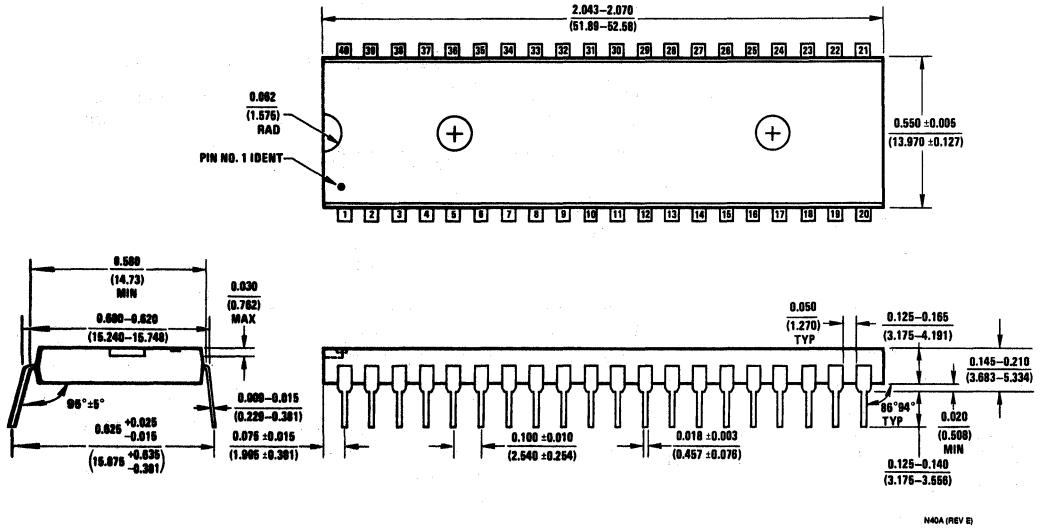
NS Package N24A



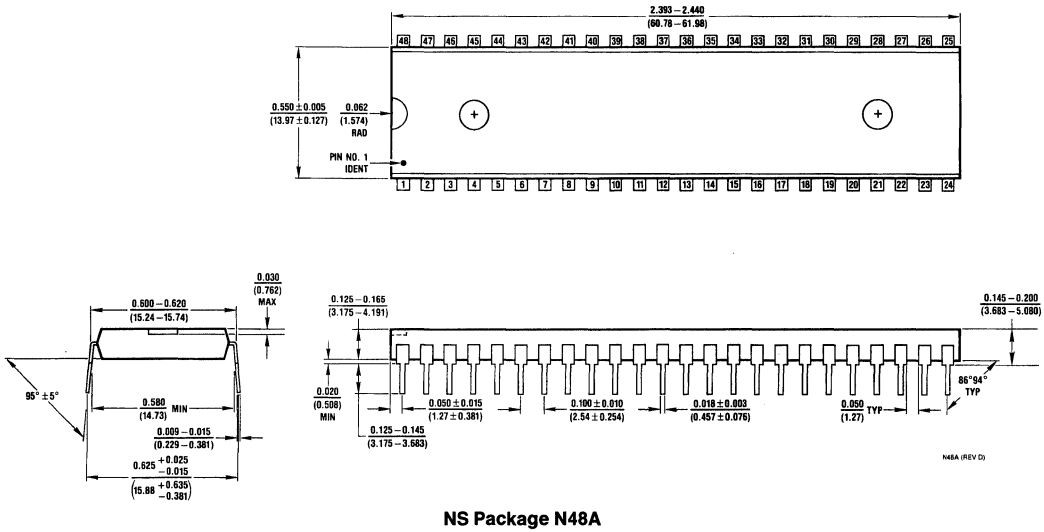
NS Package N24C



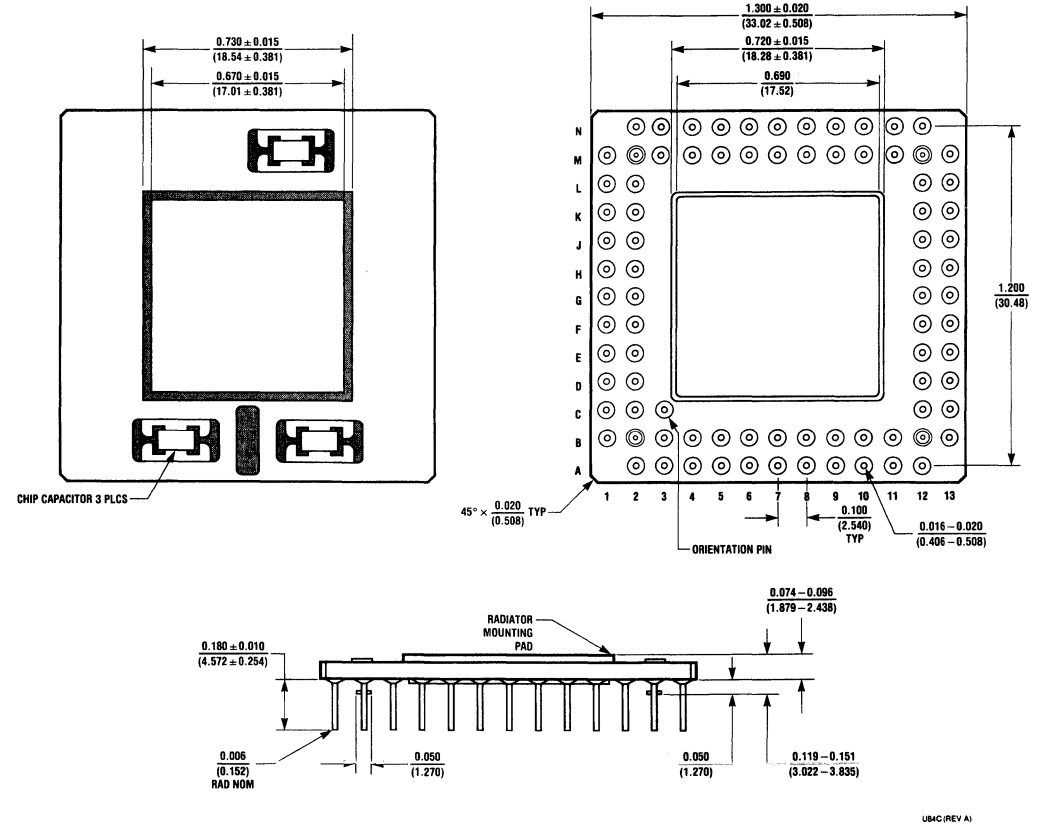
NS Package N28B



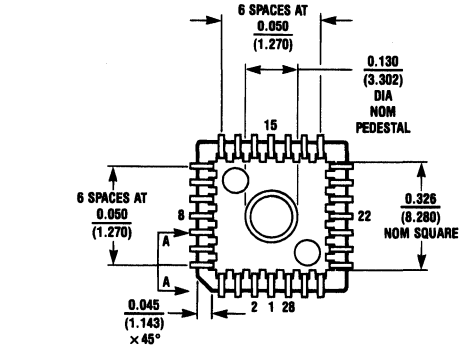
NS Package N40A

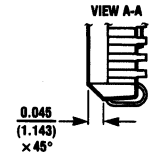


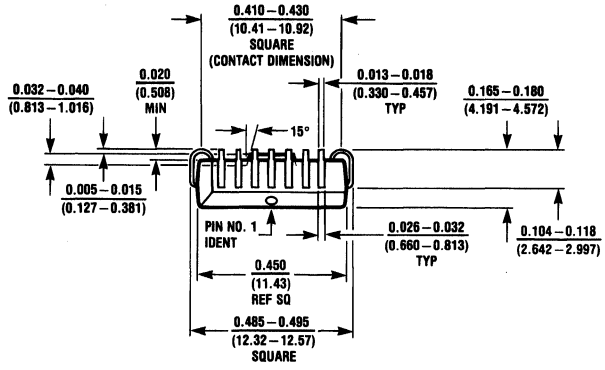
NS Package N48A



NS Package U84C

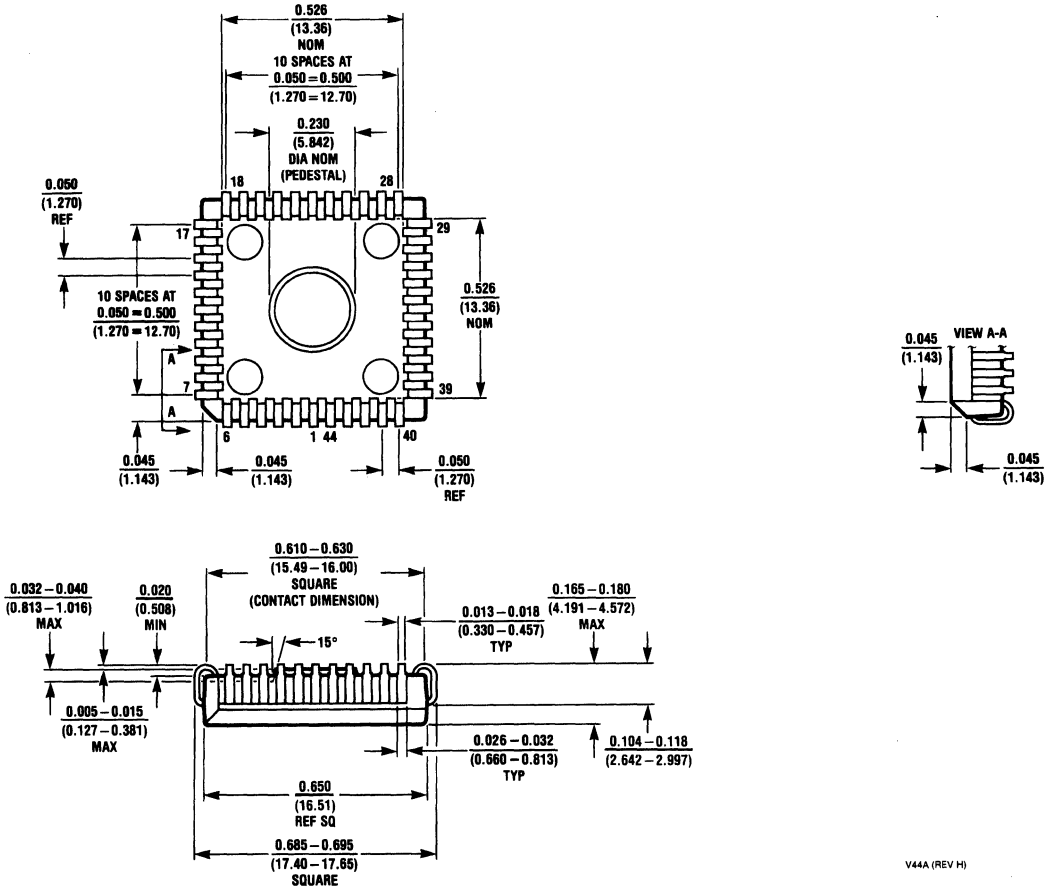




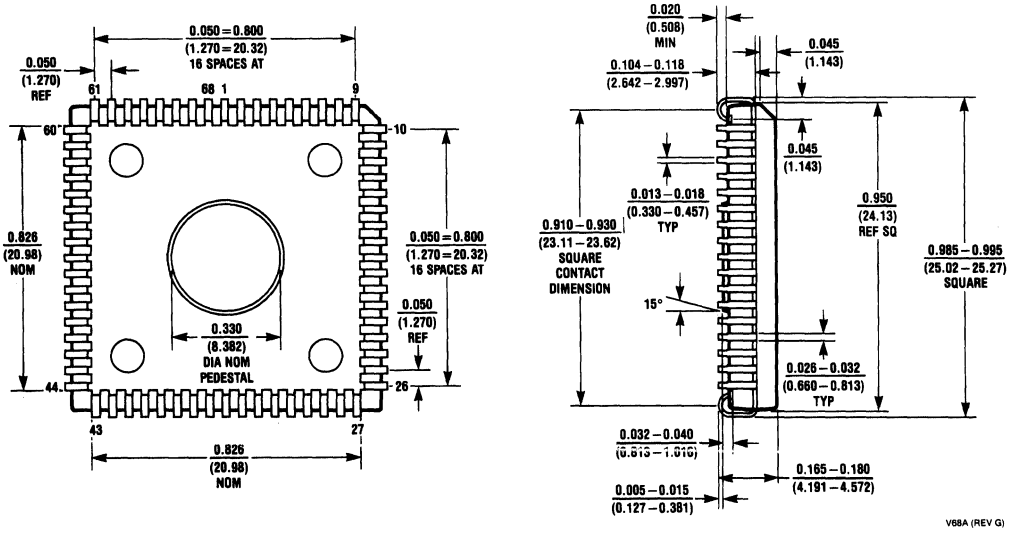


V28A (REV G)

NS Package V28A



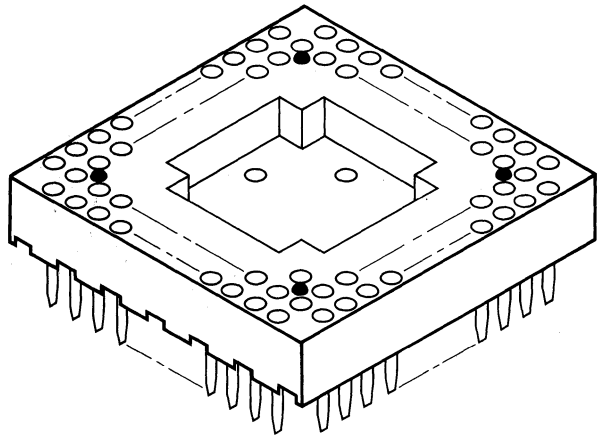
NS Package V44A



NS Package V68A

AMP Sockets for 32332 Microprocessor

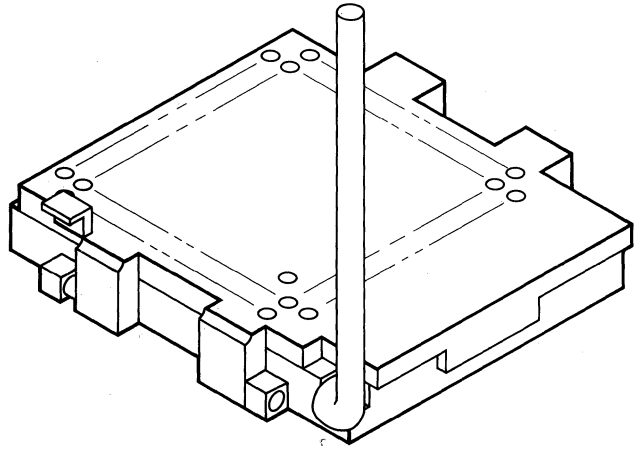
LIF SOCKET



- Low Insertion Force (LIF) Socket 55273-7 AMP tests indicate 50% reduction in insertion force compared to machined socket.

ZIF PRODUCTION SOCKET

Cam handle locks in low profile position when substrate is installed. (Handle UP for open and DOWN for closed)



- Zero Insertion Force (ZIF) Production Socket 55283-7
- Zero Insertion Force (ZIF) Production Burn-In Socket 55383-5
- Not pictured but also available: ZIF High Cycle Test Socket 55269-2 (handle up) 55346-4 (handle down)

AMP INCORPORATED Harrisburg, PA 17105 U.S.A. Phone: 717-564-0100

NOTES

NOTES

National Semiconductor Corporation

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 23-200
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090

For a recorded update of this listing plus ordering information for these books from National's Literature Distribution operation, please call (408) 749-7378.

DATA CONVERSION/ACQUISITION DATABOOK—1984

Selection Guides • Active Filters • Amplifiers • Analog Switches • Analog-to-Digital Converters
Analog-to-Digital Display (DVM) • Digital-to-Analog Converters • Sample and Hold • Sensors/Transducers
Successive Approximation Registers/Comparators • Voltage References

HYBRID PRODUCTS DATABOOK—1982

Operational Amplifiers • Buffers • Instrumentation Amplifiers • Sample & Hold Amplifiers • Comparators
Non-Linear Functions • Precision Voltage Regulators and References • Analog Switches
MOS Clock Drivers • Digital Drivers • A-D Converters • D-A Converters • Fiber-Optic Products
Active Filters & Telecommunication Products • Precision Networks • 883/RETS

INTERFACE/BIPOLAR LSI/BIPOLAR MEMORY/PROGRAMMABLE LOGIC DATABOOK—1983

Transmission Line Drivers/Receivers • Bus Transceivers • Peripheral/Power Drivers
Level Translators/Buffers • Display Controllers/Drivers • Memory Support • Dynamic Memory Support
Microprocessor Support • Data Communications Support • Disk Support • Frequency Synthesis
Interface Appendices • Bipolar PROMs • Bipolar and ECL RAMs • 2900 Family/Bipolar Microprocessor
Programmable Logic

INTUITIVE IC CMOS EVOLUTION—1984

Thomas M. Frederiksen's new book targets some of the most significant transitions in semiconductor technology since the change from germanium to silicon. *Intuitive IC CMOS Evolution* highlights the transition in the reduction in defect densities and the development of new circuit topologies. The author's latest book is a vital aid to engineers, and industry observers who need to stay abreast of the semiconductor industry.

INTUITIVE IC OP AMPS—1984

Thomas M. Frederiksen's new book, *Intuitive IC Op Amps,* explores the many uses and applications of different IC op amps. Frederiksen's detailed book differs from others in the way he focuses on the intuitive groundwork in the basic functioning concepts of the op amp. Mr. Frederiksen's latest book is a vital aid to engineers, designers, and industry observers who need to stay abreast of the computer industry.

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.

LINEAR SUPPLEMENT DATABOOK—1984

Amplifiers • Comparators • Voltage Regulators • Voltage References • Converters • Analog Switches
Sample and Hold • Sensors • Filters • Building Blocks • Motor Controllers • Consumer Circuits
Telecommunications Circuits • Speech • Special Analog Functions

LOGIC DATABOOK VOLUME I—1984

CMOS AC Switching Test Circuits and Timing Waveforms • CMOS Application Notes • MM54HC/MM74HC
MM54HCT/MM74HCT • CD4XXX • MM54CXXX/MM74CXXX • LSI/VLSI

LOGIC DATABOOK VOLUME II—1984

Introduction to Bipolar Logic • Advanced Low Power Schottky • Advanced Schottky • Low Power Schottky
Schottky • TTL • Low Power

MASS STORAGE HANDBOOK—1986

Disk Interface Design Guide and User Manual • Winchester Disk Support • Winchester Disk Data Controller
Floppy Disk Support • Drive Interface Support Circuits

MEMORY SUPPORT HANDBOOK—1986

Dynamic Memory Control • Error Checking and Correction • Microprocessor Interface and Applications
Memory Drivers and Support

MOS MEMORY DATABOOK—1984

Standard Terminology • MOS Memory Cross Reference Guide • Dynamic RAMs • NMOS Static RAMs
CMOS Static RAMs • EPROMs • EEPROMs • Military/Aerospace • Reliability

THE NSC800 MICROPROCESSOR FAMILY DATABOOK—1985

CPU • Peripherals • Evaluation Board • Logic Devices • MA2000 Macrocomponent Family

THE SWITCHED-CAPACITOR FILTER HANDBOOK—1985

Introduction to Filters • National's Switched-Capacitor Filters • Designing with Switched-Capacitor Filters
Application Circuits • Filter Design Program • Nomographs and Tables

TRANSISTOR DATABOOK—1982

NPN Transistors • PNP Transistors • Junction Field Effect Transistors • Selection Guides • Pro Electron Series
Consumer Series • NA/NB/NR Series • Process Characteristics Double-Diffused Epitaxial Transistors
Process Characteristics Power Transistors • Process Characteristics JFETs • JFET Applications Notes

VOLTAGE REGULATOR HANDBOOK—1982

Product Selection Procedures • Heat Flow & Thermal Resistance • Selection of Commercial Heat Sink
Custom Heat Sink Design • Applications Circuits and Descriptive Information • Power Supply Design
Data Sheets

48-SERIES MICROPROCESSOR HANDBOOK—1980

The 48-Series Microcomputers • The 48-Series Single-Chip System • The 48-Series Instruction Set
Expanding the 48-Series Microcomputers • Applications for the 48-Series • Development Support
Analog I/O Components • Communications Components • Digital I/O Components • Memory Components
Peripheral Control Components

NATIONAL SEMICONDUCTOR CORPORATION AUTHORIZED DISTRIBUTORS

ALABAMA
Huntsville
Hall-Mark
(205) 837-8700
Hamilton/Avnet
(205) 837-7210
Pioneer
(205) 837-9300
Schweber
(205) 882-2200

ARIZONA
Phoenix
Schweber
(602) 997-4874
Tempe
Anthem Electronics
(602) 966-6600
Hamilton/Avnet
(602) 231-5100

CALIFORNIA
Canoga Park
Hall-Mark
(818) 716-7300
Schweber
(818) 999-4702
Chatsworth
Anthem Electronics
(818) 700-1000
Avnet Electronics
(818) 700-8668
Hamilton Electro Sales
(818) 700-6050
Citrus Heights
Hall-Mark
(916) 722-8600
Costa Mesa
Avnet Electronics
(714) 754-6050
Hamilton Electro Sales
(714) 641-4159
Culver City
Hamilton Electro Sales
(213) 558-2121
Garden Grove
Bell Industries
(714) 895-7801
Gardena
Bell Industries
(213) 515-1800
Irvine
Anthem Electronics
(714) 768-4444
Schweber
(714) 863-0200
Roseville
Bell Industries
(916) 969-3100
Sacramento
Hamilton/Avnet
(916) 925-2216
Schweber
(916) 929-9732
San Diego
Anthem Electronics
(619) 279-5200
Hamilton/Avnet
(619) 571-7510
San Jose
Anthem Electronics
(408) 946-8000
Hall-Mark
(408) 946-0900
Schweber
(408) 946-7171
Sunnyvale
Bell Industries
(408) 734-8570
Hamilton/Avnet
(408) 743-3355
Thousand Oaks
Bell Industries
(805) 499-6821
Tustin
Hall-Mark
(714) 669-4700

COLORADO
Englewood
Anthem Electronics
(303) 790-4500
Hamilton/Avnet
(303) 779-9998
Schweber
(303) 799-0258
Wheatridge
Bell Industries
(303) 424-1985

CONNECTICUT
Danbury
Hamilton/Avnet
(203) 797-2800
Schweber
(203) 748-7080
Meridian
Lionex Inc.
(203) 237-2282
Norwalk
Pioneer Northeast
(203) 853-1515

FLORIDA
Altamonte Springs
Pioneer
(305) 834-9090
Schweber
(305) 331-7555
Clearwater
Hall-Mark
(813) 530-4543
Deerfield
Pioneer
(305) 428-8877
Ft. Lauderdale
Hamilton/Avnet
(305) 971-2900
Hollywood
Schweber
(305) 927-0511
Orlando
Hall-Mark
(305) 855-4020
Pompano Beach
Hall-Mark
(305) 971-9280
Winter Park
Hamilton-Avnet
(305) 628-3888

GEORGIA
Norcross
Hall-Mark
(404) 447-8000
Hamilton-Avnet
(404) 447-7500
Pioneer
(404) 448-1711
Schweber
(404) 449-9170

ILLINOIS
Bensenville
Hamilton/Avnet
(312) 860-7780
Chicago
Bell Industries
(312) 982-9210
Elk Grove Village
Pioneer
(312) 952-8440
Schweber
(312) 364-3750
Wood Dale
Hall-Mark
(312) 860-3800

INDIANA
Carmel
Hamilton/Avnet
(317) 844-9333
Indianapolis
Advent
(317) 872-4910
Pioneer
(317) 849-7300

IOWA
Cedar Rapids
Advent Electronics
(319) 363-0221
Bell Industries
(319) 395-0730
Hamilton/Avnet
(319) 362-4757
Schweber
(319) 373-1417

KANSAS
Lenexa
Hall-Mark
(913) 888-4747
Overland Park
Hamilton/Avnet
(913) 888-8900
Schweber
(913) 492-2921

MARYLAND
Columbia
Hall-Mark
(301) 988-9800
Hamilton/Avnet
(301) 995-3500
Gaithersburg
Pioneer Washington
(301) 921-0660
Schweber
(301) 840-5900

MASSACHUSETTS
Bedford
Schweber
(617) 275-5100
Billerica
Hall-Mark
(617) 935-9777
Lexington
Pioneer Northeast
(617) 861-9200
Wilmington
Lionex
(617) 657-5170
Woburn
Hamilton/Avnet
(617) 273-7500

MICHIGAN
Grand Rapids
Hamilton/Avnet
(616) 243-8805
R-M Michigan
(616) 531-9300
Livonia
Hamilton/Avnet
(313) 522-4700
Pioneer
(313) 525-1800
Schweber
(313) 525-8100

MINNESOTA
Bloomington
Hall-Mark
(612) 854-3223
Edina
Schweber
(612) 941-5280
Minnetonka
Hamilton/Avnet
(612) 932-0600
Pioneer
(612) 935-5444

MISSOURI
Earth City
Hamilton/Avnet
(314) 344-1200
Schweber
(314) 739-0526
Maryland Heights
Hall-Mark
(314) 291-5350

NEW HAMPSHIRE
Manchester
Hamilton/Avnet
(603) 624-9400
Schweber
(603) 625-2250

NEW JERSEY
Cherry Hill
Hall-Mark
(609) 424-7300
Hamilton/Avnet
(609) 424-0100
Fairfield
Hall-Mark
(201) 575-4415
Hamilton/Avnet
(201) 575-3390
Lionex
(201) 227-7960
Schweber
(201) 227-7880
Pine Brook
Pioneer Northeast
(201) 575-3510

NEW MEXICO
Albuquerque
Alliance Electronics
(505) 292-3360
Bell/Century
(505) 292-2700
Hamilton/Avnet
(505) 765-1500

NATIONAL SEMICONDUCTOR CORPORATION
AUTHORIZED DISTRIBUTORS (Continued)

NEW YORK
Buffalo
Summit Distributors
(716) 887-2800
East Syracuse
Hamilton/Avnet
(315) 437-2642
Fairport
Pioneer
(716) 381-7070
Hauppauge
Hamilton/Avnet
(516) 434-7413
Lionex
(516) 273-1660
Rochester
Hamilton/Avnet
(716) 475-9130
Schweber
(716) 424-2222
Summit Electronics
(716) 334-8110
Ronkonkoma
Hall-Mark
(516) 737-0600
Vestal
Pioneer
(607) 748-8211
Westbury
Hamilton/Avnet
(516) 997-6868
Schweber
(516) 334-7474
Woodbury
Pioneer
(516) 921-8700

NORTH CAROLINA
Charlotte
Pioneer
(704) 527-8188
Raleigh
Hall-Mark
(919) 872-0712
Hamilton/Avnet
(919) 878-0810
Schweber
(919) 876-0000

OHIO
Beechwood
Schweber
(216) 464-2970
Cleveland
Hamilton/Avnet
(216) 831-3500
Pioneer
(216) 587-3600
Dayton
Bell Industries
(513) 435-8660
Hamilton/Avnet
(513) 439-6700
Pioneer
(513) 236-9900
Highland Heights
CAM/OHIO
(216) 461-4700
Solon
Hall-Mark
(216) 349-4632
Westerville
Hall-Mark
(614) 891-4555

OKLAHOMA
Tulsa
Hall-Mark
(918) 665-3200
Quality Components
(918) 664-8812
Radio Inc.
(918) 587-9123
Schweber
(918) 622-8000

OREGON
Beaverton
Almac Electronics
(503) 641-9070
Lake Oswego
Anthem Electronics
(503) 684-2661
Bell Industries
(503) 241-4115
Hamilton/Avnet
(503) 635-7850

PENNSYLVANIA
Horsham
Lionex
(215) 443-5150
Pioneer
(215) 674-4000
Schweber
(215) 441-0600
Pittsburgh
CAM/RPC
(412) 782-3770
Pioneer
(412) 782-2300
Schweber
(412) 782-1600

TEXAS
Addison
Quality Components
(214) 733-4300
Austin
Hall-Mark
(512) 258-8848
Hamilton/Avnet
(512) 837-8911
Pioneer
(512) 835-4000
Quality Components
(512) 835-0220
Schweber
(512) 458-8253
Dallas
Hall-Mark
(214) 553-4300
Pioneer
(214) 386-7300
Schweber
(214) 661-5010
Houston
Hall-Mark
(713) 781-6100
Hamilton/Avnet
(713) 780-1771
Pioneer
(713) 988-5555
Schweber
(713) 784-3600
Irving
Hamilton/Avnet
(214) 659-4151
Sugarland
Quality Components
(713) 491-2255

UTAH
Salt Lake City
Anthem Electronics
(801) 973-8555
Bell Industries
(801) 972-6969
Hamilton/Avnet
(801) 972-4300

WASHINGTON
Bellevue
Hamilton/Avnet
(206) 453-5844
Redmond
Anthem Electronics
(208) 881-0850
Seattle
Almac Electronics
(206) 643-9992

WISCONSIN
Brookfield
Schweber
(414) 784-9020
Milwaukee
Taylor Electric Co.
(414) 241-4321
New Berlin
Hall-Mark
(414) 761-3000
Hamilton/Avnet
(414) 784-4516
Waukesha
Bell Industries
(414) 547-8879

CANADA
Western Provinces
Calgary
Hamilton/Avnet
(403) 230-3586
Zentronics
(403) 272-1021
Edmonton
Zentronics
(403) 463-3014
Richmond
Zentronics
(604) 273-5575
Winnipeg
Zentronics
(204) 775-8661

Eastern Provinces
Brampton
Zentronics
(416) 451-9600
Doval
Semad
(514) 636-4614
Markham
Semad
(416) 475-8500
Missisauga
Hamilton/Avnet
(416) 677-7432
Nepean
Hamilton/Avnet
(613) 226-1700
Ottawa
Semad
(613) 729-6145
Zentronics
(613) 238-6411
St. Laurent
Zentronics
(514) 735-5361
Waterloo
Zentronics
(519) 884-5700

National Semiconductor Corporation

National Semiconductor Corporate Headquarters
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-5000
TWX: (910) 339-9240

SALES OFFICES AND REPRESENTATIVES (Continued)

INTERNATIONAL OFFICES

Electronica NSC de Mexico SA
Juventino Rosas No. 118-2
Col Guadalupe Inn
Mexico, 01020 D.F. Mexico
Tel: (905) 524-9402

National Semicondutores Do Brasil Ltda.
Av. Brig. Faria Lima, 830
8 Andar
01452 Sao Paulo, SP. Brasil
Tel: (55/11) 212-5066
Telex: 391-1131931 NSBR BR

National Semiconductor GmbH
Westendstrasse 193-195
D-8000 Munchen 21
West Germany
Tel: (089) 5 70 95 01
Telex: 522772

National Semiconductor (UK) Ltd.
301 Harpur Centre
Horne Lane
Bedford MK40 1TR
United Kingdom
Tel: 0234-47147
Telex: 826 209

National Semiconductor Benelux
Ave Charles Quint 545
B-1080 Bruxelles
Belgium
Tel: (02) 4661807
Telex: 61007

National Semiconductor (UK) Ltd.
1, Bianco Lunos Alle
DK-1868 Copenhagen V
Denmark
Tel: (01) 213211
Telex: 15179

National Semiconductor
Expansion 10000
28, Rue de la Redoute
F-92 260 Fontenay-aux-Roses
France
Tel: (01) 660-8140
Telex: 250956

National Semiconductor S.p.A.
Via Solferino 19
20121 Milano
Italy
Tel: (02) 345-2046/7/8/9
Telex: 332835

National Semiconductor AB
Box 2016
Stensatravagen 4/11 TR
S-12702 Skarholmen
Sweden
Tel: (08) 970190
Telex: 10731

National Semiconductor
Calle Nunez Morgado 9
(Esc. Dcha. 1-A)
E-Madrid 16
Spain
Tel. (01) 733-2954/733-2958
Telex: 46133

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
CH-8304 Wallisellen-Zurich
Tel: (01) 830-2727
Telex: 59000

National Semiconductor
Pasilanraitio 6C
SF-00240 Helsinki 24
Finland
Tel: (90) 14 03 44
Telex: 124854

NS Japan Ltd.
4-403 Ikebukuro, Toshima-ku
Tokyo 171, Japan
Tel: (03) 988-2131
Fax: 011-81-3-988-1700

National Semiconductor Hong Kong Ltd.
Southeast Asia Marketing
Austin Tower, 4th Floor
22-26A Austin Avenue
Tsimshatsui, Kowloon, H.K.
Tel: 3-7231290
Cable: NSSEAMKTG
Telex: 52996 NSSEA HX

National Semiconductor (Australia) PTY, Ltd.
21/3 High Street
Bayswater, Victoria 3153
Australia
Tel: (03) 729-6333
Telex: AA32096

National Semiconductor (PTE), Ltd.
51 Goldhill Plaza, No. 10-01
Newton Road
Singapore 1130
Tel: 2506884
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: (02) 501-7227
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (Far East) Ltd.
Korea Office
Third Floor, Hankyung Bldg.
4-25 Hannam-Dong
Yongsan-Ku, Seoul 140, Korea
Tel: 797-8001/3
Telex: K24942 NSRK

 TL1911B RRD-RRD50M086/Printed in U.S.A.