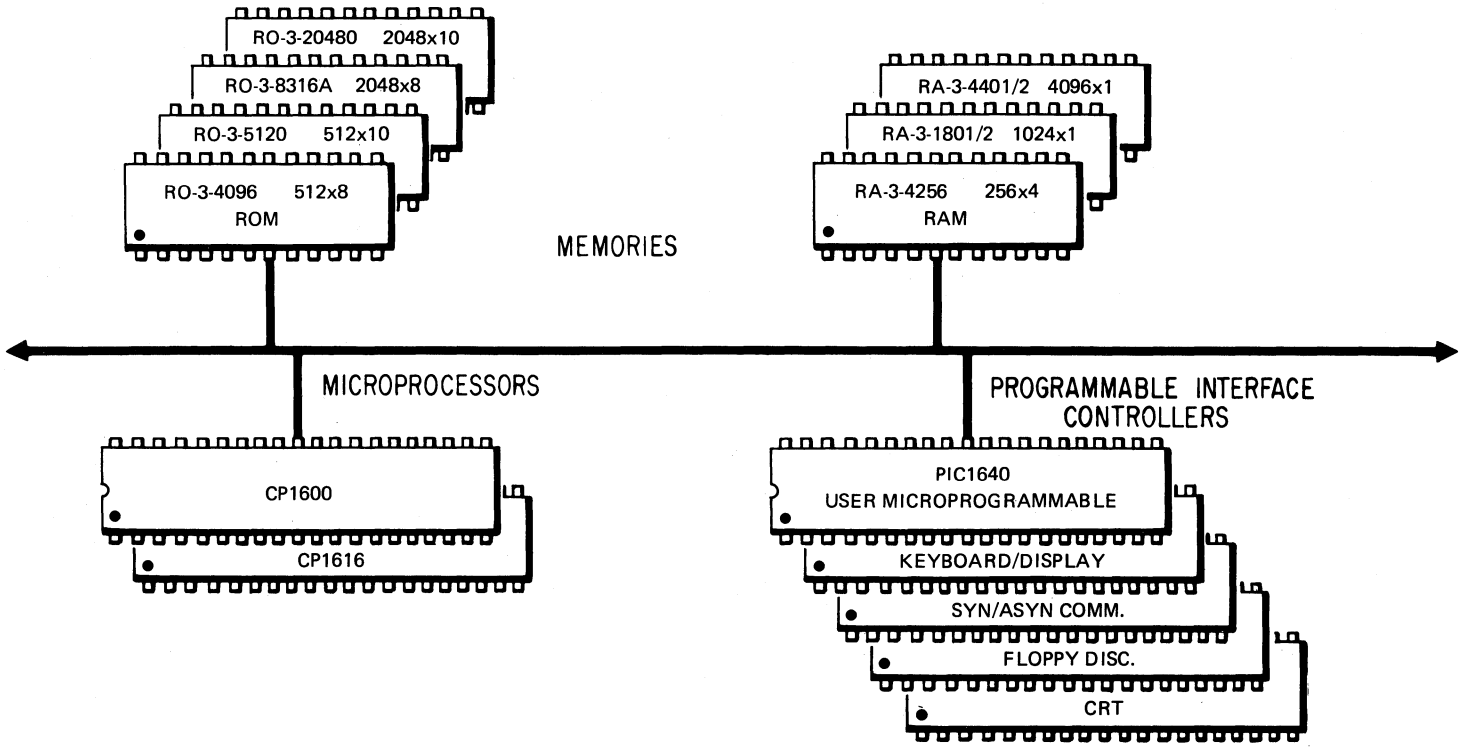
PRICE $8.00

SERIES 1600
MICROPROCESSOR SYSTEM

GENERAL INSTRUMENT
MICROELECTRONICS

MICROCOMPUTER DOCUMENTATION

THE SERIES 1600 SEMICONDUCTOR LINEUP



SERIES 1600 MICROPROCESSOR SYSTEM
MICROCOMPUTER DOCUMENTATION

GIC1600 SERIES MICROCOMPUTER USERS MANUAL

MICRO MODULES

DEVELOPMENT SYSTEMS

RESIDENT FIRMWARE

ON-LINE SOFTWARE

This manual contains a detailed description of the Series 1600 Microcomputer Modules, Prototype Development Systems and all associated hardware. Also included is a functional specification and operational information on the Resident Operating System, Debug Facilities, and On-Line Software for use on the GIC1600 Series Microcomputer Systems.

GENERAL INSTRUMENT CORPORATION ■ MICROELECTRONICS DIVISION ■ 600 W. John St., Hicksville, N.Y. 11802 ■ 516-733-3107

SUMMARY

Title:	GIC1600 Series Microcomputer Users Manual
Document No:	S16DOC-GIC1600-02 September 1975
Revision Level:	Supersedes S16DOC-GIC1600-01 May 1975
Scope:	This manual describes the GIC1600 series of microcomputer systems including details of each module in the system. It includes information on the theory of operation of the microcomputer, timing of all buses and control signals, and logic diagrams of all printed circuit modules. In addition, the ROM Resident Operating System features are detailed along with specific user operational information and command structures. Lastly, all the necessary On-Line Software to make the GIC1600 systems complete hardware and software development tools are described along with typical examples of its use.
Reference Documents:	S16DOC-CP1600-04 CP1600 Microprocessor Users Manual
	S16DOC-XALSIM-02 Series 1600 Cross Software Manual

TABLE OF CONTENTS

CHAPTER 1: GIC1600 SERIES MICROCOMPUTER SYSTEMS - HARDWARE

TABLE OF CONTENTS

CHAPTER 2: GIC1600 MICROCOMPUTER SYSTEM - SOFTWARE

TABLE OF CONTENTS

APPENDICES (continued) GIC1600 MICROCOMPUTER SYSTEM

CHAPTER 1

GIC1600 SERIES MICROCOMPUTER USERS MANUAL

1.0 INTRODUCTION

The GIC1600 Series Microcomputer Systems are general-purpose, stand-alone microcomputers built from the General Instrument family of OEM card level computer components. They are complete, self-contained development tools for both hardware and software prototyping and debugging. In addition, all the individual cards are fully-functional, stand-alone computer modules designed for easy integration into other systems.

The GIC1600 Series Microcomputer Systems are high performance 16-bit computers featuring direct addressing to 65K 16-bit words, a push down stack of unlimited depth, direct memory access (DMA), and a versatile nested interrupt system with priority resolution and self-identifying vectors. All control signals, data, and address buses are fully buffered and available on the backplane so the user can expand memory, add I/O capability, or develop and debug custom interfaces as required.

The GIC1600 Series Microcomputer Systems consist of a Central Processor Card, one or more RAM Memory Cards, an Input/Output Card, a Control Panel Card and Operators Console, and a Rack-Mountable Chassis. Additional cards may be added as required to expand the system capabilities and memory capacity.

In its standard configuration, each system can support a Teletype, a high speed paper tape reader/punch, and any RS232C compatible device, such as a Silent 700 Data Terminal. Additional interfaces and drive software as well as general purpose support cards are continually being added to the family. At present, a 4K PROM Card, additional 2K and 8K RAM Cards, and a General Purpose I/O Card are available as optional accessories.

The Rack-Mountable Chassis and Printed Circuit Backplane has 13 positions for Wire-Wrap or Printed Circuit Cards, of which nine are available to the user for expansion of memory or special Input/Output interface. A general purpose Wire-Wrap Card is available so the user can easily design and construct his own custom interfaces and plug directly into any of the nine available positions in the chassis. Extender cards are also available to provide live access to the circuitry.

The GIC1600 Series Microcomputer System hardware and software provide the user with a versatile microcomputer system that is simple to configure to his individual application.

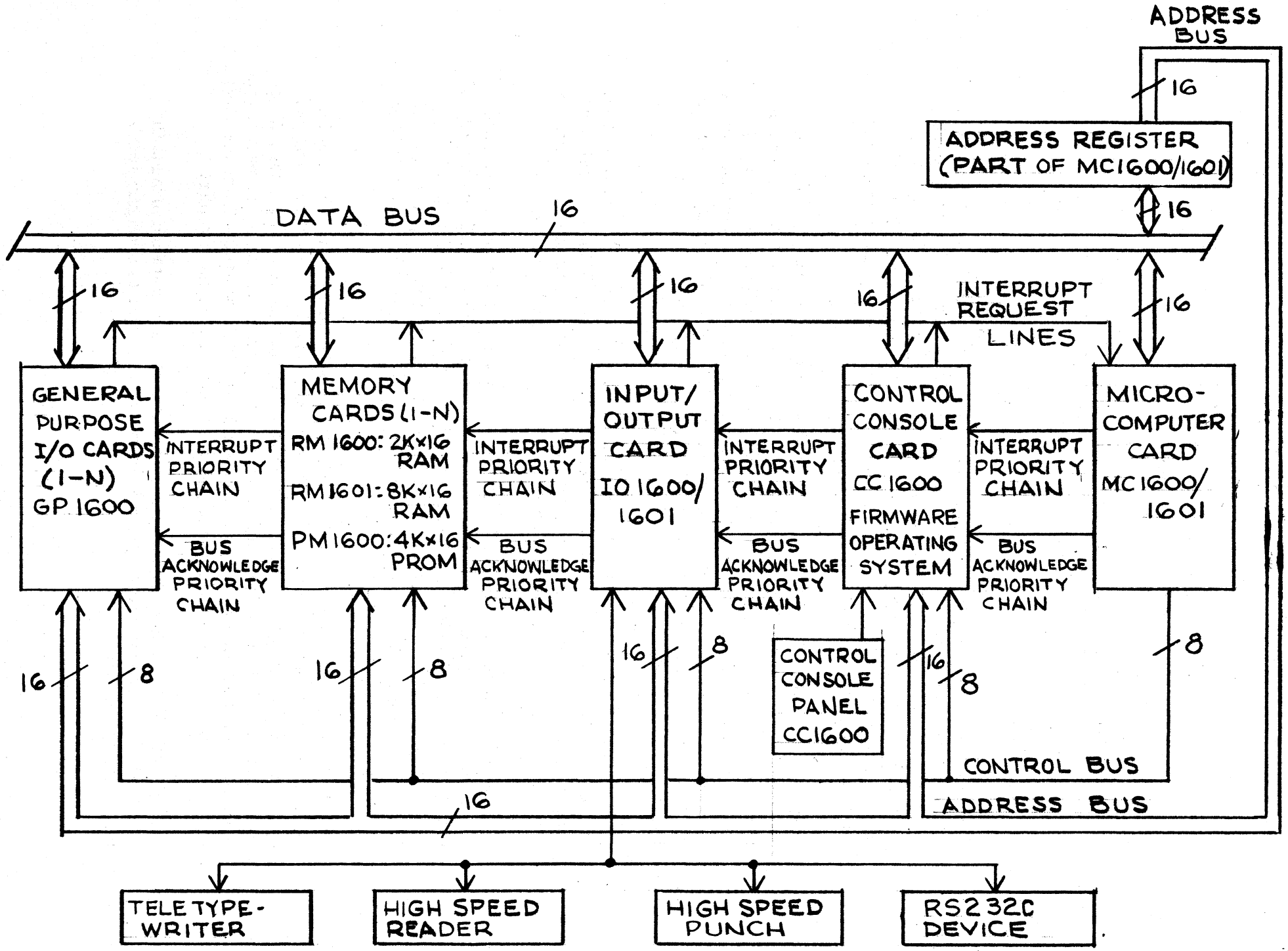


FIG. 1 SYSTEM BLOCK DIAGRAM.

1.1 SYSTEM DESCRIPTION

The GIC1600 Series Microcomputer Systems are supplied in three standard configurations. The GIC1600 includes a basic Microcomputer Module (MC1600), an 8K x 16 RAM Module (RM1601), a Control Panel Module and Operators Console (CC1600), an Input/Output Module (I/O1600), and a Chassis and Backplane Unit (CF1600). The GIC1601 is similar but includes a Microcomputer Module (MC1601) with Real Time Clock (RTC) and Power Fail Interrupt (PFI), and an Input/Output Module with RS232C capability (I/O1601).

1.1.1 System Block Diagram

The basic system block diagram is shown in Figure 1. All microcomputer system components and peripherals connect to and communicate with each other on a common Data Bus. This bidirectional bus allows any device to send data to, receive data from, and exchange data with the central processor or any other device.

The Address Bus Register (part of the Microcomputer Module) captures and latches the address information from the Data Bus under the direction of the Microprocessor. The Address Bus lines are presented staticly to every module in the system for address decoding. The Microprocessor controls the time allocation of both the Data Bus and Address Bus via the Control Bus. The Control Bus contains eight control signals that direct all bus operations.

Any card in the system can issue an interrupt request to get Microcomputer service or a DMA request to gain access to the bus structure. The Microcomputer card acknowledges the request by issuing priority signals that are serially daisy-chained down the cards, establishing a priority assignment based on "electrical closeness" to the Microcomputer Module. The Microprocessor automatically resolves any simultaneity of requests by acknowledging DMA requests before interrupt requests.

1.1.2 Basic System Modules

The MC1600 Microprocessor Module contains the CP1600 Microprocessor integrated circuit, the clock generator and crystal controlled oscillator, the Address Bus Register and buffers, the Data Bus transceivers, and the Control Bus Decoder/Driver. The MC1601 Microprocessor Module includes these items plus the Real Time Clock and Power Fail Interrupt options.

The CC1600 Control Console Module and Operators Panel monitors and directs all operations of the GIC1600 systems. A ROM resident program located on this card and occupying the top 4K of the 64K address space allows the user to perform an extensive set of front panel operations

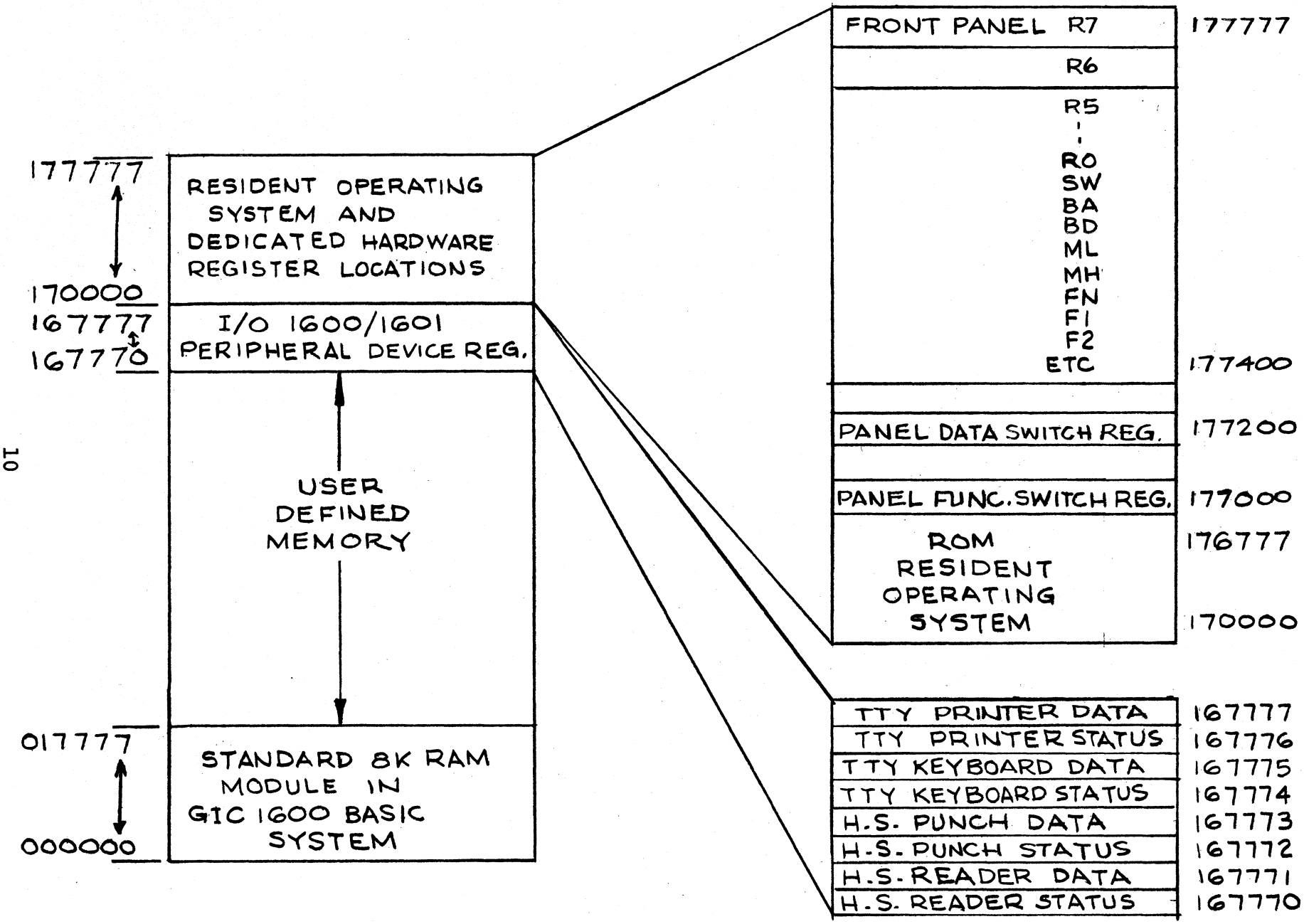


FIG. 2. GIC 1600 SYSTEM MEMORY MAP.

such as read and punch tapes, display and modify memory and CPU registers, and execute programs, etc. In addition, the continuous or single instruct mode can be selected on the Operators Panel along with a unique "repeat instruction" function. Additional controls facilitating program debugging and hardware checkout are also provided on the Control Panel Console. In addition to all these manually controlled Operators Panel functions, the ROM Resident Operating System and Monitor Program provide total interactive control of the system via the Teletype. Beside the normal display and modifying of memory and CPU registers, the user can control the peripherals, load, copy, and punch tape, and set up to eight breakpoints in the active program.

The I/O1600 Input/Output Module provides full-duplex communication between the CPU and a Teletype and/or a High Speed Reader/Punch combination. The I/O1601 Input/Output Module provides the same capabilities plus the interface to any RS232C device such as a Silent 700 Data Terminal. The Silent 700 Terminal provides magnetic tape cassette offline storage and keyboard/printer capability which is a great convenience in program loading. Both I/O Modules contain fully character buffered controllers with all device timing and interrupt logic included.

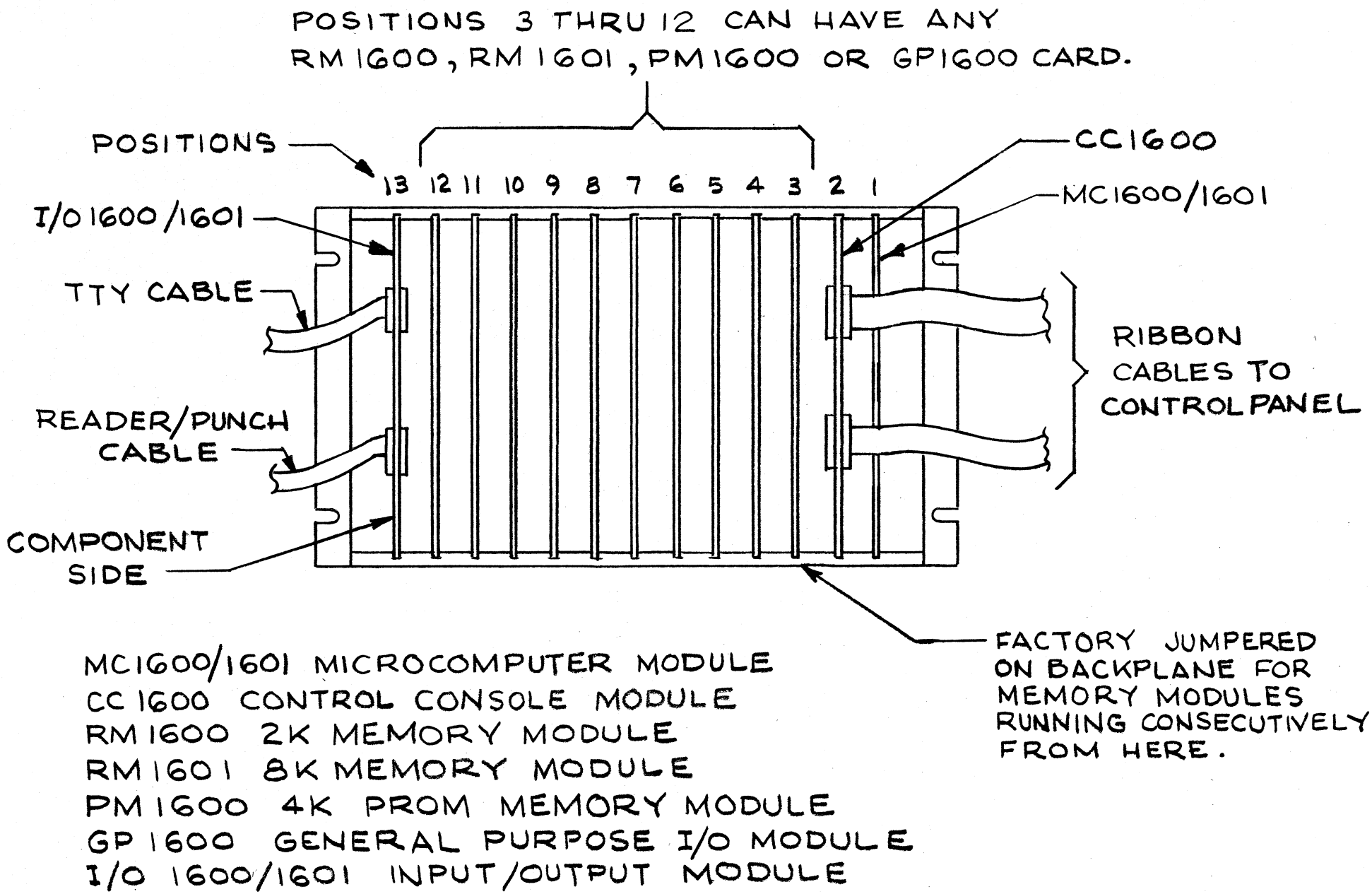
The RM1600 RAM Memory Module contains 2048 words of 16-bit fully buffered RAM memory. The RM1601 Memory Module contains 8196 words of 16-bit RAM memory. Both include user-programmable Module Address Identification Logic so that several cards can be used together to build the memory to any desired capacity up to 65K words total.

The PM1600 PROM Memory Module and the GP1600 General Purpose Input/Output Module are available as additional options to the basic systems and are discussed in subsequent sections of this manual.

1.1.3 System Memory Map

The memory map shown in Fig. 2 defines the allocation of the 65K memory space for the GIC1600 Series Microcomputer Systems. All but the upper 4K is user-defined.

The upper 4K of memory is reserved for the Resident Firmware Operating System, which is stored in ROM's on the Control Console Card. This set of programs start at address 170000 and are approximately 3K words long. Address location 177000 is used to access the Control Console Function Switches; address location 177200 is used to access the Control Console Data Switches. Memory locations 177400 to 177777 are RAM locations used for dynamic storage by the Resident Firmware Operating System.



MC1600/1601 MICROCOMPUTER MODULE
CC 1600 CONTROL CONSOLE MODULE
RM 1600 2K MEMORY MODULE
RM 1601 8K MEMORY MODULE
PM 1600 4K PROM MEMORY MODULE
GP 1600 GENERAL PURPOSE I/O MODULE
I/O 1600/1601 INPUT/OUTPUT MODULE

FIG. 3. GIC 1600 MICROCOMPUTER CARD ASSIGNMENTS.

1.2 SYSTEM SET-UP PROCEDURE

a) Unpack system components and install Modules in the appropriate slots with the component side to the left as shown in Fig. 3. The system is factory wired via backplane jumpers for memory modules (RM1600 or RM1601) to occupy consecutive card positions starting from position 3 which always contains memory location zero.

b) Connect power supply voltage and sense leads to the appropriately labeled terminals on the backplane of the microcomputer. The supplies should have the following current ratings:

+5V at 12A, ±5%; +12V at 1.5A, ±5%; -12V at 0.5A, ±5%

The supplies should have 0.1% load and line regulation and external sense capability. Use 16 AWG size wire for the +5V and +5V return leads.

c) Establish the starting address of the Interrupt Branch Table by connecting J13 (STADØ*) and J48 (STAD1*) of the MC1600/1601 card position to the desired Data Bus lines; i.e., to establish 3000 as the start of the interrupt table connect J13 to DBL9* and J48 to DBL10* respectively. See Section 1.4.3 for further details.

d) Connect the Teletype cable and the Reader/Punch cable to the I/O1600/1601 Input/Output Module in card position 13 as described in Section 1.5.1 and 1.5.2. The baud rate strap on the I/O1600/1601 card is factory wired for 110 baud. Modify the Teletype as explained in Section 1.5.1.

e) Connect the Control Console Module cables to the Control Panel (the top connector on the card should be cabled to the right connector when facing the rear of the Control Panel).

f) Turn on the power supplies. Push the MCLR switch and then the START/STOP switch. The Teletype will respond by typing: S16ODM V01B. The microcomputer system is now running and the Resident Operating System is awaiting a user command. (See Section 2.1.1).

1.3 CONTROL CONSOLE

The GIC1600 Control Console is designed to provide a convenient method of controlling and monitoring the system. It is connected to the system via two cables that plug into the Control Console Interface Card (CC1600). Indicator lamps display the bus operation during the continuous run mode and display the contents of the registers while the CPU is halted. Various function and selection switches are also provided to allow complete control of the prototype system from the Operators Console. The GIC1600 Control Console is shown in Fig. 4.

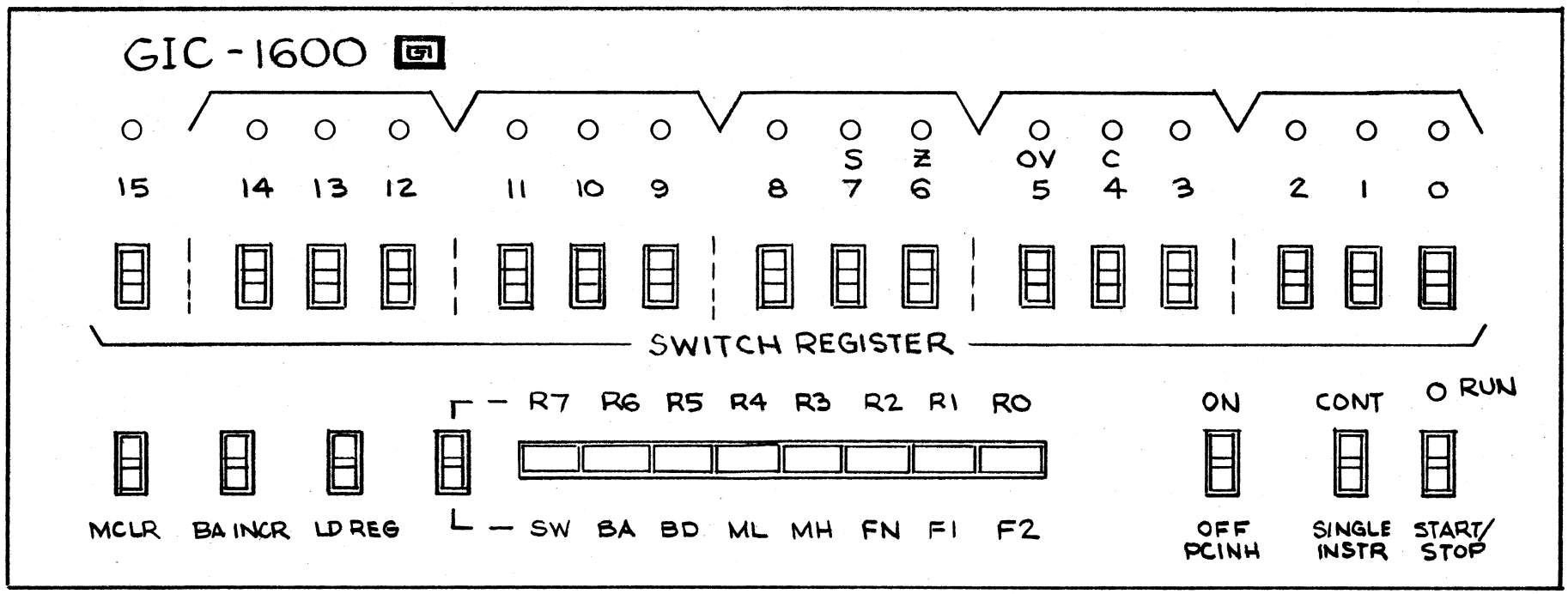


FIG. 4. CONTROL PANEL

1.3.1 Control Console Description

The console has the following indicators and switches:

1) A RUN indicator light
2) A 16-bit Data Register Display
3) A 16-bit Switch Register
4) A set of 8 interlocked Register Select Switches to indicate one of 8 registers to be displayed.
5) A Register Bank Select Switch to indicate one of two banks of 8 registers each to be applied to the Register Select Switches.
6) Control Switches:
   a) LD REG: Load value set in Switch Register into selected register
   b) BA INC: Increment BA Register and update BD Register to reflect contents of new BA address
   c) MCLR: Master Clear
   d) PCINH: Inhibit incrementing of program counter during fetch phase to enable repetitive execution of a one word instruction
   e) CONT - SINGLE INSTR: Mode of operation - continuous or single instruction
   f) START/STOP: Toggle control for starting and stopping CPU

When the system is running a program, BAINC & LD REG are disabled to prevent any disruption of the operation. The PCINH switch should not be operated while the microcomputer is running.

1.3.2 Control Console Functions

The console contains a 16-bit Switch Register that is capable of referencing a 16-bit address (i.e., 65K of memory space). A switch in the up position is considered to have a 1 value. A switch in the down position is considered to have a 0 value. The contents of the Switch Register can be loaded into any register by selecting that register on the Register Select and Bank Select Switches; it can be any one of the eight internal registers, R0 to R7, of the CPU or the SW, BA, BD, ML, MH, FN, F1 & F2 registers described below. After depressing LD REG, the contents of the selected register will be modified and the 16-bit Display Register will change to reflect the modification.

The lower bank of 8 front panel accessible registers contains a number of important and useful functions for the user. The Status Word (SW) of

the CPU contains four bits; they are located in bits 4 to 7 of a 16-bit word to correspond to their position in the CP1600 Microprocessor word. They are arranged as follows: Carry (C) in Bit 4, Overflow (OV) in Bit 5, Zero (Z) in Bit 6, Sign (S) in Bit 7. The remaining bits will always read 0 and are "don't care" when loading the SW from the front panel.

In order to provide the capability of accessing memory, two registers BA and BD have been assigned. Register BA contains a Bus Address and the contents of that bus location is displayed in the Bus Data Register BD. In order to examine a memory location, the address is first keyed into the Switch Register and BA is selected on the Register Select and Bank Select Switches. Pressing LD REG will then deposit the contents of the Switch Register into Register BA. The contents of the selected location (the address now held in BA) will automatically be loaded into Register BD. By selecting BD on the Register Select Switches, the contents of the memory address just loaded into BA will appear in the Display Register. Sequential memory locations can then be examined by depressing BAINC while BD is selected.

In order to modify any bus (memory) location the Bus Address must first be loaded into BA. Then BD is selected and the contents of that location will appear on the Data Display. The new data to be deposited is then keyed into the Switch Register. By depressing LD REG, the contents of the Switch Register will be deposited at the specified address held in BA and the Data Display will reflect the new data. Sequential memory locations can be written while BD is selected by depressing BAINC, updating the Switch Register to the new data, and then depressing LD REG.

The ML and MH registers are the Memory Low limit and the Memory High limit registers. The memory space between these two limits, which can be set by the user, is available for loading new programs. The memory space below the Memory Low limit and above the Memory High limit is protected from being overwritten during loading operations.

The FN (Function Number) register, the F1 register and F2 register can control the high speed reader/punch if a teletype is not available for communication with the Resident Firmware Operating System. After depressing MCLR, but before depressing START/STOP, select the desired operation by setting up these registers as follows:

<u>FN=1:</u> Load program tapes via the high speed reader. If relocation is desired, set F1 to the relocation address. F2 is not used in this mode.

<u>FN=2:</u> Punch the contents of memory via the high speed punch. F1 is set to the low address; F2 is set to the high address.

<u>FN=3:</u> Copy a tape via the high speed reader/punch. F1 and F2 are not used in this mode.

1.3.3 GIC1600 Operation Via Control Console

Operation of the GIC1600 Series Microcomputer Systems must initially begin with the pressing of MCLR. This function initializes all internal hardware, supplies the starting address of the Resident Operating System to the PC (R7) and halts. Execution of the Resident Operating System begins by pressing START/STOP. If the user does not wish to enter the Operating System, the PC must be modified before pressing START/STOP. A new starting address must be keyed into the Switch Register, R7 selected and LD REG pressed. When START/STOP is pressed, the CPU will begin executing instructions at the supplied address.

The system can either be in CONT (Continuous) mode or SINGLE INSTR mode. In CONT, the system will free run until either a HALT instruction is executed or START/STOP is pressed. In SINGLE INSTR mode, the CPU will normally execute one instruction before halting. Depressing START/STOP repeatedly will allow single stepping through the program. The only instructions that have been designed to be bypassed in SINGLE INSTR mode are TCI and HLT. For these instructions the CPU will stop after executing the next instruction. (If, however, the following instructions are MVO, shift or control instructions, the CPU will stop after executing the first instruction that is not a MVO, shift, control, TCI or HLT instruction.)

1.3.4 Operational Flow Charts

The diagrams shown in Figs. 5 to 8 depict step-by-step flow charts of the following procedures:

Fig.	Procedure
5	GIC1600 Start Up Procedure
6	Procedure to Examine Memory
7	Procedure to Deposit New Data into Memory
8	Procedure to Examine and/or Modify the Internal CPU Registers

1.4 DATA, ADDRESS & CONTROL BUSES

The GIC1600 Series Microcomputer Systems have a Data Bus, an Address Bus, and a Control Bus which connect the microprocessor, the memory, and all the peripherals. The form of communication is the same for every device on the bus.

Since a single Address Space concept is utilized in the CP1600 microprocessor architecture, memory and peripheral devices reside within the same 65K Address Space. The system address allocation alone differentiates memory from I/O devices; there-

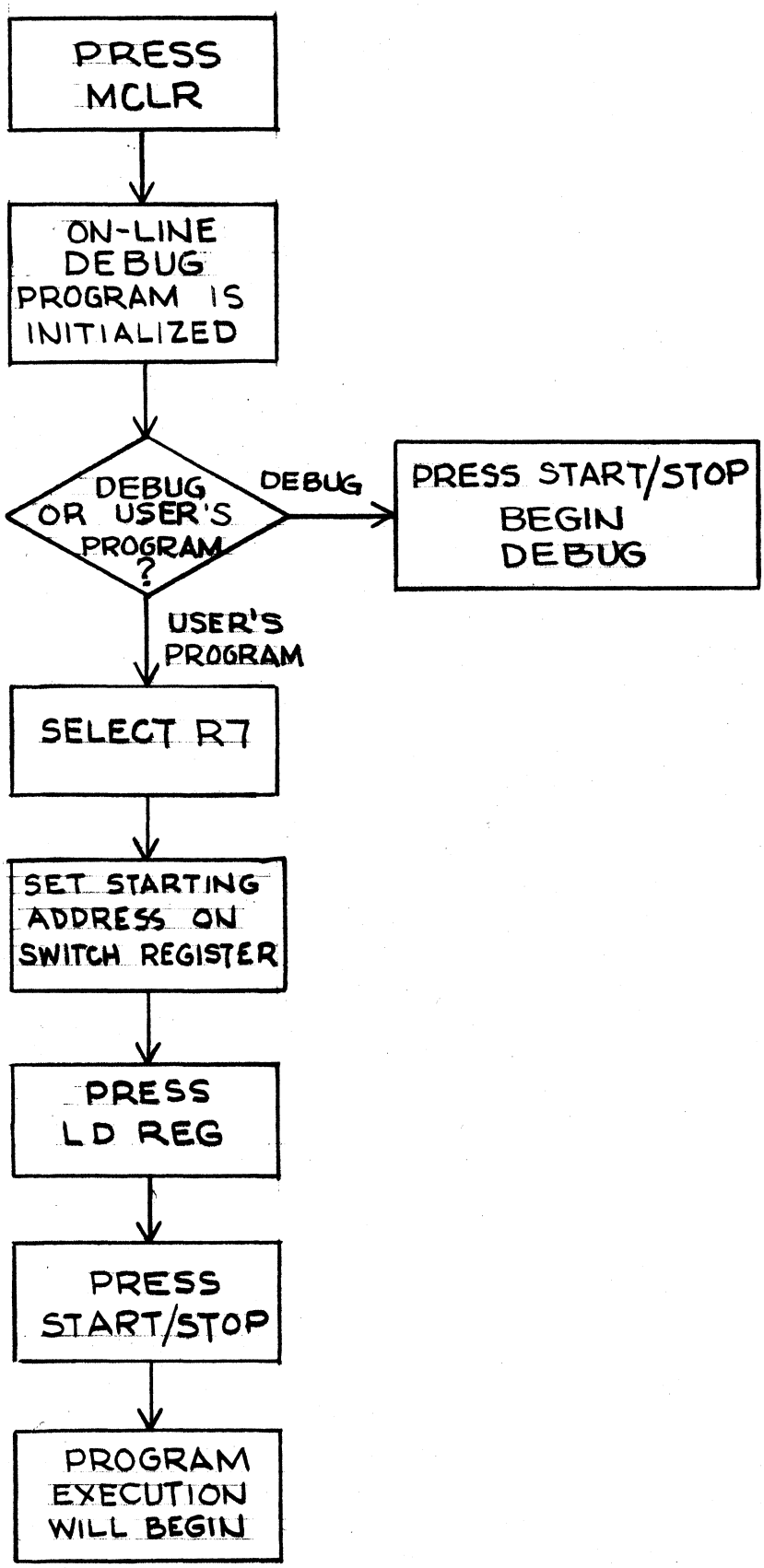


FIG. 5 . GIC 1600 START UP PROCEDURE.

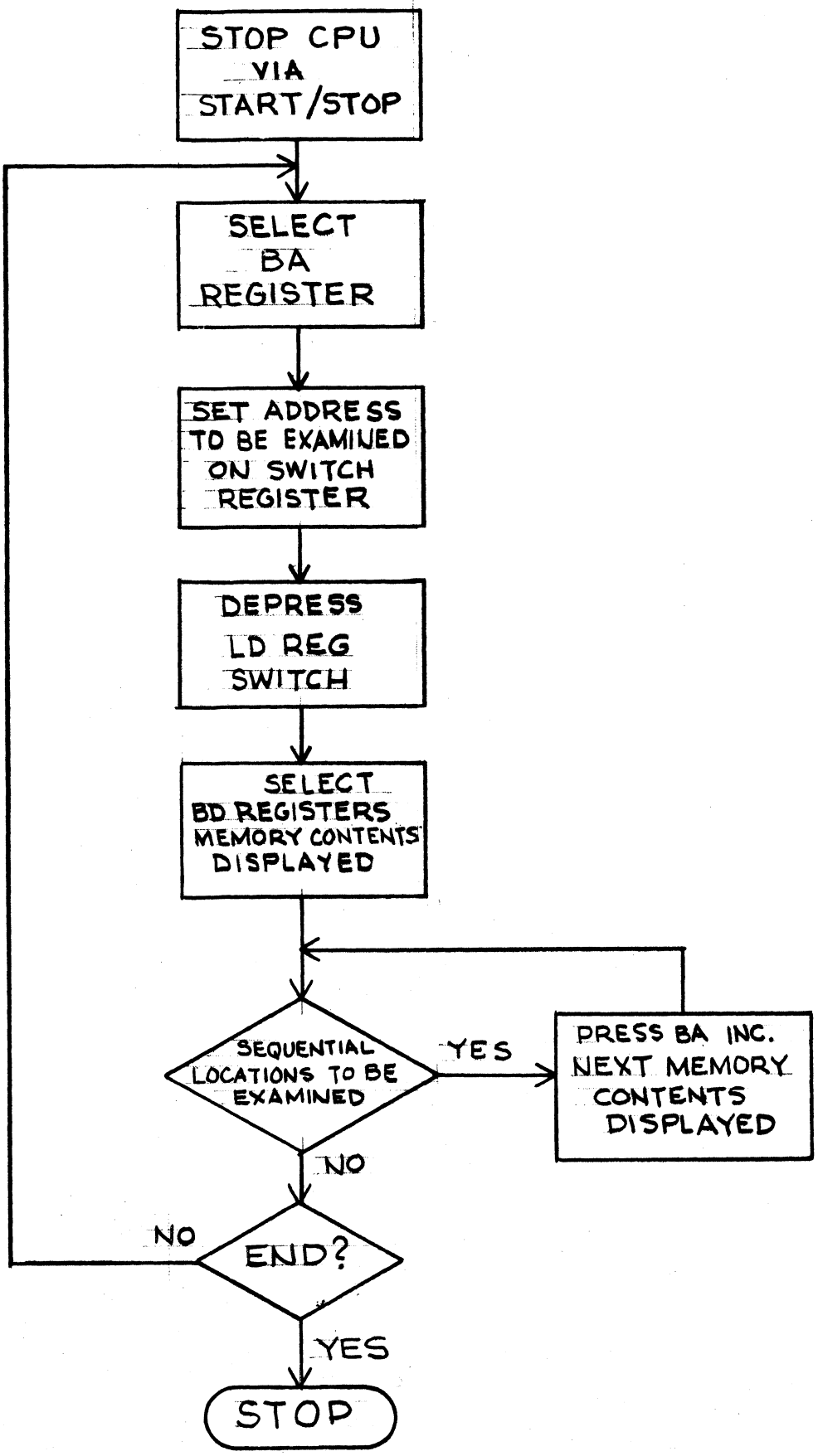


FIG. 6. PROCEDURE TO EXAMINE MEMORY.

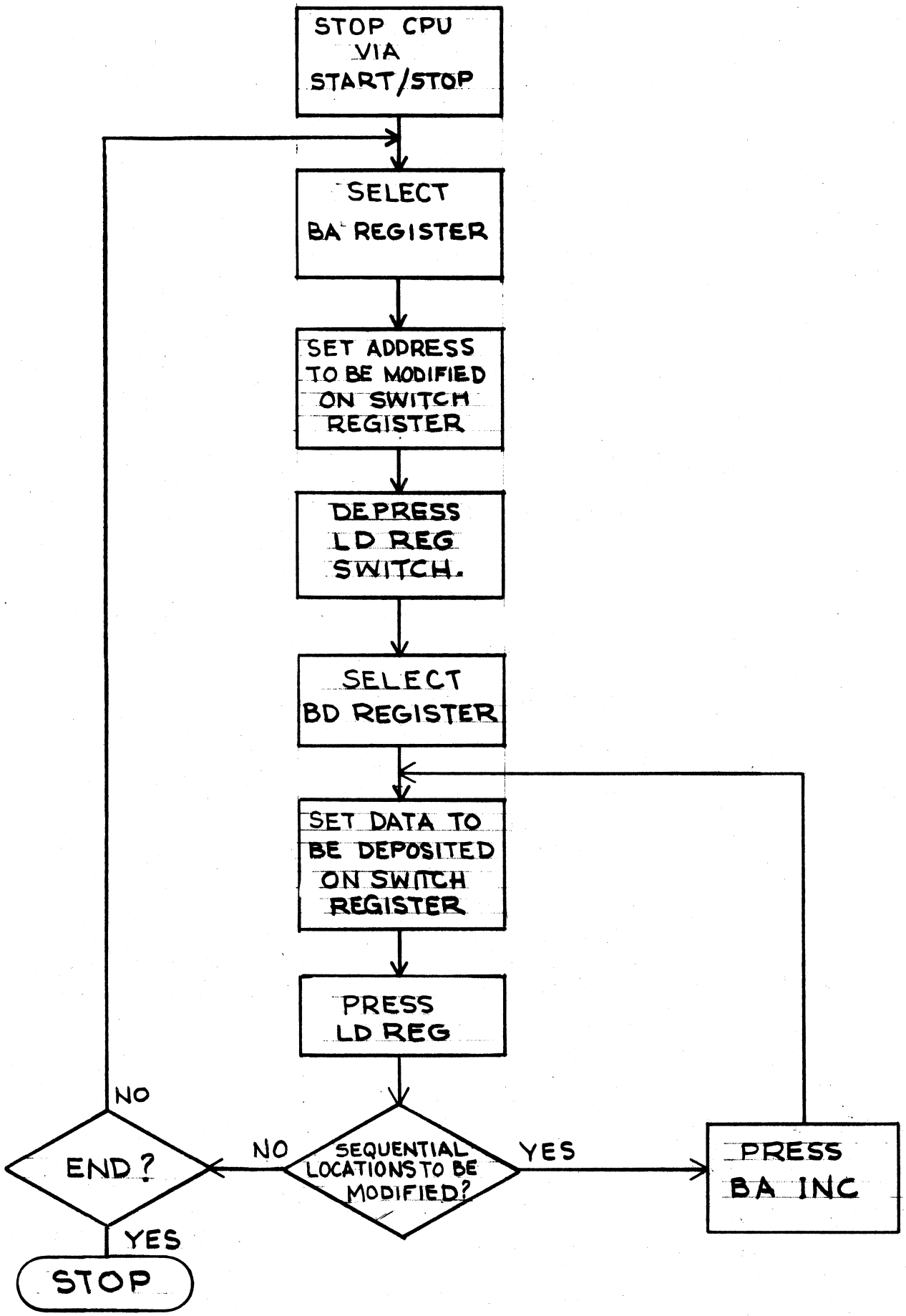


FIG. 7 . PROCEDURE TO DEPOSIT NEW DATA INTO MEMORY

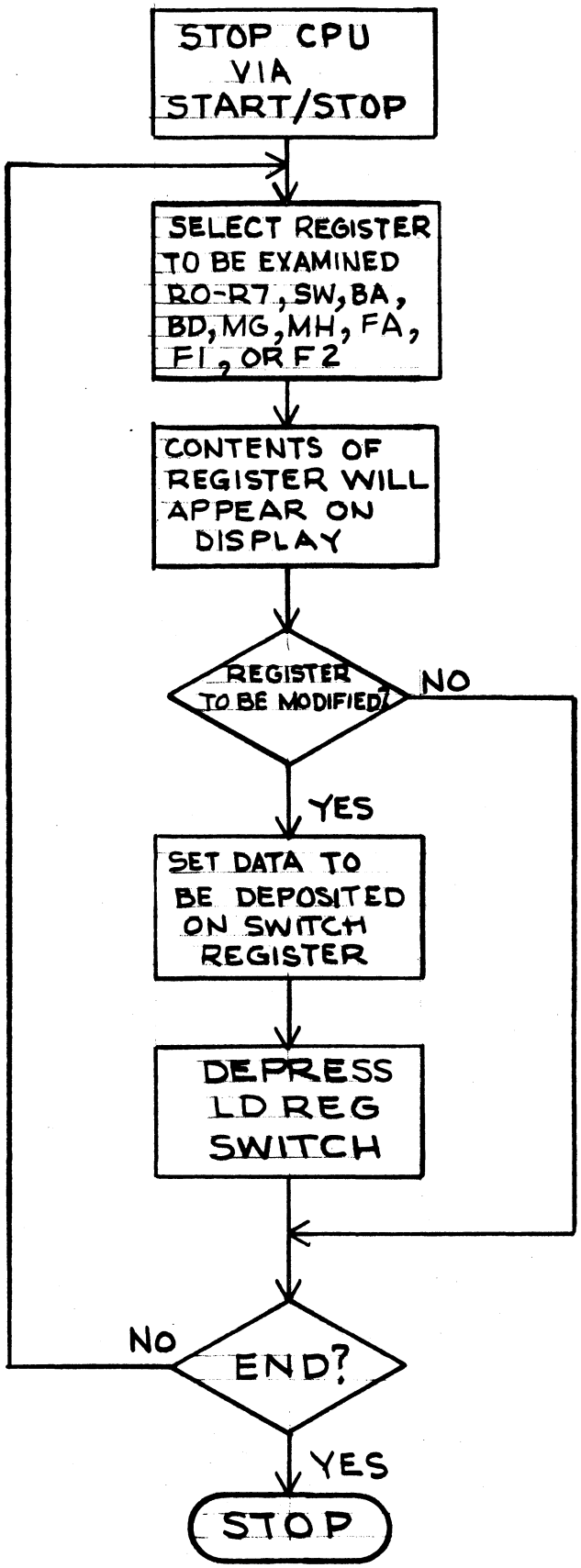


FIG. 8 . PROCEDURE TO EXAMINE AND/OR MODIFY THE INTERNAL CPU REGISTERS.

fore, no special I/O instructions are required and any External Reference instruction can access memory or peripheral devices. Peripheral devices in the system are addressed and operated upon by the software just as if they were memory locations. Since the GIC1600 Series Systems use a 16-bit address, the maximum Address Space is 65,536 locations.

The Data Bus is bidirectional, i.e., the CPU can send data to, and receive data from, any peripheral device, such as a memory card, a Teletype, or a high speed reader/punch. The Address Bus is derived from latching the data on the Data Bus into the Address Register at the appropriate intervals of time, such as during the ADAR*, BAR*, and INTAK* control pulse times.

The Control Bus provides eight buffered output control signals to define the function to be performed on the Data and Address Buses. These signals are used to control all communication between the microprocessor, memory, and all peripheral devices.

When a peripheral device requests bus control for Direct Memory Access (DMA), the Data, Address and Control Buses will enter a high output impedance state. This will enable a peripheral device to have complete control of the Data, Address and Control Buses for DMA operation or any other type of communication.

The Data, Address and Control Buses are buffered and available on the backplane of the chassis so the user can develop his own custom interface.

1.4.1 Backplane Signal Descriptions - Figs. 9-13 illustrate the use of these signals.

Data Bus

Data Bus Lines (DBL0-15*)- Backplane Terminated

These 16 signals comprise the bidirectional bus that is used for all data communication. Since the CPU sends both data and address information to the external environment, the bus is also used to load the Address Register. The function to be performed on the data bus is determined under program control via the Bus Control signals. The Data Bus must be driven by open collector drivers capable of driving 30 TTL loads (7438 or equiv.).

Drive Capability: 18 TTL loads available to user.

Address Bus

Bus Address Register Lines (BADR0-15*) - Backplane Terminated

These 16 output lines are used to select a peripheral device or an address location in memory. The address register is clocked in the middle of CK4* during either ADAR*, BAR*, or INTAK* and the address will

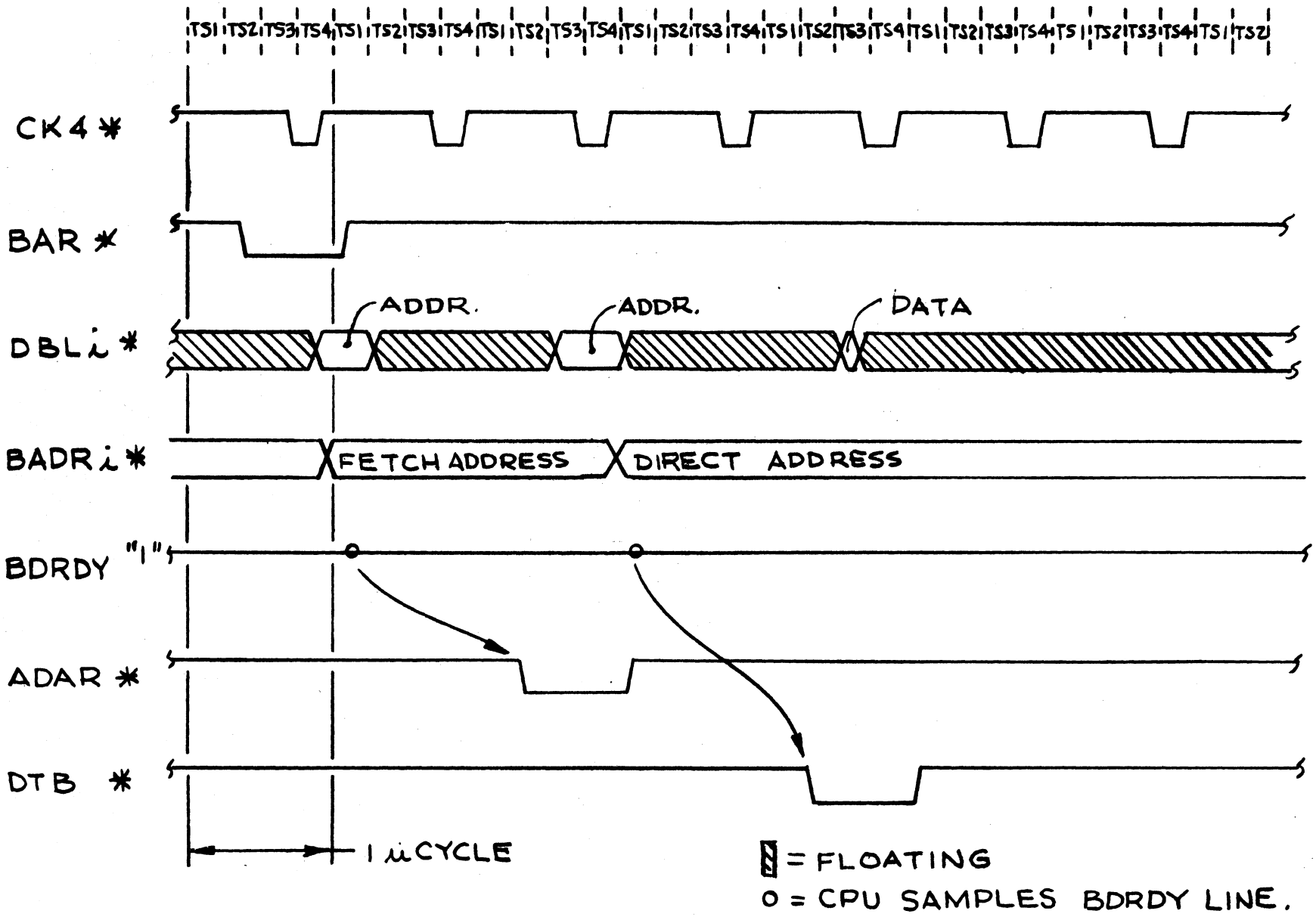


FIG. 9. READ SEQUENCE - NO DELAY DIRECT ADDRESSING

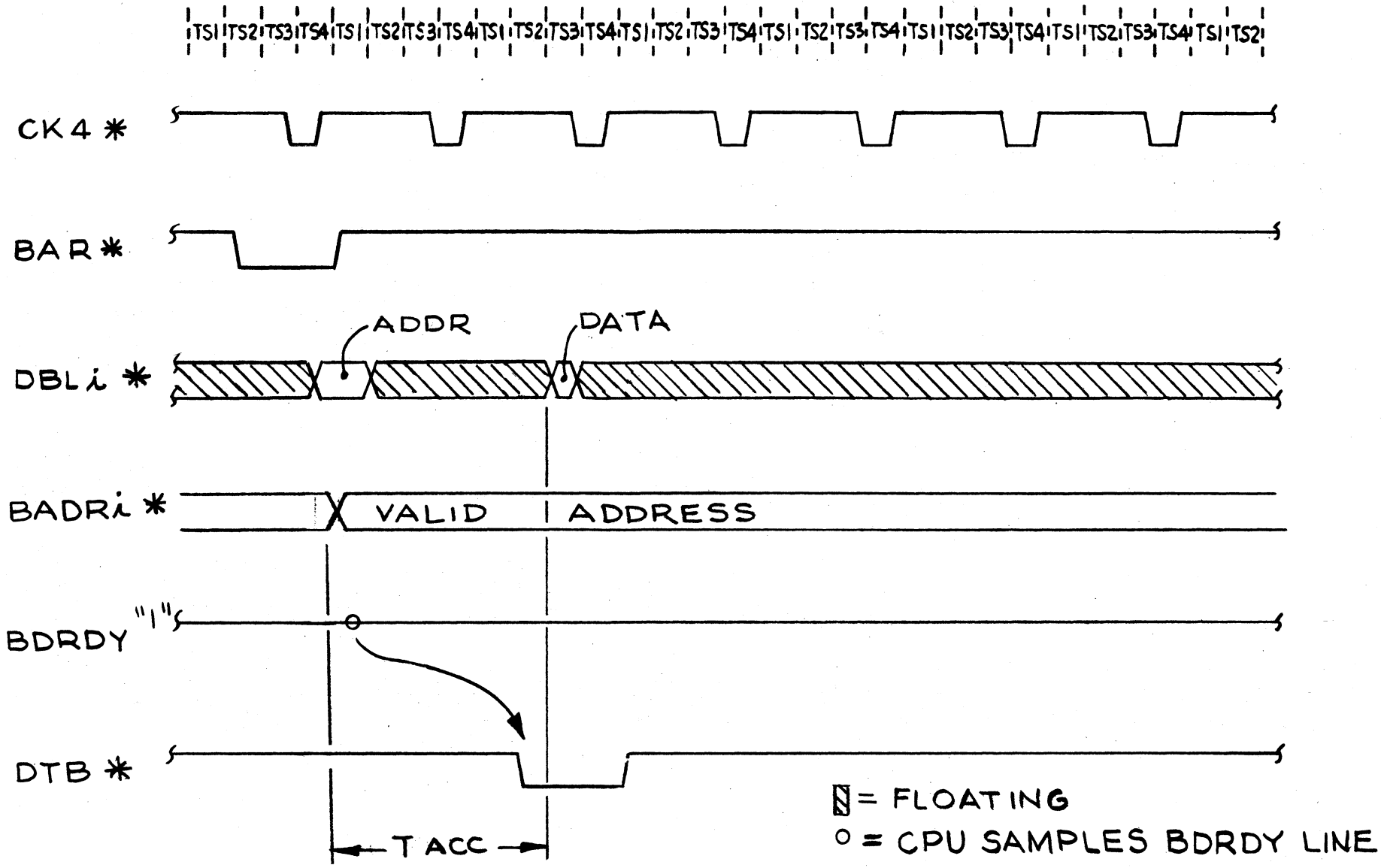
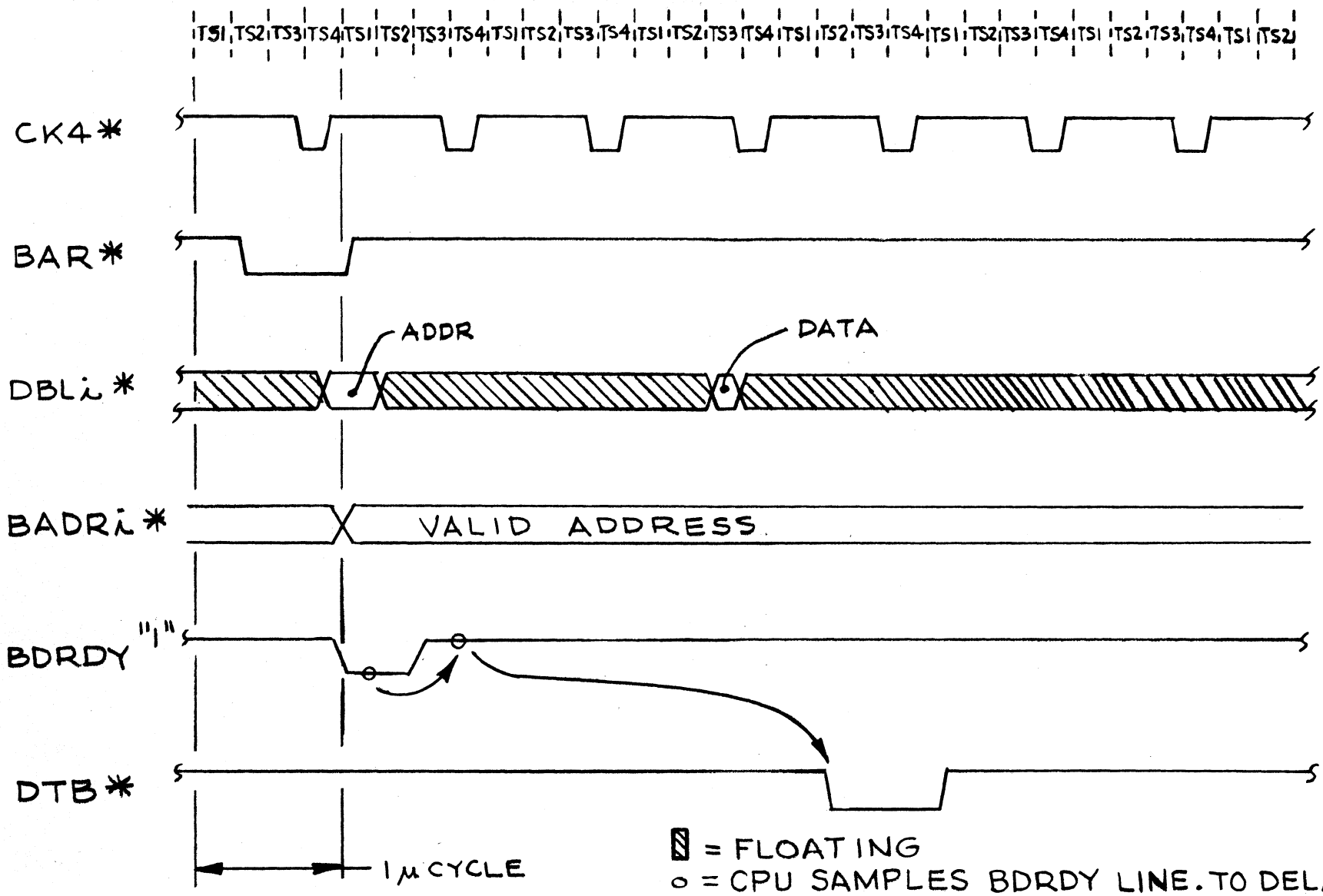
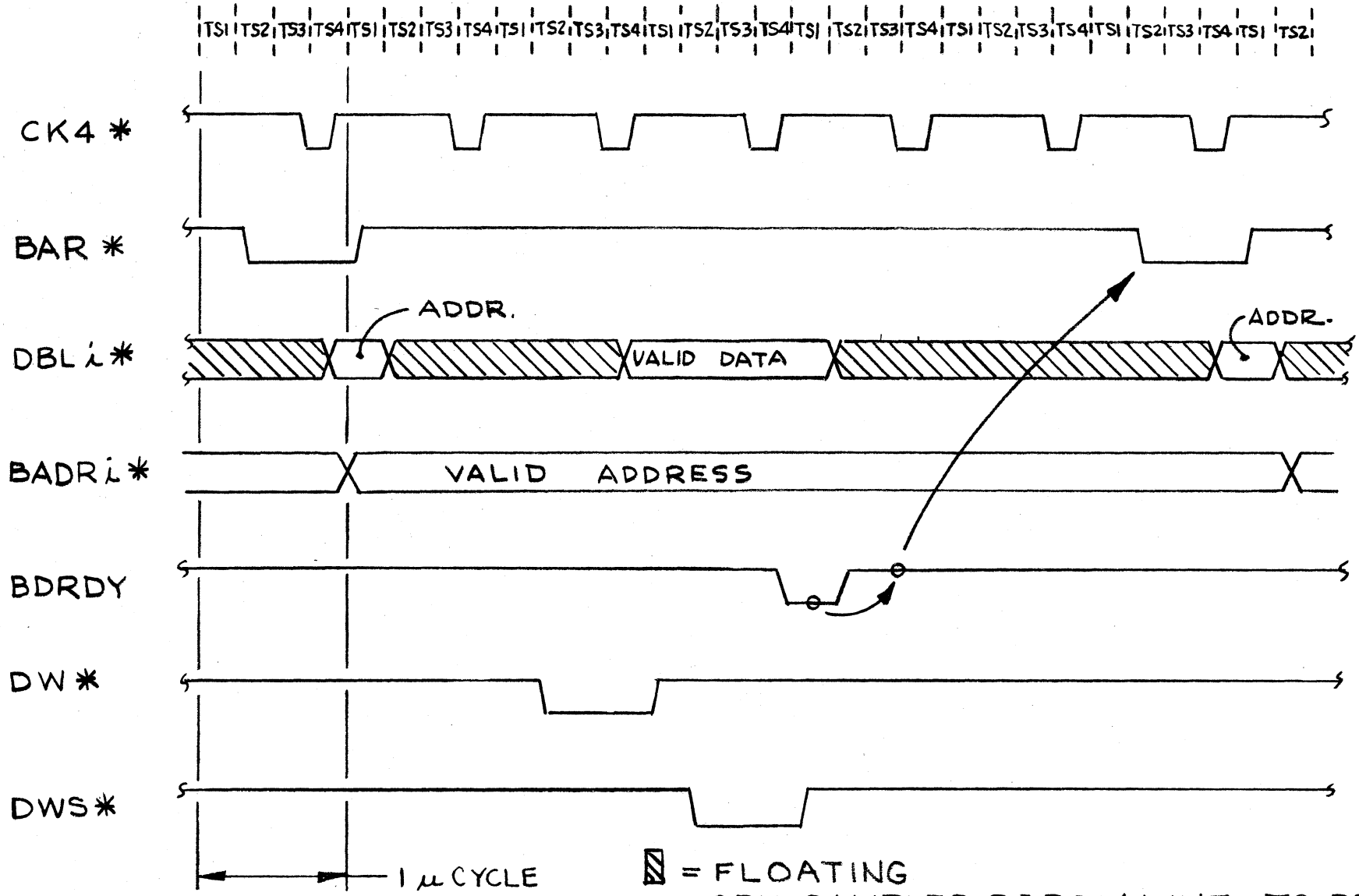


FIG. 10. READ SEQUENCE - NO DELAY
REGISTER INDIRECT ADDRESSING.



▨ = FLOATING
o = CPU SAMPLES BDRDY LINE. TO DELAY CPU, LOWER BDRDY NO LATER THAN 50 NS BEFORE THE END OF TS1; CPU WILL AGAIN SAMPLE BDRDY AT THE BEGINNING OF EVERY TS4.

FIG. 11. READ SEQUENCE - WITH ONE CYCLE OF DELAY. REGISTER INDIRECT ADDRESSING.



▨ = FLOATING

o = CPU SAMPLES BDRDY LINE. TO DELAY CPU, LOWER BDRDY NO LATER THAN 50NS BEFORE THE END OF TS1; CPU WILL AGAIN SAMPLE BDRDY AT THE BEGINNING OF EVERY TS4.

FIG. 12. DATA WRITE SEQUENCE - WITH DELAY REGISTER INDIRECT ADDRESSING.

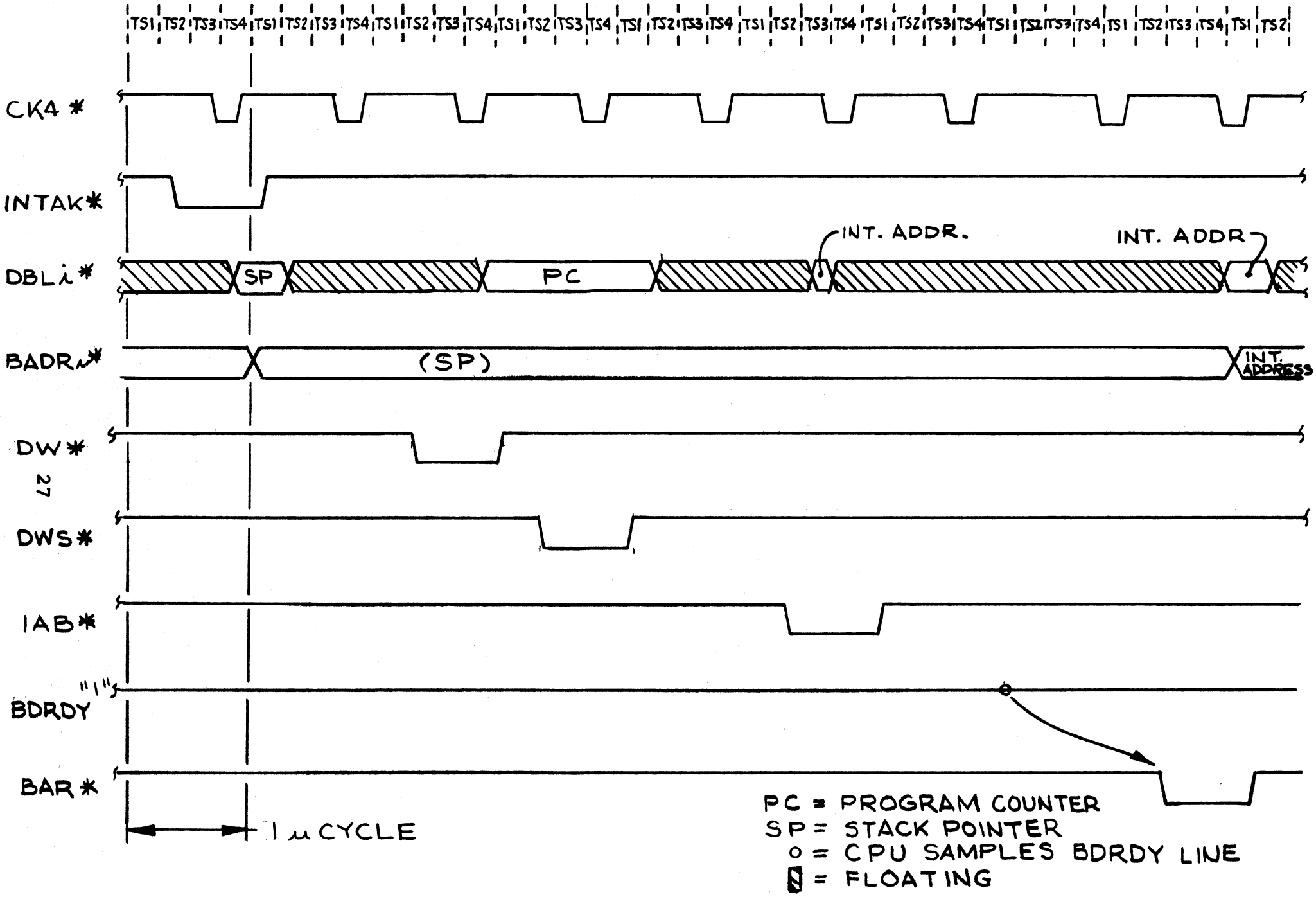


FIG. 13. INTERRUPT SEQUENCE

remain valid until the next clocking. The Address Bus is driven by open collector devices capable of driving 30 TTL loads. The lines are unidirectional and can also be driven under DMA operation with additional sets of open collector gates (7438 or equiv.) associated with the DMA device.

Drive Capability: 18 TTL loads available to user.

Control Bus

Bus to Address Register (BAR*) - Backplane Terminated

This signal is used to load the Data Bus into the Address Register. It may be useful in informing peripherals that the Address Bus is valid by the trailing edge of BAR*.

Drive Capability: 20 TTL loads available to user.

Data Write Strobe (DWS*) - Backplane Terminated

This signal is used as a write enable for memory or any peripheral device. Data will be placed on the Data Bus 100ns prior to the leading edge of DWS* and remain valid for at least 50ns after the trailing edge.

Drive Capability: 18 TTL loads available to user.

Data Write (DW*) - Backplane Terminated

This signal is functionally identical to DWS* except that it occurs one microcycle time prior to DWS*. It may be used for extended writing operations.

Drive Capability: 21 TTL loads available to user.

Data to Bus (DTB*) - Backplane Terminated

This signal is used to gate instructions, addresses, or data from memory or any peripheral device onto the Data Bus. If the data is to be input to the CPU, it must be stable at the microprocessor input pins within 80ns after the leading edge of DTB* and remain valid for another 100ns.

Drive Capability: 20 TTL loads available to user

Interrupt Acknowledge (INTAK*) - Backplane Terminated

This signal is generated by the CPU denoting its acceptance of an interrupt request and initiates the "daisy-chain" priority network to find and acknowledge the highest priority device presently requesting interrupt service.

Drive Capability: 19 TTL loads available to user.

Interrupt Address to Bus (IAB*) - Backplane Terminated

This signal occurs during the Interrupt Sequence of the CPU. It occurs after the interrupt has been acknowledged and serves to gate the starting address of the service routine for the highest priority interrupting device onto the Data Bus. IAB* also occurs after MCLR is depressed and during the power-up initialization sequence to input the starting address of the main program to the Program Counter. The address to be input to the CPU must be stable at the microprocessor input pins within 80ns after the leading edge of IAB* and remain valid for another 100ns.

Drive Capability: 19 TTL loads available to user.

Addressed Data To Address Register (ADAR*) - Backplane Terminated

This signal causes the addressed contents of memory to be gated onto the Data Bus and strobed into the Address Register. It is generated in response to all instructions which specify direct addressing.

Drive Capability: 19 TTL loads available to user.

No Action (NACT*) - Backplane Terminated

This signal indicates that the CPU is not using the bus.

Drive Capability: 20 TTL loads available to user.

Other Backplane Signals

Terminate Current Interrupt (TCI*) - Backplane Terminated

This signal, which is valid for one microcycle time, is generated by the execution of a Terminate Current Interrupt instruction in the program. It resets the highest priority interrupt presently being serviced and establishes priority at the next highest priority device which could be the main program if no other interrupts were in service or pending service.

Drive Capability: 19 TTL loads available to the user.

Clock Four (CK4*) - Backplane Terminated

This signal is available to the user to synchronize external devices with Time Slot 4 of the CPU.

Drive Capability: 19 TTL loads available to the user.

External Branch Condition (EBC0-15*) Microcomputer Input

The system provides input ports for 16 external conditions which can

be used as external branch conditions under program control. The sense lines must be stable 100ns prior to examination by the CPU for proper branching. A "0" active input will result in branching.

Load: 1 TTL load

Bus Request (BUSRQ*) - Microcomputer Terminated

The bus request line informs the processor that an external device requires the use of the Data Bus. The CPU grants the use of the bus after the completion of an interruptable instruction and responds with BUSAK*. The microprocessor then becomes inactive with its Data Bus driver/receiver inactive and NACT* active. The CPU will remain in this condition until the external device releases BUSRQ*.

Load: 4 TTL loads

Bus Acknowledge (BUSAK*) - Microcomputer Driven

BUSAK* becomes active when BUSRQ* is received by the CPU and the execution of the next interrutable instruction has been completed. This line will remain active as long as BUSRQ* remains low. This signal serves to inform the requesting device that the processor has surrendered control of the Data Bus for DMA or non-processor controlled bus operations.

Drive Capability: 10 TTL loads available to user

Bus Data Ready (BDRDY) - Microcomputer Terminated

This signal permits resynchronization of the CPU for peripheral subsystems or memories that cannot respond to requests for reads and writes at full CPU speed. BDRDY must go low no later than 50nsec after the end of either BAR* or ADAR* for reading and DWS* for writing in order to begin delay operation. The CPU then samples BDRDY at the leading edge of TS4 for additional delay. The duration of the wait period must be less than 40 microseconds to preserve the dynamic status of the CPU.

Load: 4 TTL loads

Interrupt Request (INTR*) - Microcomputer Terminated

This line is the higher priority interrupt request line of the two interrupt request lines. It is not affected by the state of the interrupt mask F/F within the CPU. The CPU will honor the request only after the

completion of the next interruptable instruction. The user should remove the INTR* being generated by a device after the CPU acknowledges that device's request with INTAK*.

Load: 4 TTL loads

Interrupt Request Maskable (INTRM*) - Microcomputer Terminated

This line is the lower priority of the two interrupt request lines. It is effective in generating an interrupt only if the interrupt mask F/F within the CPU has been cleared. The interrupt mask F/F is accessable under program control via the Enable Interrupt System (EIS) and Disable Interrupt System (DIS) instructions.

Load: 4 TTL loads

Disable Bus Address Register (DISBAR*) - Microcomputer Terminated

This signal is used for DMA operation. A logical "0" input applied to DISBAR* will float the Address Bus whenever the CPU has acknowledged a BUSRQ* signal with BUSAK*.

Load: 4 TTL loads

Halt (HALT*) - Microcomputer Driven

This signal indicates that the CPU is in the stopped mode. This mode can occur either by the toggle action from the START/STOP Switch on the front panel or by the execution of a HALT instruction.

Drive Capability: 9 TTL loads available to user

Stop/Start (STPST) - Microcomputer Input

This is a negative edge-triggered signal used to control the running condition of the CPU. If the CPU is presently running, the negative transition of STPST will cause the CPU to stop but only after the completion of an interruptable instruction. The CPU will generate a high active HALT signal indicating the stopped condition. The next negative transition of STPST will cause the CPU to return to the run mode. The HALT output will then return to a logic "0" (low) condition.

Note: The STPST signal is used for Control Console operation only.

Master Clear (MCLR*) - Microcomputer Terminated

This signal is used to initialize all internal hardware and reset the internal timing of the processor to its starting condition. MCLR* is

hardwired to the front panel switch MCLR but may also be "wire-or" connected to any peripheral device's MCLR* signal to initialize the device to a known state.

Load: 4 TTL loads

Disable Data to Bus (DISDTB*) - Control Console Driven

This signal is generated by the control console card to disable all memory and peripheral devices from interfering with bus operations at certain critical times. For proper control console operation every I/O device must use this signal to disable its address decoder.

Drive Capability: 1 TTL load

High Byte (HGBT*) - Microcomputer Terminated

This signal is used during the reading of RAM memory. A logic "0" applied to HGBT* will mask out the lower byte (Bits 0-7) of an addressed memory location from outputting onto the Data Bus. The high byte (Bits 8-15) will not be effected. This signal is not used in the GIC1600 Systems and is permanently tied high (inactive) on the Microcomputer Module.

Low Byte (LWBT*) - Microcomputer Terminated

This signal is used during the reading of RAM memory. A logic "0" applied to LWBT* will mask out the higher byte (Bits 8-15) of an addressed memory location from outputting onto the Data Bus. The low byte (Bits 0-7) will not be effected. This signal is not used in the GIC1600 Systems and is permanently tied high (inactive) on the Microcomputer Module.

Interrupt Priority In (IPRI*)
Interrupt Priority Out (IPRO*) — Daisy-Chained Card to Card

These two signals are used to resolve interrupt device priority and to allow INTAK* to acknowledge only the highest device.

Interrupt Mask In (IMSKI)
Interrupt Mask Out (IMSKO) — Daisy-Chained Card to Card

These two signals are used to mask out all lower priority devices from requesting an interrupt while a higher device is being serviced.

Bus Acknowledge In (BAKI*)
Bus Acknowledge Out (BAKO*) — Daisy-Chained Card to Card

These two signals are used to resolve DMA priority and to allow BUSAK*

to acknowledge only the highest peripheral request.

Bus Mask In (BMSKI)
Bus Mask Out (BMSKO) — Daisy-Chained Card to Card

These two signals are used to mask out all lower priority devices from requesting use of the bus while a higher device has bus control.

Program Counter Inhibit (PCIT*) - Microcomputer driven and Terminated

This signal provides two functions:

a) As an input, this signal is a low active signal that prevents the incrementing of the Program Counter (R7) during the fetch phase of all instructions.

b) As an output, this signal will generate a low active pulse during the execution of the SIN (Software INterrupt) instruction. This signal is received by the Control Console Card which in turn generates an interrupt request on the INTR* line. This interrupt request is acknowledged at the end of the SIN instruction resulting in a jump into the Resident Operating System (TRAP function). These functions are designed so that they will not interact with each other under normal operation of the system.

Note: The PCIT*signal is used for Control Console operation only.

1.4.2 Direct Memory Access Operation

The GIC1600 Series Microcomputer Systems have the capability to handle high speed data transfers via DMA operation. External devices requesting DMA service must activate BUSRQ* and then take control of the Data Bus when the CPU transmits back the BUSAK* signal. When the device receives BUSAK*, the interface can perform high speed transfers at speeds limited only by the memory system cycle time. The interface has the option of doing address and data transfers sequentially over the Data Bus using the Bus Address Register on the Microcomputer Module as a temporary address buffer. It can also do address and data transfers in parallel driving both Data and Address Buses at the same time. In order to use the parallel mode, the device must activate DISBAR* (Disable Bus Address Register) which disconnects the Bus Address Register from the Address Bus so that it can be driven externally. In either case, the external device must drive the Control Bus signals (except for NACT* which is always driven by the CPU when it is not using the bus) to define the bus operation during DMA cycles. The external device thus becomes bus master and is responsible for complete timing and control of information transfers on the Data Bus.

All Data Bus, Address Bus and Control Bus signals from the GIC1600 Series Microcomputers are in a high-output impedance state during a DMA operation. The NACT* signal provides a 375ns pulse every microcycle time which can be utilized by the external device for DMA control. In addition, the Microcomputer Module also provides a CK4* pulse every microcycle for additional timing flexibility. If more than one external device requests DMA operation, priority associated with these devices is handled via BAKI*/BAKO* and BMSKI/BMSKO priority daisy-chain networks located on the device controllers.

1.4.3 Interrupt Operation

The GIC1600 Microcomputer System has two programmable interrupt lines, INTR* and INTRM*. These signals request the CPU to honor an interrupt at the completion of any interruptable instruction under the following conditions:

a) INTR* is always honored by the CPU and hence is the highest priority interrupt request line.

b) INTRM* is honored by the CPU only if the internal CPU interrupt flip-flop is enabled.

The GIC1600 Systems acknowledge either INTR* or INTRM* by giving an INTAK* signal to the peripheral devices. This is followed by an IAB* signal to bring the starting address of the interrupt service routine for the acknowledged interrupting device into the Program Counter of the CPU. All interrupt system servicing is handed via hardware/software combination. Refer to Device Interrupt Logic shown in Fig. 14.

In order for peripheral devices to resolve priority, each device has a priority chain set-up as shown in Figure 15. Each device connects to the Interrupt Request line in order to request an interrupt operation. The CPU acknowledges the interrupt request by sending an interrupt acknowledge pulse, INTAK*, to the first device in the interrupt priority daisy-chain network. The pulse proceeds down the chain from IPRI* to IPRO* of each device until it finds the device requesting an interrupt. This device then becomes acknowledged, blocks the priority pulse from further propagation, and sends out a mask signal IMSKO that propagates to all lower devices via the IMSKO to IMSKI chain. The mask chain insures that no lower priority devices can interrupt the service of the acknowledged device. The CPU then issues an IAB* signal to all devices; however, only the highest priority device presently acknowledged uses this signal to present the starting address of its service routine to the Data Bus. This starting address is then strobed into the Program of the CPU.

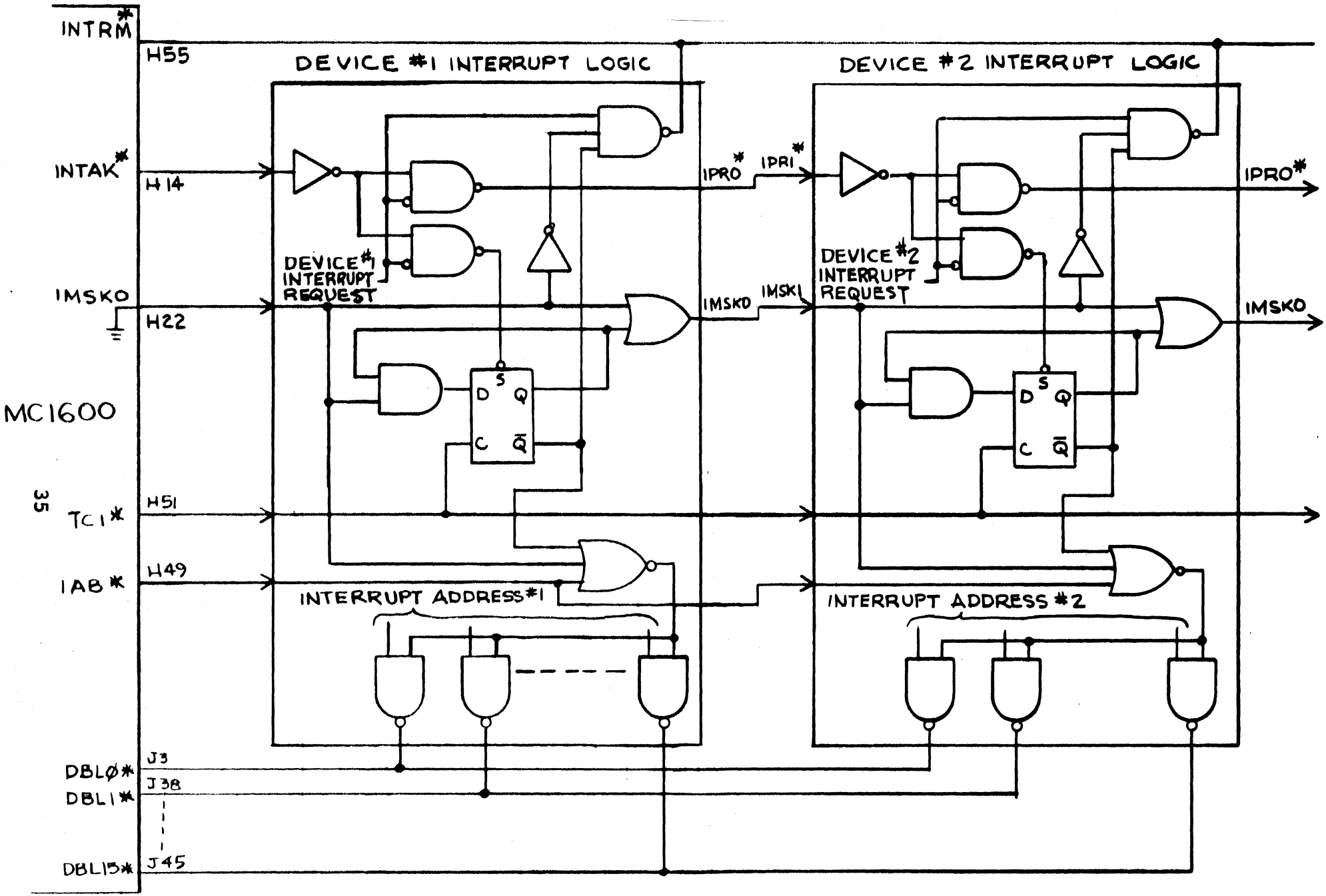
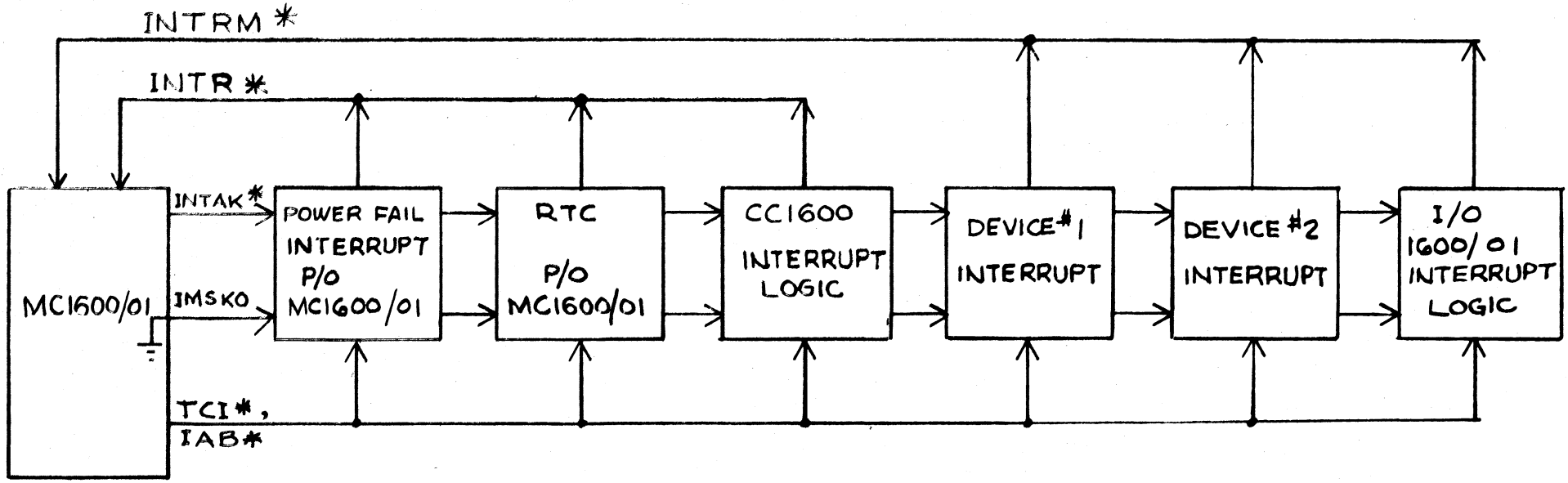


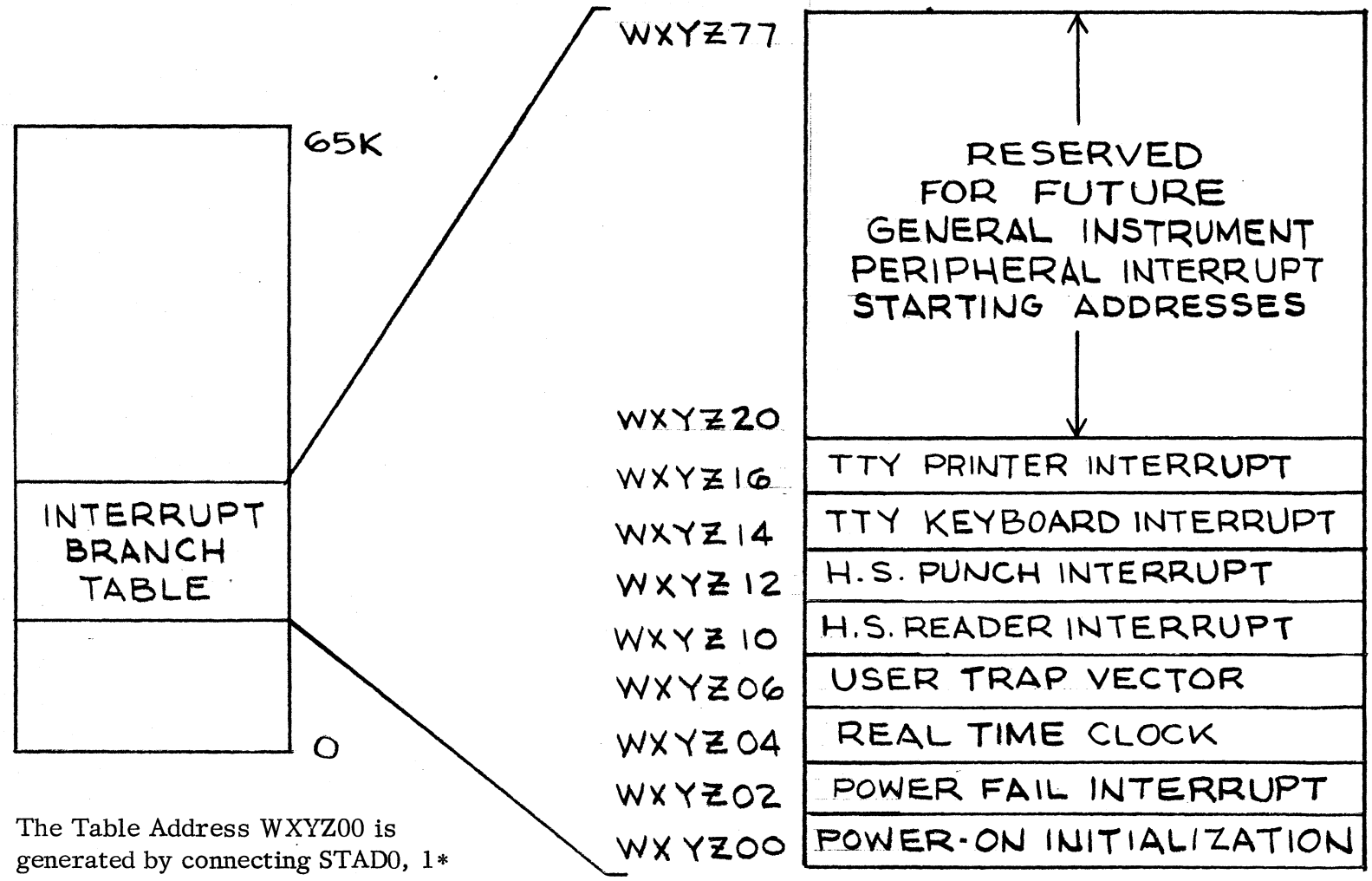
FIG. 14 DEVICE INTERRUPT LOGIC.



The basic GIC1600 Series Microcomputer Systems are factory wired such that the Power Fail Interrupt & Real Time Clock (on MC1601) are connected to INTR* followed by the Control Console Interface Interrupt. All additional peripherals are wired into the priority chain as the user desires. The interrupt logic on I/O1600/1601 is wired to have the lowest order priority.

Once a device is connected to INTRM* rather than INTR*, all lower order priority devices must be connected to INTRM* also.

FIG. 15. INTERRUPT SYSTEM CONNECTION.



The Table Address WXYZ00 is generated by connecting STAD0, 1* to any two DATA BUS LINES (DBL) between DBL6 and DBL15. The device identification code is contained in the lower six bit positions. W, X, Y, Z are octal numbers representing where STAD0, 1* are connected; e.g. 014000 would indicate that STAD0, 1* are connected to DBL11 & DBL12.

Each Interrupt Vector is 2 words normally containing a BRANCH or JUMP instruction to the start of the service routine.

FIG. 16. INTERRUPT BRANCH TABLE

The interrupt addresses supplied by the peripherals in the GIC1600 Systems are arranged to be consecutive entries into an Interrupt Branch Table. Each entry in this table consists of two words which normally contain a BRANCH instruction to direct the CPU to the appropriate service routine. The Interrupt Branch Table can be located anywhere in memory and is defined by the connection of J13 (STADO*) and J48 (STAD1*) to the appropriate Data Bus lines (DBL0-15*). The interrupting device generates a unique code defining the low order part of its interrupt address and the IAB* signal automatically gates STADO* and STAD1* defining the upper part of all interrupt addresses. The result is an Interrupt Branch Table as shown in Fig.16.

At the completion of the interrupt service routine, the CPU generates a Terminate Current Interrupt signal, TCI, to reset the current interrupt (highest priority interrupt presently being serviced) and re-establish priority at the next lowest device needing service.

1.5 PERIPHERAL OPERATION

The basic GIC1600 Series Microcomputer System supports both Teletype and high speed paper tape reader/punch peripheral devices. All necessary hardware is contained in the fully character-buffered controllers located on the I/O1600/1601 Input/Output Module and all necessary software drivers are contained within the ROM Resident Operating System.

1.5.1 Teletype/EIA Devices

An ASR33 Teletype (Model ASR33, Catalog No. 3320/XXX with 20/60mA current loop) or any EIA compatible device can be used to provide interactive communication between the GIC1600 Series Microcomputers and the user. The TTY (or EIA device) can input data to the microcomputer via its keyboard or paper tape reader. The microcomputer can output information to the TTY (or EIA device) printer or paper tape punch.

To use the Teletype with the GIC1600 Series Microcomputers, install the TTY cable plug into the smaller connector (10 pins) on the I/O1600/1601 Input/Output Module. The ASR33 Teletype must receive the following internal modifications and external connections (refer to Figures 17 and 18).

Internal Modifications

1. The current source resistor value must be changed to 1450 ohms. This is accomplished by moving a single wire.

2. A full duplex hook-up must be created internally. This is accomplished by moving two wires on a terminal strip.

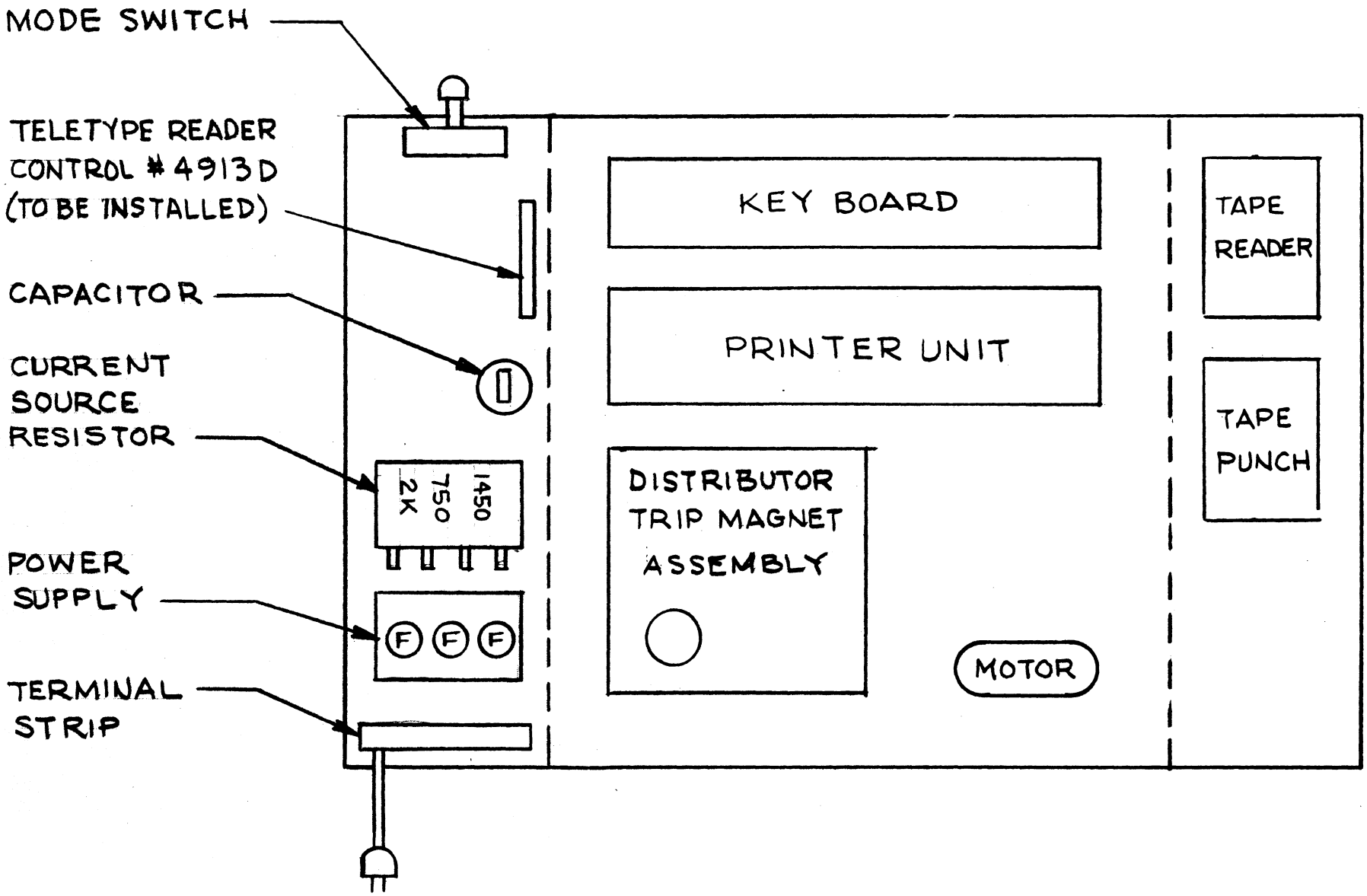


FIG. 17. TOP VIEW - TELETYPE MODEL ASR33.

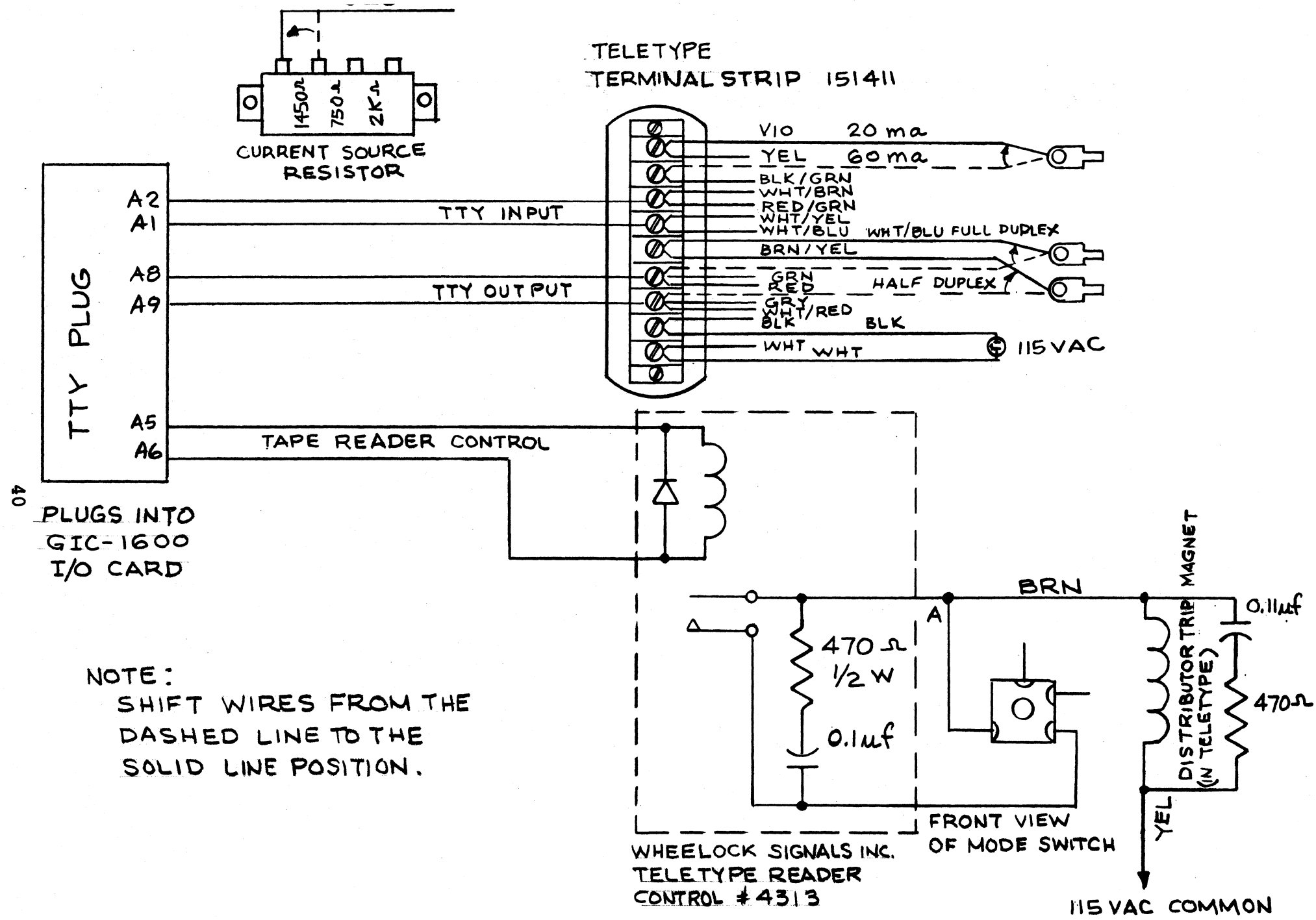


FIG. 18 . INTERFACE OF TELETYPE ASR33 AND GIC-1600 I/O 1600/1601 CARD.

3. The receiver current level must be changed from 60mA to 20mA. This is accomplished by moving a single wire.

4. A relay circuit must be introduced into the paper tape reader drive circuit if the TTY reader is to be used to asynchronously read tapes under program control. The recommended circuit consists of a relay, a resistor, a capacitor, and suitable mounting fixture. This relay network is manufactured by Wheelock Signals, Inc., as part "Teletype Reader Control #4913D". It may be mounted in the Teletype by using two tapped holes in the mounting plate shown in Figure 17 The relay network may then be added without alteration of the existing Teletype circuits. That is, Wire "A", to be connected to the brown wire in Figure 18, may be spliced into the brown wire near its connector plug. . The "Line" and "Local" wires must then be connected to the mode switch. Existing reader control circuitry within the teletype need not be altered.

External Connections

1. A two-wire receive loop, a two-wire send loop, and a two-wire tape reader control loop must be created. This is accomplished by the connection of six wires between the Teletype and the TTY Plug to the I/O1600/1601 card.

2. The TTY/EIA interface cable should have the following pin assignments:

SIGNAL	GIC1600 (Upper Connector on I/O1600/1601 Module) (3M Connector #3473-0000)
TTY PRINTER-	A1
TTY PRINTER+	A2
EIA SEND	A3
GND (EIA SEND)	A4
TTY RDR CONTROL+	A5
TTY RDR CONTROL-	A6
EIA RCVE	A7
TTY KYBD+	A8
TTY KYBD-	A9
GND (EIA RCVE)	A10

1.5.2 High Speed Reader and Punch

A high-speed paper tape reader/punch can be a time-saving device during program preparation. The GIC1600 Series Microcomputer Systems can accommodate Remex (1733G Alton St., Santa Ana, Calif.) models

RAB6375 and RAF6375 or equal. Tally (8301 So. 180th St., Kent, Washington) model 1315C is a plug-compatible equivalent. EECO (1441 E. Chestnut St., Santa Ana, Calif.) models RP-9360 and RPF9360 are equivalents in performance, but be sure to specify plug-compatibility with Remex 6375 series when ordering. For detailed reader/punch description and operating procedures, refer to the manual of the specific high-speed reader/punch being used.

The interface cables should be less than 10 feet in length and should have the following pin assignments:

Signal	GIC1600 Lower Connector on I/O1600/1601 Module (3M Connector #3414-3000)	Remex Punch (P1) (Cannon BD25P	Remex Reader (P2) (Cannon DB24S)
HSP 0	B1	1	-
HSP 1	B2	2	-
HSP 2	B3	3	-
HSP 3	B4	4	-
HSP 4	B5	5	-
HSP 5	B6	6	-
HSP 6	B7	7	-
HSP 7	B8	8	-
GND	B9	18	-
PUNCH COMMAND	B10	11	-
GND	B11	25	-
DIRECTION	B12	10	-
PUNCH INPUT	-	-	
MODE SELECT	B13	14	-
PUNCH OUTPUT	-	-	
MODE SELECT	B14	15	-
TAPE/CHAD ERROR	B15	20	-
TAPE LOW	B16	21	-
SYSTEM READY	B17	13	-
GND	B18	23	-
PUNCH READY	B19	12	-
GND	B20	-	11
HSR DATA RDY	B21	-	9
GND	B22	-	13
HSR READY	B23	-	14
DRIVE LEFT	B24	-	17
HSR0	B25	-	1
HSR MODE SELECT	B26	-	10
HSR 1	B27	-	2
HSR 2	B28	-	3
HSR 3	B29	-	4
HSR 4	B30	-	5
HSR 5	B31	-	6
GND	B32	-	24
HSR 6	B33	-	7
HSR 7	B34	-	8

1.6 SYSTEM MODULES

The GIC1600 Series Microcomputer Systems are built from a family of stand-alone, functional computer modules. All of these printed circuit cards are backplane compatible and can be combined in various combinations to suit individual customer requirements. At present the GIC1600 Series Modules include the following:

1.6.1 Microcomputer Card

The Microcomputer Card contains the CP1600 Microprocessor and all the basic elements that are necessary for implementing a microcomputer system. It is packaged on a 9.75" x 9.25" x .062" printed circuit board with a 140 pin connector for mounting into the card cage.

The major functional units on the card are shown in Figure 19 and are composed of the following:

1) CP1600 Microprocessor
2) 10 MHz Crystal Oscillator
3) Clock Generator
4) External Branch Multiplexer
5) Data Bus Driver/Receiver
6) Address Register/Address Bus Control Logic
7) Control Bus Decoder/Driver
8) Real Time Clock (Optional)
9) Power Fail (Optional)
10) Initialization Address Selection Logic

The following description details the logic and operation of the MC1600/1601 Card with reference to Schematic Dwg. No. DS-MC-002.

CP1600 Microprocessor - This chip is the Central Processing Unit for the GIC1600 Series Microcomputer Systems. It is described in detail in the CP1600 Microprocessor User's Manual.

Oscillator - The card contains an 8 MHz crystal oscillator (Y1 & U2-8, 9, 12, 13) that may or may not be used. The output (OSC) is provided at the connector. In order to use the oscillator a jumper is inserted between J22 (MCLK) and J57 (OSC) on the backplane of the Microcomputer.

Clock Generator - The clock generator circuitry divides the oscillator frequency (MCLK) into the high level, high speed, non-overlapping two phase clocks that are necessary for proper CPU operation. In addition, timing clock CK4* is provided for the user. This signal corresponds to an internal CPU time slot (TS4) where most Data Bus and Control signal clocking is performed.

U19 is a 4-bit parallel-access, presettable shift register. Initially, a

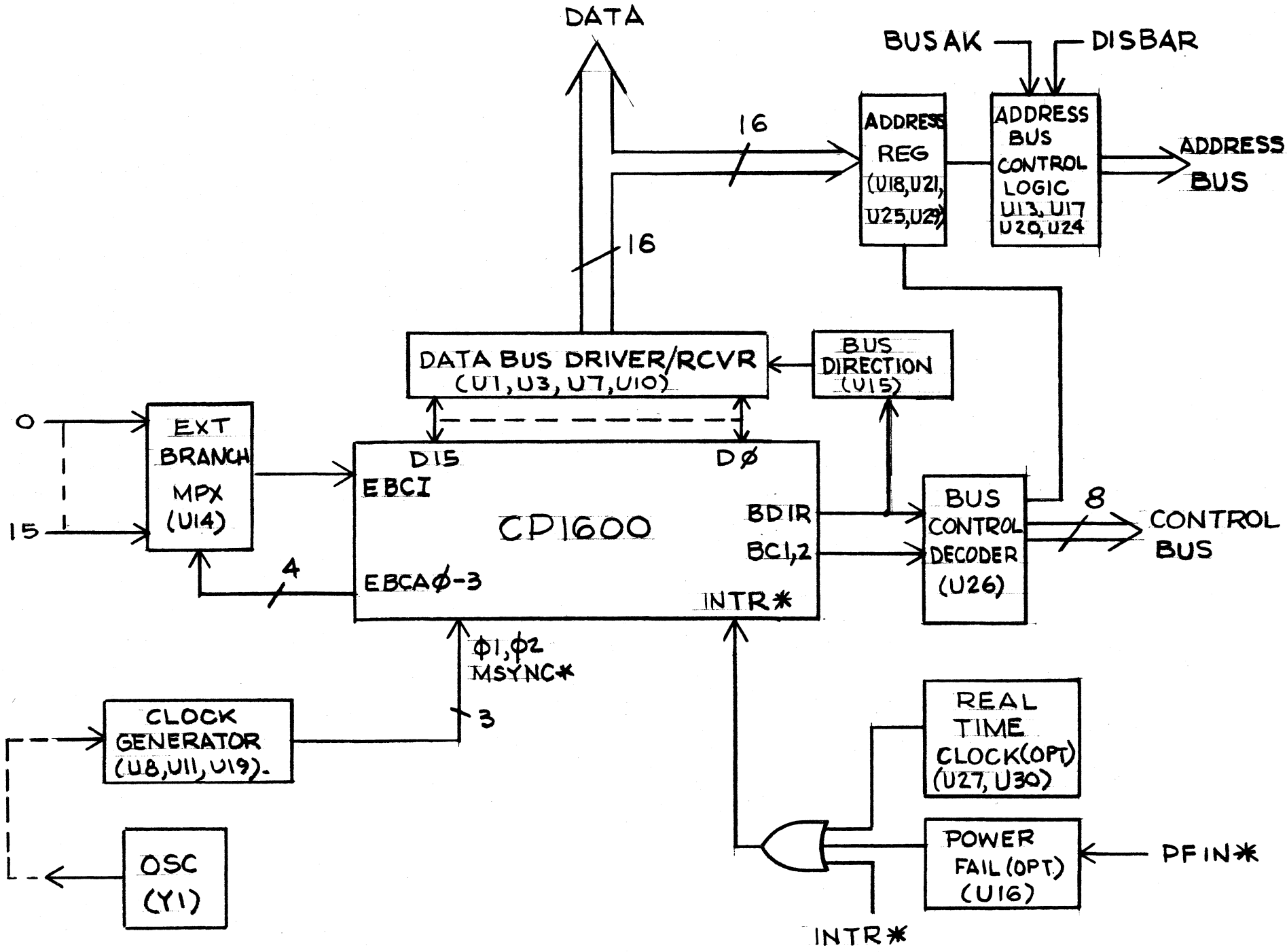


FIG. 19 MICROCOMPUTER CARD BLOCK DIAGRAM.

0111 pattern is loaded while MCLR* remains at its active low level. When MCLR* is high, the 0111 pattern is continuously recirculated in the shift register by the MCLK signal since the output of the last flip-flop is connected to the input of the first stage. CK1* and CK3* are ORed together via U11-3, 4, 5, 6 and buffered via U8-2, 7 to provide 01 for the CP1600. 02 is formed in a similar manner. U11 is cross-coupled to prevent both 01 and 02 from being high at the same time.

The card input, MCLR*, must be held low momentraily after the power supplies are stable. In the GIC1600 Systems, the MCLR push button on the Control Console would be depressed after the power supplies have been turned on. U22, 1-6, synchronizes the positive transition of MSYNC* with the falling edge of CK3*; this is needed to synchronize the CP1600 with the clock.

External Branch Multiplexer - In order to provide external branching capability, a sixteen to one multiplexer (U14) is supplied on this board. Each of the external branch conditions (EBC0-15*) is "0" active so that a true condition is sensed by a logical "0". The BEXT instructions perform the branch when the external branch inputs (EBC0-15*) are logic "0".

Data Bus Driver/Receiver - A 16-bit high speed driver/receiver comprised of U1, U3, U7 and U10 is provided to buffer each of the 16-data lines of the CPU. The output of the driver is the Data Bus which is connected to the memory and all the peripheral by U15. During CK1* both inputs to latch U15, 1-6, are held high. On the rising edge of CK1*, pin 1 will be set low if BDIR from the CPU is high, causing the latch output to latch high (CPU data to be transferred to the bus).

Address Register/Address Bus Control Logic - A sixteen bit wide address register (U18, U21, U25 and U29) is supplied on this card along with an output driver (U13, U17, U20 and U24). This register is clocked during CK4* if ADAR*, BAR* or INTAK* is decoded by U26. Access to the Address Bus for DMA operation is provided. A logical "0" input applied to DISBAR* will floate the address bus via U28, 10-11 and U22, 11-13 as long as the CPU has acknowledged BUSRQ* with BUSAK*.

A MAXADR* input is also supplied, but it is used exclusively by the control console card to force the storage of the current program counter into address 177777 for TRAP interrupt instructions associated with the On Line Debug Program (S16ODP) which is part of the Resident Operating System.

Bus Control Decoder/Driver - CPU control lines BC1, BC2 and BDIR are decoded by U26 to provide the Control Bus signals. Each output control line of the decoder has an open collector output so that DMA control can be achieved by wire "OR"ing. As the decoder is enabled by CK1* the Control Bus is valid only during CK2, CK3 and CK4.

Real Time Clock - The Real Time Clock option interrupts the CPU and supplies a starting address to the bus at specified intervals of time. U30 is a timer whose period can be set by user-selected components, R_A, R_B and C_T. U27 syncs the RTC interrupt request with the CPU.

Initially, the Interrupt Acknowledge (IACK) FF (U16) is "0". This allows an INTR* (Interrupt Request Not) via U9, 4-6, if there is a RTC or Power Fail (PF) interrupt request. The IACK FF is preset if either the RTC or the Power Fail (PF) is requesting an interrupt (U12, 1-3) and the INTAK* (Interrupt Acknolwedge Not) pulse comes from the Bus Control Logic (U9, 1-3) in response to the INTR*. INTR* is then raised high as soon as IACK is preset, again via U9, 4-6.

During the NACT* time, which follows the INTAK* in an interrupt sequence, (see the CP1600 Microprocessor User's Manual) the RTCRQ FF (U27, 8-13) will be set to "0". The IACK FF enables U6, 11-13, which ultimately allows the starting address of the RTC interrupt routine (RTCSTAD*) to be placed on the bus. The IAB* pulse which appears in an interrupt sequence gates the address onto the bus via U9, 11-13 and also clears U27, 1-6 via U12, 4-6, and U6, 8-10, to allow future interrupts. The TCI (Terminate Current Interrupt) software command will cause the CPU to generate a TCI signal on pin 26 of the CP1600, which will clear the IACK FF at the end of the RTC interrupt routine.

Power Fail Option - The Power Fail Option enables the user to save critical register information if he has either a core memory or a semiconductor RAM memory with battery backup. The Power Fail Sense Signal In (PFIN*) is synchronized to the CPU by being clocked into U16, 8-13 by the next available NACT* signal. As with the RTC above, PFRQ generates INTR*. When the IACK is preset, PFSTAD* (Power Fail Starting Address) is allowed onto the bus during IAB.

If the Real Time Clock Option or the Power Fail Option is not used, IPRO* (Interrupt Priority Out Not) is strapped to INTAK* (S1 to B0), so that INTAK* is enabled for the next card down the priority chain. U28-1 is strapped to PB which is +5 volts or the Power Fail Option is used, IPRO* is strapped to pin 8 of U12 (S1 to B1) and U28-1 is strapped to U16-6 (S0 to A1). Gate U12, 8-10, prevents the INTAK* pulse from acknowledging an interrupt further down the priority chain if RTC or PF requests an interrupt. Strap S0 to A1 is needed to mask out further interrupts down the priority chain while the IACK FF is set to a "1".

Gate U6, 1-3, decides in favor of the Power Fail if the Power Fail requests an interrupt within the time an interrupt is requested by the RTC and the IAB pulse presents the interrupt address to the bus.

Initialization Address Selection Logic - Two signals, STAD0*-1*, are

provided by the Microcomputer Card for generating the starting address of the user's Interrupt Branch Table. The user may tie on the backplane either or both of these lines to Data Bus lines to eliminate the need for extra open collector drivers. For example, if the main program begins at address 030000: DBL12*, 13*, should be tied to STAD0*, 1*, respectively. They are gated (U5, 1-6) by DISTAD* (Disable Starting Address Not). DISTAD* is provided for use by the Control Console Card. The Control Console Card lowers DISTAD* when it interrupts the CPU, so that the program counter is forced to an address specified by the Control Console Card during IAB rather than the one specified by the user on the CPU card. On startup, for example, the Control Console Card would place the entry address of the Resident Operating System on the bus and lower DISTAD* to the CPU card; thereby causing the GIC1600 Systems to always start by typing S16ODP V01A.

1.6.2 2K Memory Card

The 2K x 16 Memory Card provides the GIC1600 Systems with a 2K x 16-bit static random access memory (RAM). There are thrity-two 22 pin 256 x 4 static RAMs packaged on a 9.75" x 9.25" x .062" printed circuit board with a 140 pin I/O connector. These RAMs are TTL compatible and operate from a single +5 volt supply. Read access and write cycle times for the card are each specified as 550nsec maximum for all rated variations in power supply over the 0°C to 55°C temperature range.

The 2K x 16 Memory Card is shown in block diagram from in Figure 20. The major functional units on the card are:

1) Memory Matrix
2) Row Decoder
3) Card Decoder
4) Bus Control Logic
5) Read/Write Logic

The following description details the logic and operation of the RM1600 card with reference to Schematic Dwg. No. S-RM-012.

Memory Matrix - The static memory is organized as 2048 rows of 16 bits each. Each horizontal grouping of four RAM's (main row) contains 256 of the rows. Each of the four RAM's in a horizontal grouping (main row) contributes 4 of the 16 outputs. Addresses 0-7 (BADR0* - BADR7*) are applied to each group of 256 rows via U29 and U30.

Main Row Decoder - The desired horizontal group of four RAM's (main row) is selected by enabling the chip select inputs on the four RAM's via the 3-to-8 decoder U24.

Card Decoder - U35 and U40 decide if this is the particular memory card

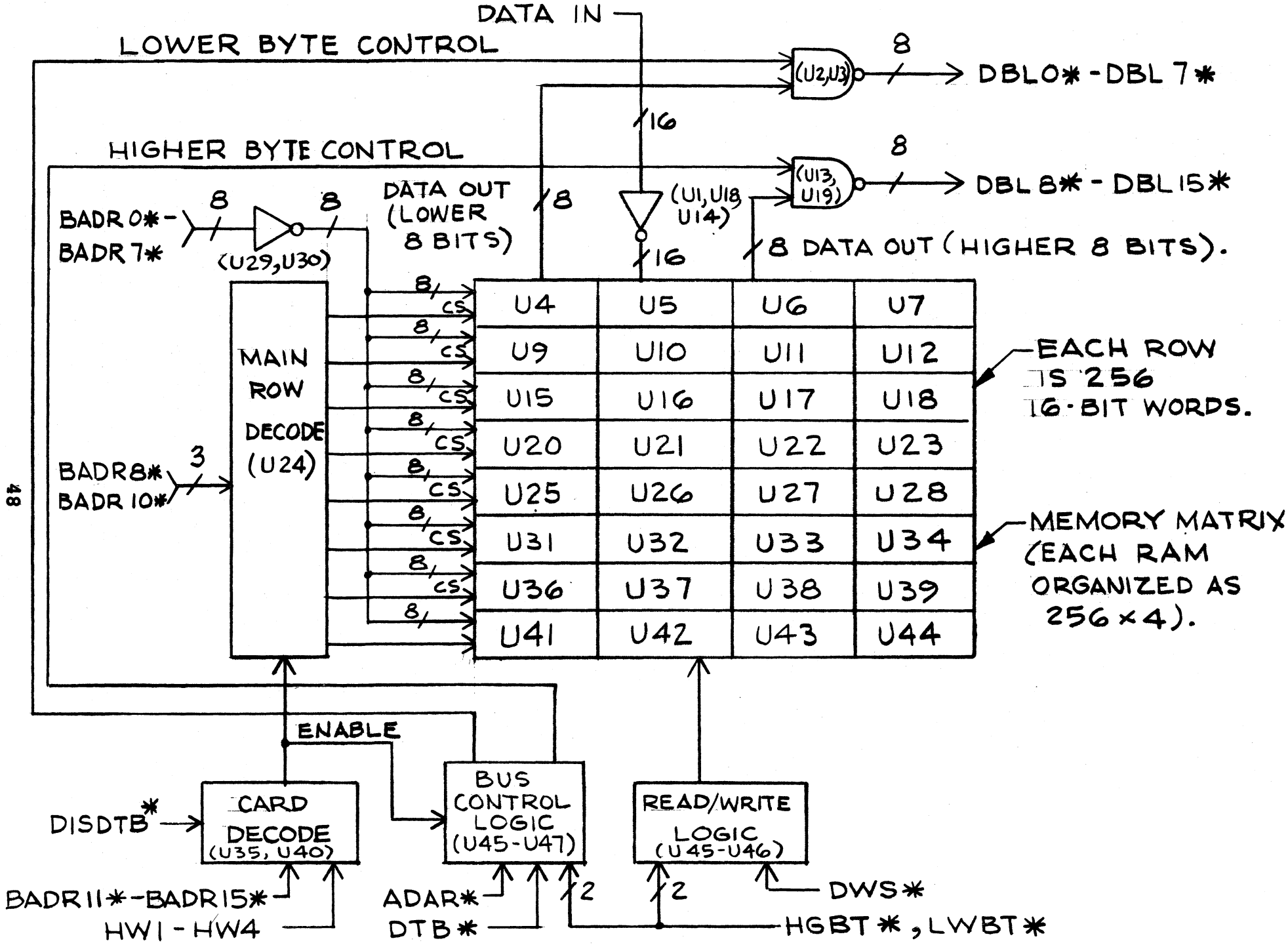


FIG. 20. 2K × 16 MEMORY CARD.

out of a number of memory cards that needs to be addressed. The output of U40 enables the row decoder (U24) and the output data bus control logic. The user is provided with four backplane pins HW1-4 on each general purpose slot that may be wired to either Vo (J41) or to AND. For example if the beginning address is 20000 HW3 should be tied to Vo while HW1, 2, 4 should be tied to gnd. For systems using 4-2K boards the backplane is wired for the lowest 8K (i.e., '0-'17777). Note that the standard memory module is strapped for the lower 32K, although the card can be modified for the higher 32K by strapping So to U and removing the jumper between So and L on the card.

Bus Control Logic - If the card is to be accessed, U46-6 enables U47-1 and U47-3. Higher and/or lower byte selection is done via U45, 8-13, enabling U47-2 and U47-4. If the MC1600 Microprocessor Card outputs a DTB* or an ADAR* signal requesting the use of the memory, U46-8 presents a pulse to U47-5 and U47-13. If the higher byte is selected, U13 and U19 then present DBL8*-DBL15* to the Data Bus. If the lower byte is selected, U2 and U3 then present DBL0*-DBL7* to the Data Bus.

Read/Write Logic - When the write command DWS* is presented from the MC1600 Microprocessor Card, U45, 1-6, select whether the higher or lower byte of a given 16-bit word is written into the memory.

Note: The basic MC1600/1601 Microcomputer Module does not use the byte select feature of the RM1600 Memory Module.

1.6.3 8K Memory Card

The 8K x 16 Memory Card provides the GIC1600 Systems with an 8K x 16-bit RAM for program and data storage. There are thirty-two 22-pin 4096 x 1 RAM's packaged on a 9.75" x 9.25" x .062" printed circuit board with a 140 pin I/O connector. As with the 2K x 16 Memory Card, read access and write cycle times for the card are each specified as 550nsec maximum for all rated variations in the power supplies over the 0°C to 55°C temperature range.

The 8K x 16 Memory Card is shown in block diagram form in Figure 21. The major functional units on the card are:

1) Memory Matrix
2) Main Row Decoder
3) Card Decoder
4) Bus Control Logic
5) Read/Write Logic
6) DC-to-DC Converter

The following description details of function and operation of the RM1601

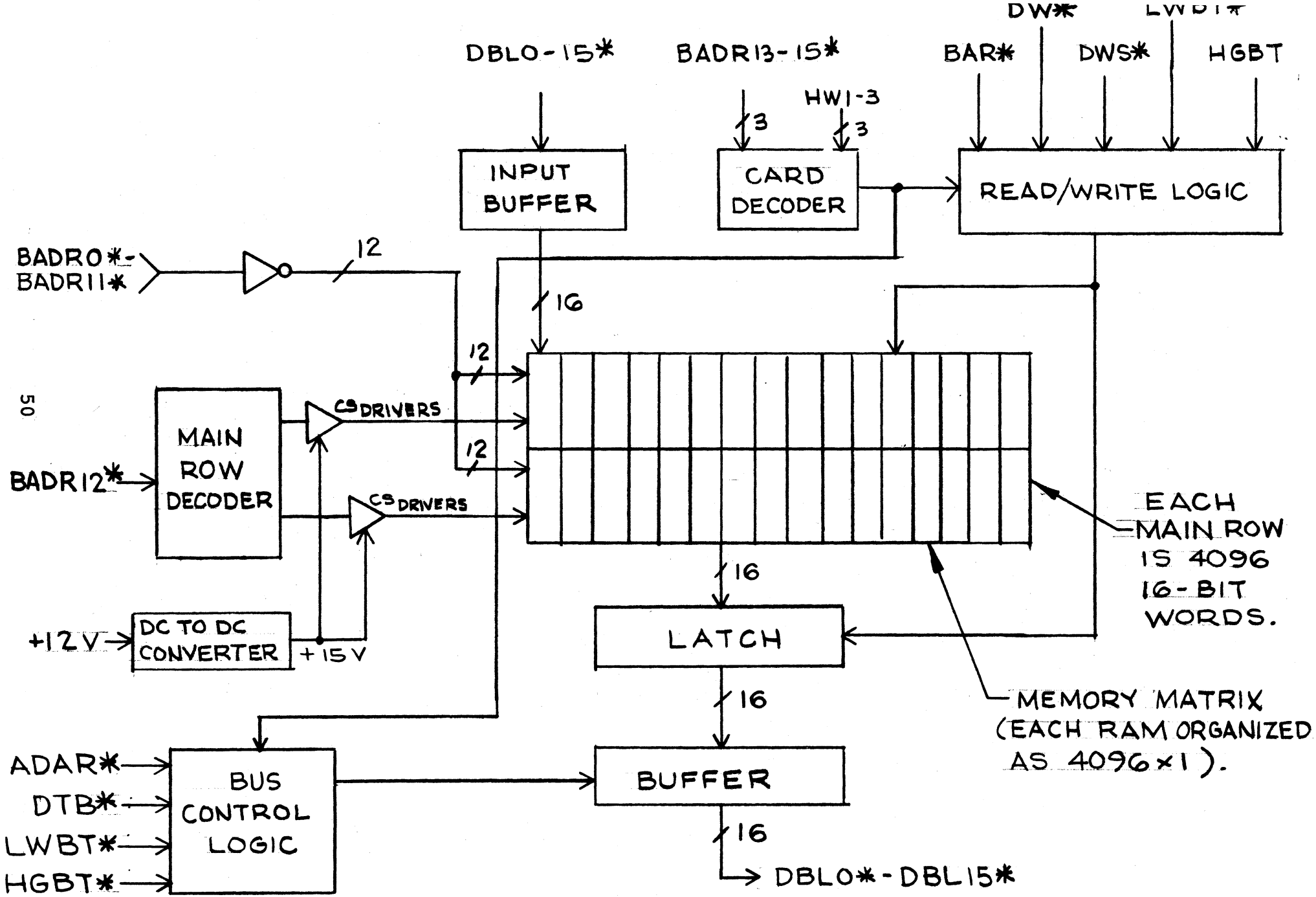


FIG. 21. 8K×16 MEMORY CARD.

Memory Module:

Memory Matrix - The memory is organized as 8192 rows of 16 bits each. Each of the two main rows (horizontal groupings of 16 RAMs) contain 4096 of the rows. Each of the 16 RAMs in a main row contributes 1 of the 16 outputs. Addresses 0-11 (BADR0* - BADR11*) are applied to each group of 4096 rows.

Main Row Decoder - The desired horizontal grouping of 16 RAMs (main row) is selected by enabling the chip select inputs on the 16 RAMs. BADR12* is steered via LWBT* and HGBT* (as on the 2K x 16 Memory card) to two chip select drivers for the top row and BADR12 is steered via LWBT* and HGBT* to two chip select drivers for the bottom row.

Card Decoder - The card is selected via gating similar to the RM1600 2K x 16 Memory Card. HW1-3 are provided to select the particular 8K memory address block that the module will occupy. For example, if the beginning address is 20000, HW1 should be tied to Vo while HW2 & 3 should be tied to GND. For systems using 1-8K module the backplane is wired for the lowest 8K in slot 3.

Bus Control Logic - The bus is driven via logic similar to the RM1600 2K x 16 Memory Card discussed above.

Read/Write Logic - In addition to providing the read/write controls to the RAMs, this logic provides a NACT* signal to latch the output data after a read operation since the output data is valid only for a certain amount of time after the chip select signal falls.

DC-to-DC Converter - A 12 volt to 15 volt converter is required to power the chip select drivers.

1.6.4 Control Console Card

The Control Console Card provides the interface between the Control Panel and the CP1600 Microprocessor. It is packaged on a 9.75" x 9.25" x .062" printed circuit board and a 140 pin connector is located on the edge of the card for mounting into the card cage supplied. Connection to the control panel is achieved with two flex cables and are connected to another edge-board connector mounted on the Control Console Card.

Thie card contains control logic to handle all front panel commands as well as the required interrupt logic to interface with the CPU. The basic functional units on the card are shown in Figure 22 and consist of the following:

1) Data Multiplexer and Data Bus Driver
2) Switch Register
3) Status Register

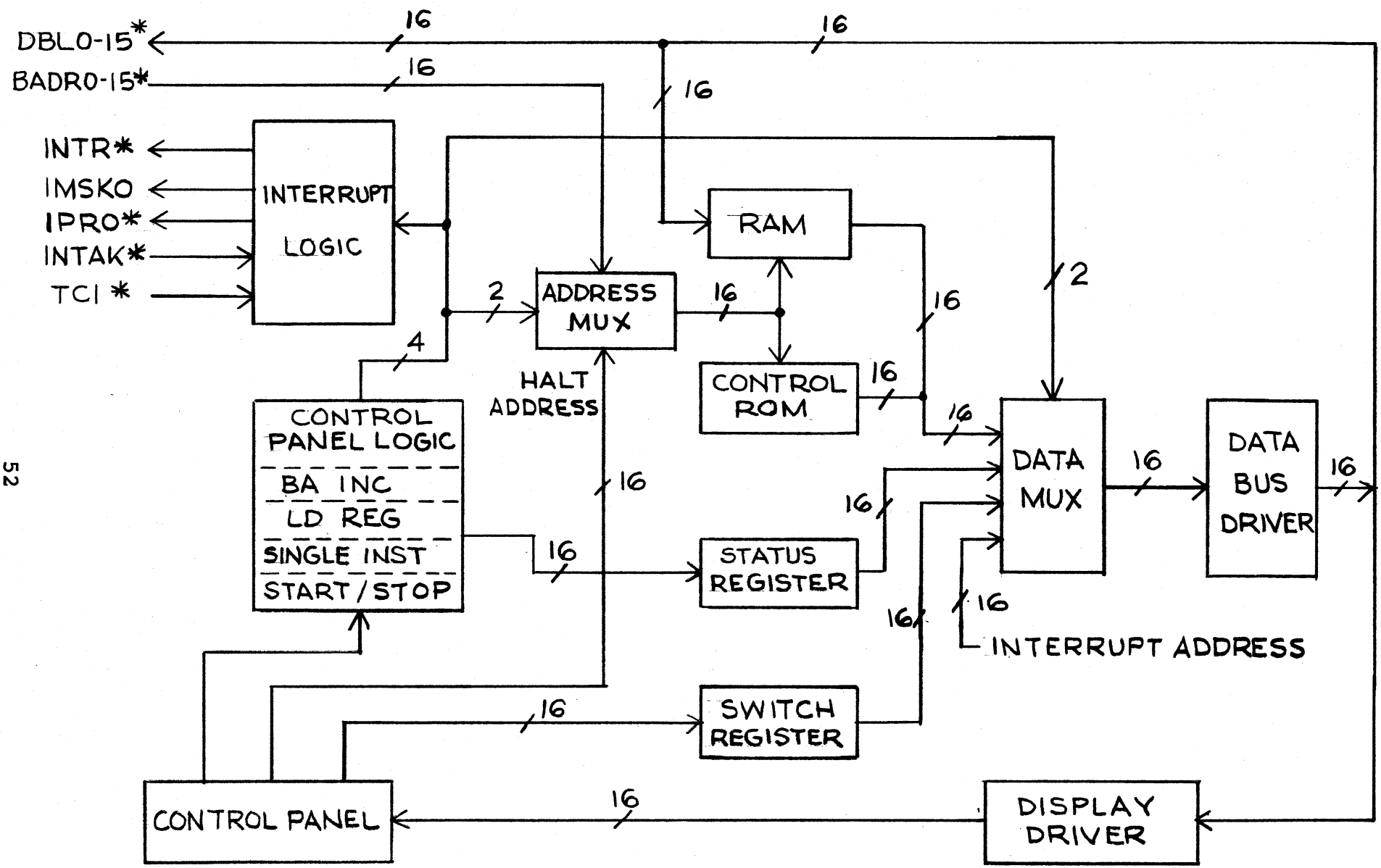


FIG. 22. CONTROL CONSOLE CARD AND PANEL BLOCK DIAGRAM.

4) Address Multiplexer
5) Control ROMs (Resident Operating System)
6) Scratchpad memory (RAM)
7) Control Panel Logic
8) Interrupt Logic

The following description details the function and operation of the CC1600 Control Module with reference to Schematic Dwg. No. DS-CC-007.

Data Multiplexer and DAta Bus Driver, Switch Register and Status Register

The data multiplexer comprised of U24, U23, U19, U18, U14, U13, U4 and U3 enables data to be placed on the data bus as a function of its two control lines, A and B:

A	B	DATA
0	0	ROM/RAM Output
0	1	Switch Register Data
1	0	Status Register Data
1	1	Interrupt Address (170000 for ODP)

A=B=1:

When the Master Clear Switch (MCLR*) is depressed after power is applied, MAXADR* at U48-8 goes low. This, in turn, presets U44, 8-13 to a "1", forcing the multiplexer control signals (A at U33-11 and B at U33-3) to a "1". The Interrupt Address for the Resident Operating System (170000) will now be presented to the inputs of the data bus drivers U29, U28, U9 and U8.

The first control signal from the CP1600 after MSYNC* goes high is an Interrupt Address to Bus (IAB*). (See Sec. 2.3 of the CP1600 Microprocessor User's Manual). IAB* is buffered by U48, 1-3. It then passes thru the Bus Control OR gate U52, 1-6, and allows the Interrupt Address onto the bus by gating on the data bus drivers. During the NACT time that follows the IAB on the Bus Control lines, the Interrupt address is stored in R7 (the Program Counter) in the CP1600. Then the CP1600 outputs the Program Counter to the Bus Address Register on the MC1600 card and the On-Line Debug Program begins. (Part of Resident Operating System). ODP clears its pseudo-registers located in the ODP RAM (see Figure 2 - Memory Map).

A=B=0:

If data from the output of the 16-bit RAM comprised of U2, U7, U17 and U22 or one of the 10-bit On-Line Debug ROMs comprised of U27, U37, U42, U47, U56 and U62 (the higher six bits are tied to a "1") are to be placed on the data bus, the control signals A and B must be "0". A and

B being low is the normal quiesient state of the multiplexer; if the switch register, the status register or the interrupt is not requesting service by the user, A and B will be "0". Logically, to have U33-3 and 11 low, U33-1, 2, 12, 13 must be high. U33-2 and 12 are high since U44, 8-13 will not be set. U33-1 and 13 will be high since address 177XXX (which is required by the switch or status registers being steered thru the multiplexer) has not been decoded at U38-9. The "1" at U38-9 is ORed thru U43, 4-6, U-43, 11-13 and U43, 8-10 and presented to U33-1 and 13. Thus, the multiplexer will be steering RAM/ROM data in the data bus drivers unless otherwise directed.

A=1; B=0:

The switch register of the control panel is used for data and address entry. It can be inspected by the CP1600 by a MVI 177200, R0 type instruction. When the first four high order bits appear as 1's on the address register, U58-2 goes low enabling U38, U38-9, in turn, will go low for the first seven high order bits being 1. U43-6 will then go low for 1 111 111 0XX XXX XXX. If BADR7 is high, B will be one and A will be zero. (The Interrupt Address FF, U44, 8-13, is not set and both U33, 1-3 and U33, 11-13 are low. Note all addresses between 177200 and 177377 will decode in an identical manner and thus all are reserved for the switch register). With A=1 and B=0, the multiplexer gates SWR0 thru SWR15 onto the data bus.

A=0, B=1:

The status register of the Control Console Panel is defined below:

8	7	6	5	4	3	2	1	0	Bit
Single Instruct	Trap	Stop	BA INCR	LD REG					

Bit 3: 1 — 0 thru 7: R0-R7

Bit 0-2: 0 — 0: SW, 1: BA, 2: BD, 3: ML, 4: MH, 5: FN, 6: F1, 7: F2 — See Section 1.3.2

Note: All the bits in this register are low active; i.e., a "0" in bit 8 indicates that the Single Instruct lever is depressed. Bits 0-3 indicate to the Control Console Card which register is to be displayed.

This status register can be inspected by the CP1600 by a MVI 177000, R0 type instruction. The decoding is the same as described above for the switch register, except that A will be zero and B will be one, since BADR7 is now 0 and not 1. (Note again that all addresses between 177000 and 177177 will decode in an identical manner and thus all are reserved for

the status register). U15 encodes the 8 register lines to 3 bits. With A=0 and B=1, the multiplexer gates the status register onto the bus.

The data multiplexer is enabled for the ROM/RAM, the switch register and the status register since they all require 17XXXX to be decoded which is done by U52, 8-13. Presenting the Interrupt Address to the data bus also requires 17XXXX to be decoded; MAXADR* created by the interrupt places 177777 on the bus. Thus, the data multiplexer is enabled for all four uses.

U40, U45 and U50 drive the data display lamps in the Control Console Panel. U52, 1-6, serves as an OR gate to allow DTB*, IAB* and ADAR* to strobe the multiplexed data onto the bus through gates U8, U9, U28 and U29 when commanded by the CP1600.

ADDRESS MULTIPLEXER

Multiplexer U36, U41, U46 and U51 directs the Bus Address Register onto the Control Console Card in the Run mode (if the machine is not in the Single Instruct mode). In the Halt mode, the multiplexer places all 1's onto the card, except for the least four significant bits, which are the encoded register lines that are used as addresses to display the top 16 locations of the RAM contents. When halted, the contents of the R0 to R7 registers and the contents of the SW, BA, BD, ML, MH, FN, F1 and F2 registers can thus be displayed. The Halt signal forces 1's on multiplexer U51's output. U52, 8-13, decode these 1's to a 0, which enables the data multiplexer. If the Control Panel is not disabled by DISCC* (Disable Control Console Not), the high level of the Halt signal on U61-2 passes to U52-6, which enables the data bus drivers and thus allows the RAM contents to be displayed during halt, if selected.

In the Single Instruction mode, an interruptable instruction is placed on the data bus in place of certain non-interruptable instructions. When this occurs, DISDTB* at U21-10 goes low forcing U51-1 high, which places 0's on all the outputs of U51. This disables the data multiplexers and forces all their outputs low. When DTB enables the data bus drivers, the data multiplexer will not contribute any data to the bus, thereby freeing the bus for the NOP instruction needed for the Single Instruction mode.

CONTROL ROMs

The firmware necessary for the Resident Operating System and the On-Line Debug Program is incorporated in ROMs U27, U37, U42, U47, U56 and U62. They are all supplied with input addresses BADR0-8; each chip is selected by decoder U38 using BADR9-11. The ROM outputs are routed thru the data multiplexer as discussed above.

SCRATCHPAD MEMORY (RAMs)

The top 16 locations of the Scratchpad Memory formed by RAMs, U2, U7, U17 and U22 store the R0 to 7 registers and the SW, BA, BD, ML, MH, FN, F1 and F2 registers as shown in the Memory Map in Figure 2. The other 240 locations are used for temporary storage for the On-Line Debug Program. The RAMs are all supplied with input addresses BADR0-7; all chips are selected by addresses BADR8-11 being all 1's via U63, 8-10. The RAM outputs are routed thru the data multiplexer as discussed above.

CONTROL PANEL LOGIC

The Control Panel Logic in conjunction with the On-Line Debug Program enable the user to inspect and modify CPU registers, memory locations and I/O device registers, and to read and punch paper tapes.

When the CPU is halted, the control panel is enabled by U50, 3-4. U30 debounces and latches LDREG, BA INCR, START/STOP and PCINH. If LDREG or BAINC is depressed when the machine is halted, an interrupt request is sent to the CPU via U34 and U57, 11-13. The CPU is also started in a delayed fashion via U25, 1-6 and delay inverters U32, 12-13 and U12, 1-4 so that the interrupt request is present at the CPU when the start command is given. The appropriate routine is then selected and executed under control of the ODP program. The machine is then halted.

If the START/STOP switch is depressed and the machine is halted, the negative transition on U32-8 will cause a negative transition on U39-8. This will be ORed through U25, 1-6 causing the machine to start. After the machine is started, the HALT signal on U53-13 will go low, resetting the STRT FF, U39, 8-13.

If the START/STOP switch is depressed and the machine is running, U39-6 will go low causing an interrupt and lowering STOP* via U5, 3-6. A routine in ODP will then be entered which will note that STOP* is active and dump out the CPU's internal registers and the status word into ODP's RAM pseudo-registers before halting. In order to also update the pseudo registers when the machine is halted via an instruction, a circuit is needed to detect a halt while running. When the machine is started, the low-going signal on U39-8 also resets U49, 6-12. When a halt occurs in the program, U44, 1-6 is set to a "1" by the halt. U44-6 going low restarts the programs via U25, 1-6, interrupts the CPU via U34, and lowers STOP* in the status register via U5, 3-6, which tells ODP to write the CPU's registers and status word into ODP's RAM and halt. This last halt will not cause an interrupt again because U44-6 going low from the halt in the program disarmed the D input of U44, 1-6, via U5-2 and 12 and U49, 8-12. U49, 8-12 is also set to a "1" via U5-1, 12 & 13 when the machine is stopped via the

STOP/START switch to prevent the halt while running FF, U44, 1-6 from acting when it is not needed.

When PCINH is depressed, the program counter is inhibited from being incremented to enable repetitive execution of a one word instruction. In this case, U21-12 drives PCIT* low until the switch is reset.

PCIT* will also go low when the CPU decodes a TRAP instruction, which enables ODP to provide program breakpoints. The negative transition of PCIT* will fire the 500ns one-shot U26, presetting the TRAP FF U16, 1-5. U33-8 will then fall causing an interrupt and telling the status register a TRAP has occurred. After the TRAP routine is executed, the TCI* (Terminate Current Interrupt Not) pulse clears the TRAP FF.

Note that when PCINH is depressed on the control panel, the one-shot U26 is also fired setting the TRAP FF. Since a TRAP routine is unwanted, however, U32-10 does not allow the TRAP signal thru gate U33, 8-10 and also resets the TRAP FF thru U53, 4-6.

To inspect the state of the microcomputer after each instruction, place the CONT/SINGLE INSTR switch down. The program can now be stepped through using the START/STOP SWITCH. Every time the START/STOP switch is depressed, an interrupt is generated which executes that one instruction, updates the pseudo-registers and the status word, and then halts the machine. For most instructions, this is done via pin 2 of state decoder U59 going low. U59 is made active only in the SINGLE INSTR mode via pin 12.

U61, 8-13, decodes a MVO instruction; U60 decodes a CONTROL-type instruction (EIS, DIS, CLRC, SETC) and U25, 8-13 decodes a SHIFT-type instruction. These particular instructions are not interruptible in the CP1600 microprocessor. The state flow logic driving U59 allows the user to single step even these non-interruptible instructions. After executing the instruction, the logic forces an interruptible instruction, GSWD R4 onto the bus via U57 and U21, 10-11. The interrupt put out by U59-5 can then be recognized. The control console card's RAM is again then updated and the machine is halted. With this logic there are only two instructions that cannot be single-stepped through: TCI and HLT. The machine will halt on the first interruptible instruction following a TCI or HLT.

INTERRUPT LOGIC

If the interrupt priority is enabling interrupts for the Control Console Card, ISMI on U48-4 and U31-9 will be low (not masked). Since the IACK FF U16, 8-13, is not set, U57-13 will be high, allowing the interrupt request

when an interrupt occurs at U34-8. The acknowledge pulse (IPRI*) which will occur in response to the interrupt request, passes thru gates U63, 1-3 and U48, 8-10, setting the IACK FF. It also forces the address latch on the MC1600/1601 card to the maximum address where the present PC will be stored for the return to the program. The set IACK FF turns off the interrupt request via U31, 8-10 and masks out lower priority devices from interrupting via U21, 5-6. The IACK FF also lowers the DISTAD* signal to prevent the machine from starting at the address specified on the CPU card, rather than at ODP's starting address generated on the Control Console Card. If a higher priority device interrupt service routine is completed, the IACK FF will again be set high when the higher priority device program finishes its routine and issues a TCI. The TCI pulse will clock the high (formed by the AND of IMSKI being high due to the higher priority interrupt and IACK being set) on its D input to its output. This allows the Control Console Card to finish its interrupt routine and keep lower order priorities masked out. At the end of the routine, the TCI pulse will clear the IACK FF.

1.6.5 Input/Output Card

The Input/Output card provides full duplex buffered communication interfacing between the Microcomputer and a Teletype, a High Speed Reader/Punch combination and/or an EIA device, such as a T.I. Silent 700 Data Terminal or the G.E. Terminet printer.

The Teletype and the EIA interface is asynchronous; a Universal Asynchronous Receiver/Transmitter (UAR/T) is used. When receiving data, The UAR/T converts an asynchronous serial character from the Teletype into a parallel character required for transfer to microcomputer bus. This parallel character can then be gated through the bus to the microprocessor registers or to some other device's register. When transmitting data, a parallel character from the bus is converted to a serial data stream for transmission to the Teletype Punch/Printer. The two data transfer units are independent; therefore, they are capable of simultaneous two-way communication (full duplex operation).

The high speed reader/punch works on a synchronous basis under the control of the Input/Output card.

Figure 23 is the I/O1600/1601 Input/Output Module Block Diagram. The basic functional units of the card are:

1) Data Bus Multiplexer and Status Register Multiplexer and Control Logic.
2) TTY/EIA Reader/Keyboard Receiver & Receive Status Registers
3) TTY/EIA Punch/Printer Transmit and Transmit Status Registers
4) High Speed Reader & HSR Status Registers
5) High Speed Punch & HSP Status Registers
6) Interrupt Logic

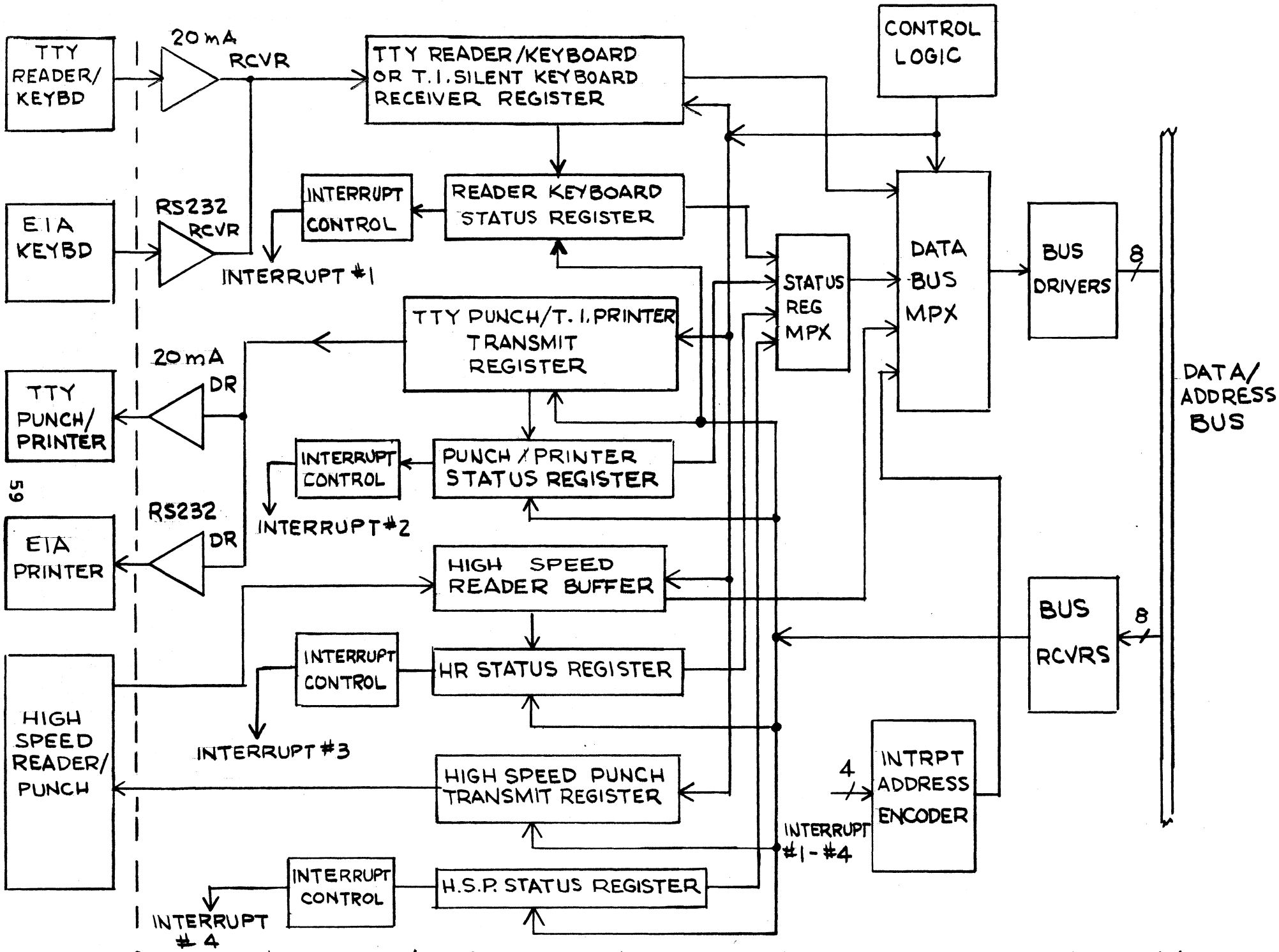


FIG. 23. I/O 1600/1601 INPUT/OUTPUT MODULE BLOCK DIAGRAM

The receiver registers, transmitter registers, and status registers are each assigned an address. These addresses are assigned as follows:

ADDRESS	FUNCTION
167770	High Speed Reader Status - 8-Bit Wide
167771	High Speed Reader Data - 8-Bit Wide
167772	High Speed Punch Status - 8-Bit Wide
167773	High Speed Punch Data - 8-Bit Wide
167774	TTY Reader/Keyboard Status - 8-Bit Wide
167775	TTY Reader/Keyboard Data - 8-Bit Wide
167776	TTY Printer/Punch Status - 8-Bit Wide
167777	TTY Printer/Punch Data - 8-Bit Wide

The formats for the contents of each of these addresses are as follows:

Address 167770 - High Speed Reader Status Register

7	6	5	4	3	2	1	0
0	0	0	0	Interrupt Enable	Reader Enable	Error Summary	Ready

BIT	NAME	FUNCTION	Read Only Bits
0	Ready	Set to a logical "1" when a character is available in the Reader Data Buffer. Cleared by pressing the MCLR switch or by referencing the Reader Data Buffer.	
1	Error Summary	Set to a logical "1" whenever the Run-Load switch of the reader is in Load or when there is no tape or torn tape in the reader. Cleared when the error condition is corrected or when the MCLR switch is depressed.	
2	Reader Enable	A logical "1" drives tape to the left to read one character. Cleared by pressing the MCLR switch. Also cleared by the Ready signal indicating character available.	
3.	Interrupt Enable	Set to logical "1" to allow Ready = 1 or Error = 1 to cause an interrupt. Cleared by pressing the MCLR switch or by disabling interrupts in the program.	

Address 167771 - High Speed Reader Data

15	7	0
0	DATA	

Bits 0 thru 7 hold the data from the Reader. This buffer can only be read. When this data buffer is referenced, Ready is cleared.

Address 167772 - High Speed Punch Status Register

7	6	5	4	3	2	1	0
0	0	0	0	Interrupt Enable	0	Error Summary	Ready

Read Only Bits

BIT	NAME	FUNCTION
0	Ready	A logical "1" indicates unit is ready to accept data to be punched. Cleared by pressing the MCLR switch; also cleared while the data is being punched. It is reset when new data can be loaded.
1	Error Summary	A logical "1" indicates one of the following conditions: a) Perforator Run-Load switch in Load. b) Tape from supply is loose, broken or tight. c) Chad level has reached a predetermine height. d) Tape supply is low. It is cleared when the MCLR switch is depressed or when the error condition is corrected.
2	Not used	
3	Interrupt Enable	Set to logical "1" to allow Ready = 1 or Error = 1 to cause an interrupt. Cleared by pressing the MCLR switch or by disabling interrupts in the program.

Address 167773 - High Speed Punch Data

15	7	0
0	DATA	

Bits 0 thru 7 hold the character to be punched. In this buffer, data can only be stored, not read. Storing data in this buffer causes that data to

be punched, which clears Ready until the data is punched and the device can once again accept data.

Address 167774 - TTY Ready/Keyboard Status

7	6	5	4	3	2	1	0
0	0	0	0	Interrupt Enable	Reader Enable	Error Summary	Ready

Read Only Bits

BIT	NAME	FUNCTION
0	Ready	Set to logical "1" when a character is available in UAR/T Data Buffer. Ready is cleared when data register is read into the CPU or when the MCLR switch is depressed.
1	Error Summary	True signal indicates that a Parity Error, Framing Error, or Over-Run has occurred; cleared when the MCLR switch is depressed or when the error condition is corrected.
2	Reader Enable	Set to a logical "1" to enable the Paper Tape Reader (not the keyboard) to read a character. Reader Enable is cleared when a legitimate start bit is detected or when the MCLR switch is depressed.
3	Interrupt Enable	Set to allow an interrupt when Ready or Error is set to a logical "1". Cleared by pressing the MCLR switch or by disabling interrupts in the program.

Address 167775 - TTY Reader/Keyboard Data

15	7	0
0	DATA	

Bits thru 7 holds the coded data from the TTY Reader or Keyboard. This buffer is for read only operation. When data register is referenced, Ready is cleared.

Address 167776 - TTY Printer/Punch Status

7	6	5	4	3	2	1	0
0	0	0	0	Interrupt Enable	0	0	Ready

Read Only Bit

BIT	NAME	FUNCTION
0	Ready	Set to logical "1" when Punch/Printer is available to accept data. It is cleared when Printer/Punch Data Buffer (UART/T) is loaded and reset when new data can be loaded. It is also cleared when the MCLR switch is depressed.
3	Interrupt Enable	Set to logical "1" to allow Ready = 1 to cause an interrupt. Cleared by pressing the MCLR switch or by disabling interrupts in the program.

Address 167777 - TTY Punch Data

15	7	0
0	DATA	

Bits 0 thru 7 hold the character to be punched. In this buffer, data can only be stored, not read. Storing data in this buffer causes that data to be printed and punched, which clears Ready until the device can once again accept data.

Typical flow diagrams showing computer interaction with the keyboard, reader, punch and printer are shown in Figures 24, 25, 26.

The following description details the function and operation of the I/O1600/1601 Input/Output Module with reference to Schematic Dwg. No. DS-IO-017. Data from the TTY Reader/Keyboard Buffer, the High Speed Reader Buffer, the Status Registers, and the Interrupt Address are fed into a multiplexer. The multiplexer control is determined by which device is selected by the software program.

The input/output interface has four channels of interrupts; two for the receiver section (High Speed Reader and TTY Reader) and two for the transmitter section (High Speed Punch and TTY Punch). These four circuits operate independently, except that the receiver takes priority on simultaneous interrupts. The high speed reader and punch have a higher priority than the TTY reader and punch.

Four interrupt vector addresses are generated by each interrupt section. The interrupt addresses can be changed by selection of jumpers provided on the interface board.

The input/output interface has a 20 mA current loop, electrical interface for the TTY paper tape Reader control, and a control line to energize the paper tape Reader Run Relay which is an internal modification to the

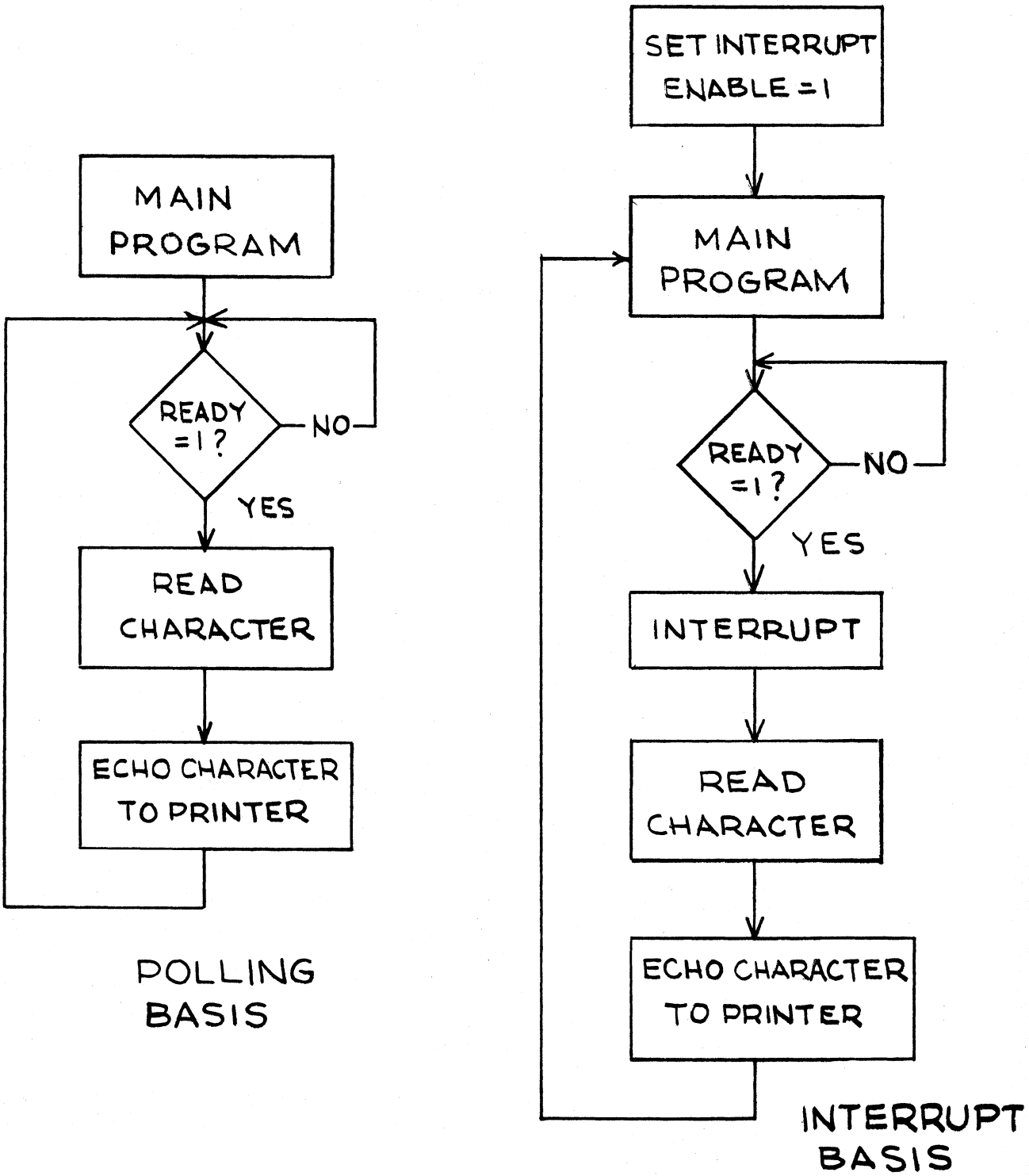


FIG.24. TTY/EIA KEYBOARD FLOW DIAGRAM.

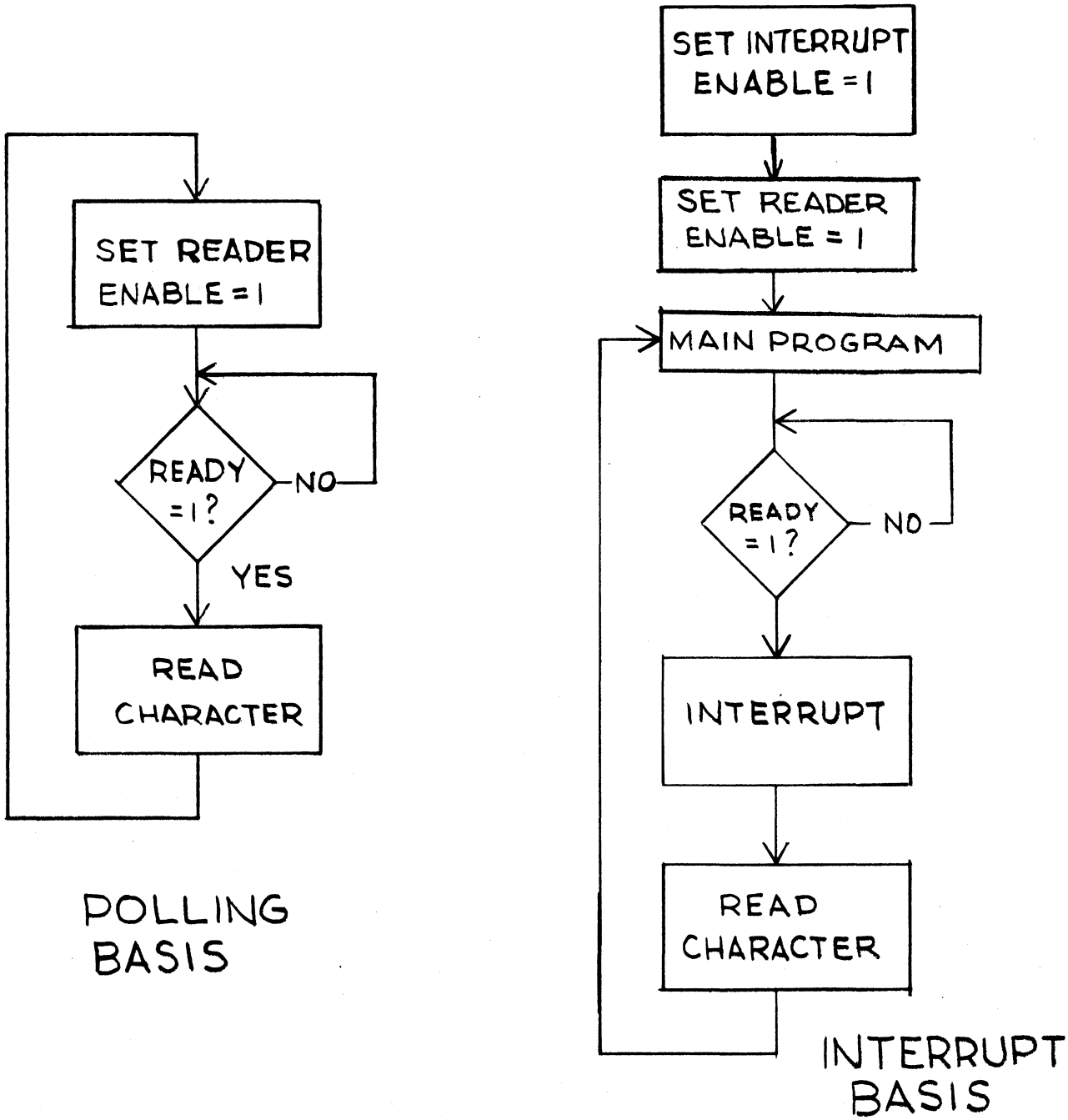
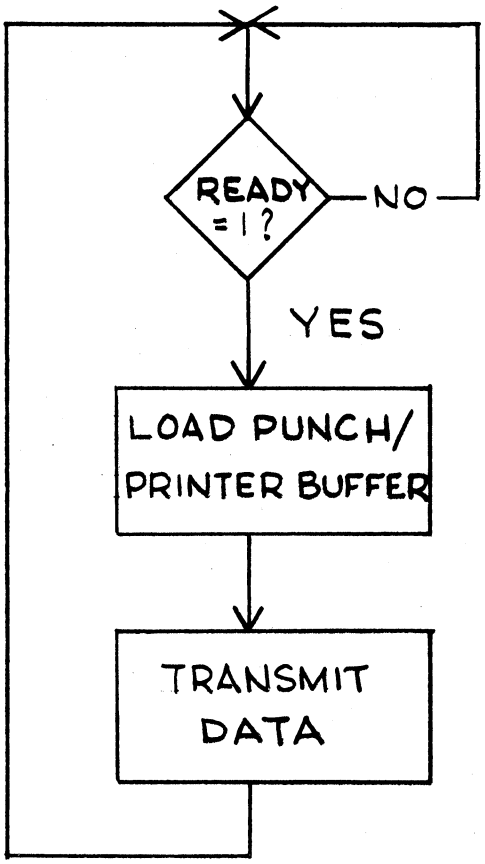
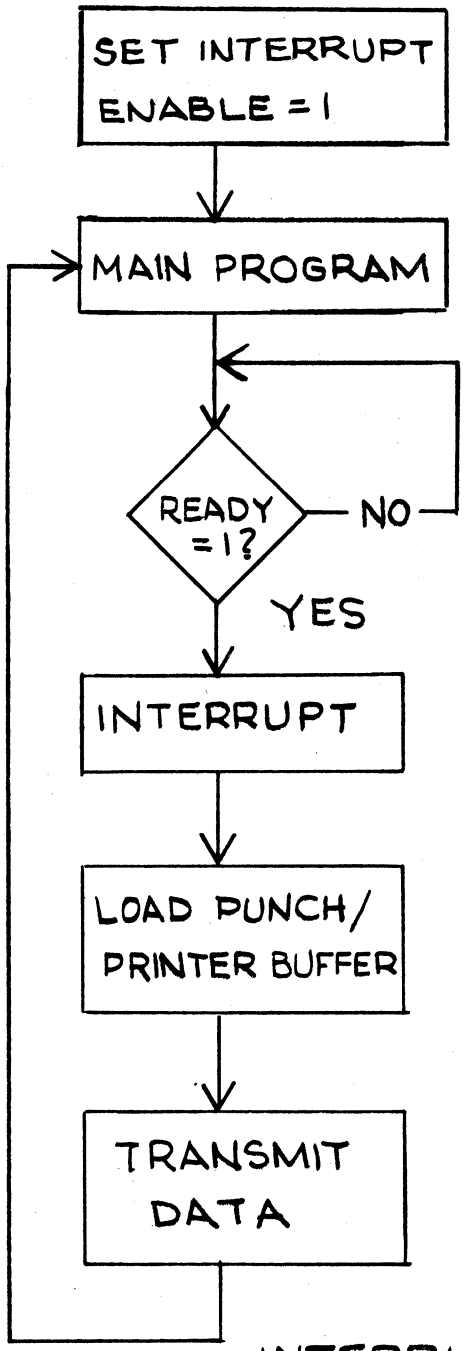


FIG. 25. HIGH SPEED/TTY READER FLOW DIAGRAM.



POLLING BASIS



INTERRUPT BASIS

FIG. 26. TTY PUNCH /HIGH SPEED PUNCH AND TTY/EIA PRINTER FLOW DIAGRAM.

ASR 33 explained in Section 1.5.1. This relay allows computer control of the reader during on-line operation.

A.T.I. Silent 700 Data Terminal Model 733 or equivalent EIA-Compatible terminal, which can provide selectable printing speeds of 30 characters per second as well as keyboard operation, can be controlled by the input/output interface board. A Remex Model 6375 Reader/Punch Combination (or equivalent - see Section 1.5.2) capable of reading paper tape at 300 characters per second and punching tape at 75 characters per second, can also be driven by the input/output interface. The basic Card Dimensions of the input/output interface is 9.75" x 9.25" x .062" with a 140 Pin Input/Output connector.

Data Bus Multiplexer & Status Register Multiplexer and Control Logic

U13, U8, U7 and U12 form the Data Bus Multiplexer. The multiplexers are enabled via pins 1 and 15 for all status register data (addresses 167770, 167772, 167774, 167776), for the low and high speed reader data registers (167771 and 167775), and for interrupts. U25, 8-11 provide the required ORing function of the address decodes and the interrupt signal. Data enabled to be placed on the data bus is also a function of its two control lines, A and B:

A	B	DATA
0	0	Low Speed Reader (UAR)
0	1	Status Register
1	0	High Speed Reader
1	1	Interrupt Address

A = B = 1:

The quiescent state (when address 16777X is not being decoded) of the control lines is A = B = 1. This allows one of four interrupt addresses to be placed on the data bus when an interrupt occurs. Although the interrupt addresses can be changed via changing the straps, the standard strapping from the factory is assigned as follows:

Interrupting Device	Interrupt Address
High Speed Reader	0000010
High Speed Punch	0000012
Low Speed Reader	0000014
Low Speed Punch	0000016

A = B = 0:

Data from the Low Speed Reader is held in the low speed receiver register,

which is the holding register of Universal Asynchronous Receiver/Transmitter (UART) U3, 5-12.

Gates U30, U31, U25 (1, 2, 12, 13), U11 (8, 9), and U39 (5, 6) decode the 16777X part of the reader/punch addresses defined in the previous section. U44 decodes the last three bits into eight lines. If the LSR (167775) is selected, U44-3 would go low, forcing control line B (U42-6) to 0. Control line A (U46-8) is also 0, since U31-6 is 0 (16777X is decoded) and U44-6 is 1 (167771 is not decoded).

The LSR data thus selected will be presented to the data bus when BDTB* on U10-10 goes low, enabling data bus drivers U1 and U5.

A = 1, B = 0:

Data from the high speed reader are presented to the data bus drivers when address 167771 is decoded on U44-6, causing A (U46-8) to go high and B (U42-6) to go low.

A = 0, B = 1:

Data from the status multiplexers U27 and U28 are presented to the data bus drivers when address 16777X is decoded on U31-6, but address 167771 (high speed reader) or address 167775 (low speed reader) is not decoded. These conditions will force A to be 0 and B to be 1. (The punch data addresses never enable the data multiplexers and thus can be ignored).

The four (LSR, LSP, HSR, HSP) status registers will place their data (see bit assignments in previous section) onto four least significant data bits of the data bus as follows:

A	B	STATUS REGISTER	ADDRESS
0	0	High Speed Reader	167770
0	1	Low Speed Reader	167772
1	0	High Speed Punch	167774
1	1	Low Speed Punch	167776

If the HSR status is required, for example, both A for the status mux (U46-11) and B for the status mux (U46-3) will be 0, since only U44-9 and not U44-2, 4 or 6 is low. The other status mux controls are similarly decoded by U46, 1-3 and U46, 11-13 from the outputs of address decoder U44.

The Low Speed Reader Status Register is contained in flip-flops U33, U23, 5-9 and U23, 9-15. Errors in parity (U3-13), framing (U3-14) or overflow (U3-15) are ORed together to form the LSR error summary bit of

the TTY Reader/Keyboard Status register after being clocked by NACT* into the LSR Error FF, U23, 9-15. The UART's Data Available U3-19 serves as the LSR Ready bit, after being clocked by NACT* into the LSR Ready FF U23, 5-9. When the LSR address 167775 is decoded on U44-3, U45-13 will go high when strobed by BDTB* and reset DAV (Data Available). The Low Speed Reader Enable FF, U33, 1-5 and the Interrupt Enable FF, U33, 9-13 will be set on the clock edge if DBL2 and DBL3, respectively, are high (see bit assignment in previous section). The flip-flops are clocked by the Low Speed Reader decoded address 167774 ANDed with DWS* (Data Write Strobe Not). The Reader Enable FF is reset by the start bit of the incoming asynchronous data word via U4, 11-13.

TTY/EIA Punch/Printer Transmit & Transmit Status Register

DBL0* thru DLB7* are loaded into the Data Bit Inputs of the transmitter section of UART U3 with the pulse on U49-13 formed by the AND of the LSP address decode on U44-1 (167777) and DWS*. U9-3, 4, 10, 11 drives the TTY punch/printer. U14 provides EIA level signals (EIA interfacing is available only on I/O1601 Module). U34 and U29 provide the required clocking signals for the UART. The BAUD RATE strap is factory set at 110 baud, although 330 band (30 char/sec) can also be selected.

The LSP Status Register is contained in flip-flops U23, 9-12 and U50, 9-13. The LSP Interrupt Enable FF, U50, 9-13, will be set on the clock edge if DBL3 is high. The FF is clocked by the LSP decoded address (167776) ANDed with DWS*. The LSP Ready FF is set if TBMT (Transmit Buffer Empty) from the UART is a "1", signifying that the UART is ready to accept another character.

High Speed Reader & HSR Status Registers

The HSR data register is effectively contained in the high speed reader itself and enters the I/O1600/1601 module on J2-1 thru J2-8. Whenever HSR data is to be read, the Reader Enable (Drive) FF U51, 9-13 must first be set via the program. The program will set DBL2 high, setting the flip-flop upon receipt of the clock signal (U49-1) formed by the AND of the HSR address decode (167770) and DWS*. (The HSR Interrupt Enable FF U51, 1-5 is set in a similar manner). The Reader Enable (Drive) FF causes the reader to drive to the left one step.

When data is available from the HSR, HSRDAV on B21 goes high, firing one-shot U38, 1-4 and 14-15. (U38-4 going low will reset the Reader Enable (Drive) FF via U4, 4-6). The rising edge of the one-shot's pulse, which will occur 1.5 ms later, will clock a one into U37, 9-13 if the reader system is ready; i.e., the RUN/LOAD switch is in the RUN position and the power is on. The 1.5 ms delay serves to insure settling

of the reader drive before data is read and to prevent the computer from exceeding the maximum operating speed of the reader. NACT* will clock HSR Data Ready FF, U32, 9-12 and the HSR Error FF, U32, 9-14.

High Speed Punch & HSP Status Registers

DBL0* thru DBL7* are loaded into latches U17 and U18 via a clock on U45-4 formed by the AND of the HSP address 167773 and DWS*. The falling edge of the HSPSTB clock triggers one-shot U38, 6-11 to form a 10μs Punch Command pulse. HSP Error and HSP Ready are clocked by NACT* into FF's U32-2, 4, 9 and U32-5, 7, 9, respectively. HSP Error is an OR of Tape/Chad Error, Tape Low and System Ready Not.

Interrupt Logic

The priority of interrupts of the four devices serviced by this card is as follows: high speed reader, high speed punch, low speed reader and low speed punch. U43 ANDs the respective interrupt enable signal with the signal that can cause an interrupt (the OR of Ready and Error for the HSR, HSP and LSR and just Ready for the LSP). If interrupts to this card are not masked, priority encoder U48 is enabled via pin 4. If a high due to a HSR Ready Interrupt appears on U43-8, for example, it will cause U48-10 to go high. This interrupt signal will pass thru gates U53, 11-13, U52, 8-13 and U9, 8-9 causing an interrupt to the CPU. The interrupt acknowledge pulse will appear on H56 as IPRI* when all higher priority cards are done being serviced. The pulse will set interrupt FF U37, 1-6 via U19, 8-10 and Interrupt Address Multiplexer Drive FF U20, 9-13 via U24, 8-10. U37-5, now a 1, holds U48-10 to a 1 state via U47, 11-13, so that a lower priority interrupt will be locked out until the higher priority routine is finished. U20-9, now a 1, enables data multiplexers U13, U8, U7 and U12 via U25, 8-11 to enable an interrupt address to be presented to the data bus. IAB* can now pass thru U25, 3-6 (pin 4 is low since one of the interrupt FF's U37, 1-6, U15, 1-6, U15, 8-13 and U20, 1-6 is set; pin 5 is low since the card is not masked) and U10, 8-10 will enable the interrupt address 10 to the bus. After the address is presented to the bus, U20, 9-13 is cleared on the rising edge of IAB*, releasing the data mux. After servicing the interrupting device TCI* will clear the appropriate interrupt acknowledge FF.

If a lower device is interrupted by a higher device, TCI* will reset only the acknowledge FF that was just serviced. If, for example, U21-5 went high due to a higher order interrupt while U37, 1-6 was set, the TCI* that would clear the higher order interrupt would also maintain U37, 1-6 since the output of AND gate U21, 4-6 would be a 1. Likewise, U26, 1-10 and U41, 1-3 transfer the busy '1' signal down the chain, so that a lower order device two or three steps down the line also will not

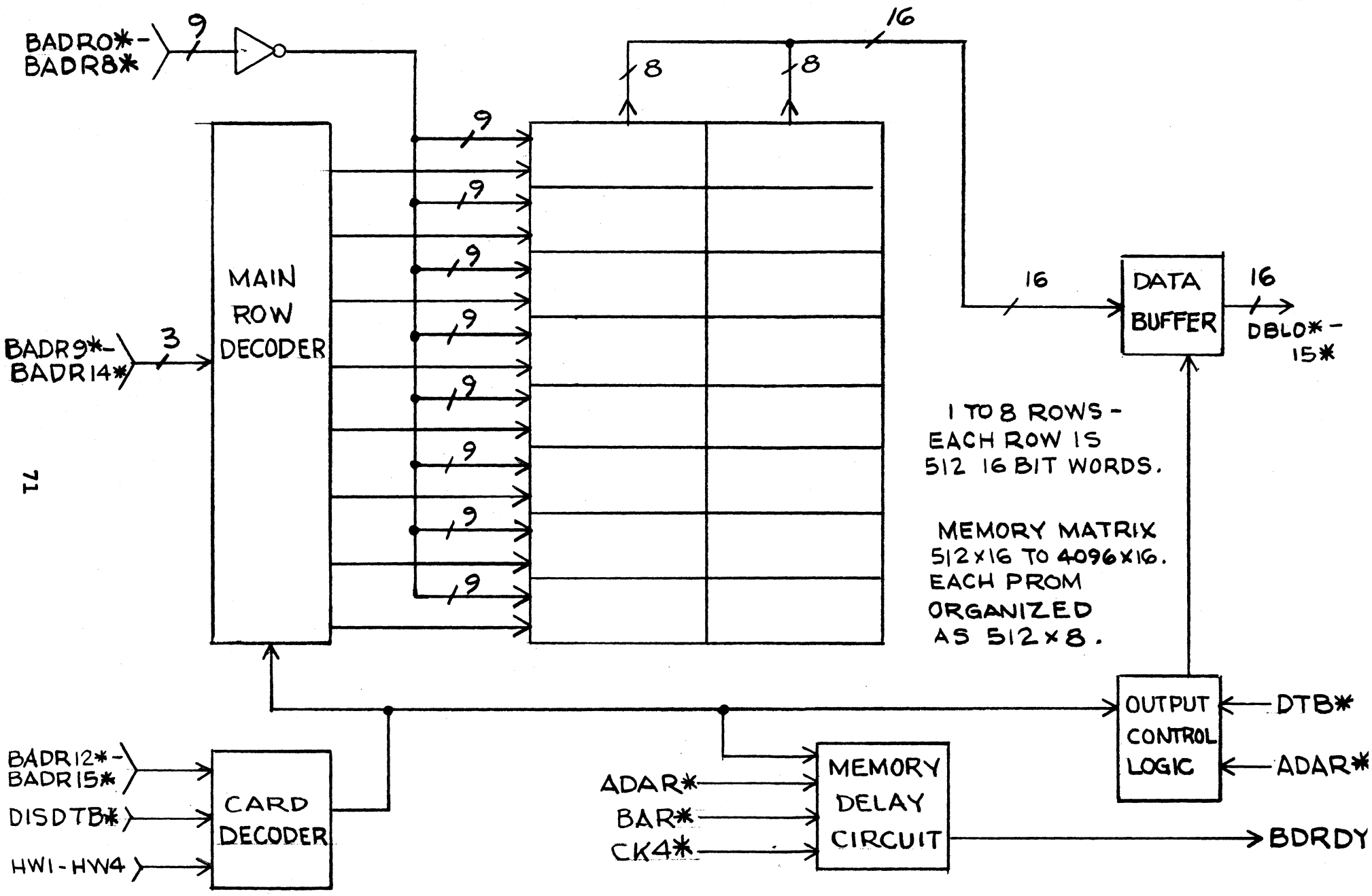


FIG. 27. 4K×16 PROM CARD

reset its acknowledge FFif it has already commenced its device routine.

1.6.6 4K PROM Card

The PM1600 PROM Memory Module has 16 sockets for 512 x 8 PROM chips, chips, such as National Semiconductor's MM5204. The total capacity of the card is 4096 words of 16 bits each and as many cards as required can be used to meet the particular program requirement. The PM1600 PROM Memory Module is useful during the initial product design phase before freezing the program for a production quantity of lower cost masked ROM's, such as General Instrument's RO-3-8316 (2K x 8) or RO-3-20480 (2K x 10).

The nine lower order address bits are applied to each PROM. Refer to Figure 27. BADR9*-BADR14* select a main row of 2 PROM's, while BADR12*-BADR15* select the particular PM1600 module. A memory delay circuit is used to insure that the CP1600 waits until stable data is available from the PROM's.

1.6.7 General Purpose Interfacing Card

Additional peripheral interfacing capability to the GIC1600 Systems can be provided by the GP1600 General Purpose I/O Module. It has 2 16-bit data input ports, 2 4-bit status data input ports, 2 16-bit latching data output ports and 2 4-bit latching status data output ports. All ports are accessible under software control. Wire/wrap sockets can be accommodated on the card for specific interface circuitry. The Data Bus lines, Address Bus lines, Control Bus lines and other signal lines, such as the interrupt and bus request handshaking signals, are brought from the edge connector to wire/wrap pins to facilitate prototype development.

The module select logic uses BADR3*-15* and four wires on the backplane (HW1-4). Refer to Figure 28. The three low order address bits BADR0*-2* are decoded such that one of the eight ports can be addressed. The input ports are fed to the data bus thru a multiplexer whose control lines are fed from the address decoder. The output ports are fed from the Data Bus to latches which are strobed by the address decoder.

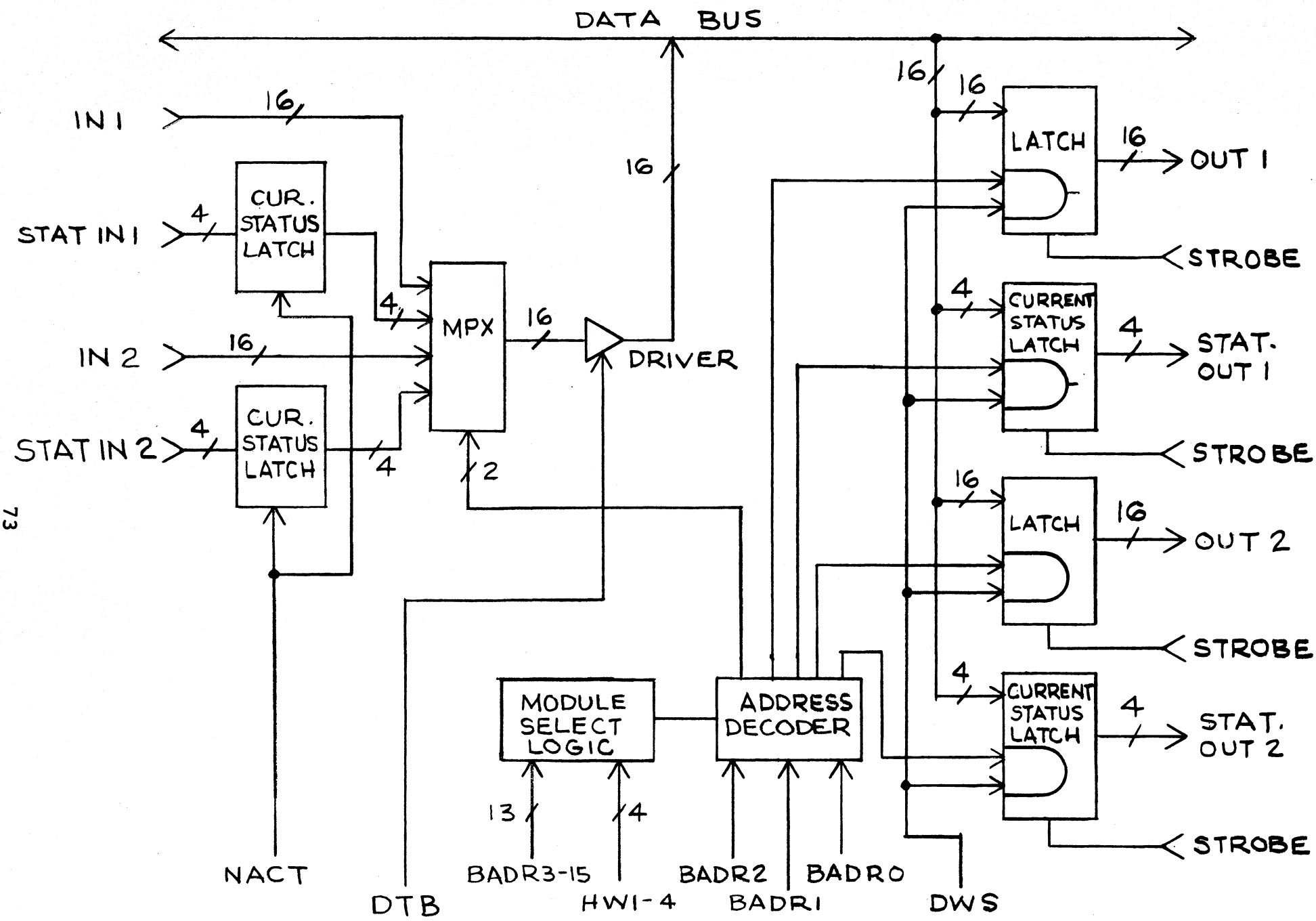


FIG. 28. GP1600 GENERAL PURPOSE I/O MODULE - BLOCK DIAGRAM.

CHAPTER 2

GIC1600 SUPPORT PROGRAMS

2.0 INTRODUCTION

The GIC1600 Series Microcomputer System is supported by a comprehensive firmware and software package which provides the user with a powerful yet low cost program development facility. The firmware, resident in Read Only Memory (ROM) is a small operating system which includes a monitor, an on-line program debugger, a relocating program loader, a memory dump routine and a collection of utility subroutines. The software includes a symbolic assembler, a relocating linking loader, a text editor, a system diagnostic program and an extensive library of subroutines.

2.1 RESIDENT FIRMWARE

The ROM resident operating system, an integral part of the GIC1600 microcomputer, provides the user with a convenient and easy to use facility for developing and debugging CP1600 programs.

2.1.1 MONITOR

The resident monitor supports control panel functions and recognizes a set of symbolic commands which are used to control the GIC1600 microcomputer. Front panel functions are provided by interrupt driven routines which enable the user to inspect and/or modify CP1600 registers & memory and input/output device registers by depressing appropriate switches. Control commands are processed by the resident monitor interactively via a teletype terminal.

Upon initial start-up via a system master clear, the monitor types an identifying message on the teletype and then types a currency symbol "$" on the next line. The "$" prompts the user to enter a control command on the teletype keyboard. Such commands are entered as lines terminated by a carriage return character. The monitor responds to a command with a line feed and then processes the command. If a command is unrecognized or contains an error, the monitor types a question mark "?", indicating the command has been rejected. Another command prompt "$" is then typed on the next line. When processing of a valid command is completed, another command prompt is typed on the next line. When additional data is required during command processing, the user is prompted by a colon ":". Data is entered as unsigned octal numbers of one to six digits, i.e., 0-177777, with leading zeros optional. Data is always displayed as unsigned six digit octal numbers. Characters in a line may be deleted one at a time by typing a "rubout" for each character to be deleted. The monitor responds by typing the characters deleted in order surrounded by "\". Entire lines may be deleted by typing "←"

instead of a carriage return. Monitor input/output is conducted using full duplex polled mode so that programs utilizing the teletype in interrupt driven mode can be debugged on-line.

2.1.2 ON-LINE DEBUGGING

The On-Line Debug Program (ODP), an integral component of the resident firmware, aids in rapid debugging of CP1600 programs by providing the following capabilities: Display and/or modify CPU registers, status word, memory and I/O device registers; search and initialize memory; program breakpoints and relocation origins and modifications of branch and jump instruction destinations.

Program execution may be suspended and control returned to ODP during program debugging in order to inspect registers, memory, etc. by setting breakpoints. Up to eight program breakpoints may be set simultaneously, each of which causes ODP to insert a SIN instruction at the specified address and save the original instruction. When a SIN is executed, an interrupt is generated which causes program control to be returned to ODP and the address of the executed SIN to be saved. Upon entry, ODP scans its table of active breakpoints to determine if entry is due to a breakpoint. If a breakpoint caused entry, "Bn@aaaaa" is typed to inform the user which program breakpoint was reached. ODP then removes all active breakpoints so that the program is restored to its original state, i.e., no SIN instructions before interaction with the user via the monitor is resumed. When the user enters a continue command indicating the program execution is to be resumed at the current breakpoint, ODP checks for an active breakpoint entry, rejecting the command if no breakpoint entry is active. ODP then analyzes the breakpointed instruction and CPU status to determine which instruction after the breakpointed instruction will be executed. The instruction is saved, a SIN inserted at this address and the breakpointed instruction restored. Program control is then transferred to the restored breakpointed instruction, causing the CPU to execute it and the immediately trap back to ODP because of the SIN at the next instruction address. The breakpointed instruction is then saved again, a SIN inserted, the continuation restored and program execution resumed. Thus, whenever the user is interacting with ODP, the program environment is undisturbed and the instructions at breakpoints may be changed before execution continues. There are no restrictions on breakpoint placement except that the breakpoint address must be at the first word of a multi-word instruction. If a SIN entry is unknown, i.e., not a breakpoint or the user has not defined a trap address, "T@aaaaa?" is displayed indicating that an unknown trap from address aaaaa caused entry. In this case all active breakpoints are removed from the program and interaction with the user via the monitor initiated. A subsequent continue command, however, will be rejected because a valid breakpoint did not cause entry.

Although there are no restrictions on breakpoint placement, continuation from

a breakpointed BEXT instruction or from most instructions which alter the contents of register 7, the program counter, is not permitted. Continuation is, however, provided for from breakpointed PULR PC (MVI R6, R&), JR Rn (MOVR Rn, R7) and MVII i, R7 instructions.

Program development and debugging efforts can be minimized by using modular programming techniques. CP1600 software enables the user to separate a program into logical sections or modules conveniently. These modules may be assembled separately and then linked together, relocated, and loaded prior to execution. ODP enables the user to reference addresses within a module relative to its assembly base address by setting an Origin to the relocated module base address. The user is thus relieved of the task of computing relocation addresses during debugging. An address may be expressed relative to Origin n (0-7) by entering the address value in the following format: On+a. For example, if Origin 3 is set to 2053, O3+16 specifies address 2071. Addressing may be specified in this fasion in any command which contains address specification(s).

When ODP is processing a command which results in extended output on the teletype, such as display addresses and search address, the user can cancel the activity by depressing the "CTRL" key while striking the "C" key. This causes the activity to be canceled and another command prompt "$" to be displayed.

In the following command descriptions, required items are underlined and "↵" represents a carriage return.

Command	Function
$;c.....c↵	Commentary.
$Aa↵	Inspect/modify contents of address a.
$Bn, a↵	Set breakpoint n (0-7) at address a.
$B, a↵	Set next available breakpoint at address a.
$Bn↵	Remove breakpoint n (0-7).
$B↵	Remove all breakpoints.
$Cn↵	Continue execution from current breakpoint (n times).
$DAl, h↵	Display contents of addresses l to h inclusive.
$DB↵	Display active breakpoints.
$DO↵	Display active origins.

Command	Function
$DR↓	Display contents of registers 0-7 and status word.
$DT↓	Display current SIN trap vector address.
$Ea↓	Execute at address a, if no address is specified the current value of R7 is used.
$IAl, h, v, m↓	Initialize the contents of addresses l to h inclusive to value v using mask m. If no mask is specified, 177777 is used.
$MBa, b↓	Modify the branch instruction at address a for destination address b.
$MJa,b↓	Modify the jump instruction at address a for destination address b.
$On, a↓	Set origin n (0-7) to address a.
$O, a↓	Set next available origin to address a.
$On↓	Remove origin n (0-7).
$O↓	Remove all origins.
$Rn↓	Inspect/modify contents of register n (0-7).
$Sa↓	Single step instruction at address a, if no address i specified the current value to R7 is used.
$SAl, h, v, m↓	Search address l to h inclusive for value v using mask m. The mask is used to extract the corresponding bits from the contents of each address before the comparison with the value takes place. If no mask is specified, 177777 is used.
$SW↓	Inspect/modify status word.
$Ta↓	Trap to address a, i.e., set SIN trap rector address.
$T↓	Deactivate SIN trap vector.

2.1.3 RELOCATING LOADER

The resident relocating program loader is used to read relocatable and absolute paper tape load modules into memory. The following commands are used to initiate loader activity.

Command	Function
$Lda↵	Load relocatable or absolute binary paper tape from device d (L = low speed reader, i.e., TTY; H=high speed reader). If address a is specified and the tape is relocatable, loading begins at address a. If address a is not specified, loading begins at the origin address specified on the tape.
$MLa, b↵	Set lower and upper load memory limits to addresses a and b respectively. If a and b are not specified, the current memory limits are displayed.

The memory limit command is used to protect program code which may be resident in memory from accidental corruption during program loading. Upon successful completion of a program load, a summary is typed which indicates the initial address and final address loaded and if an entry was specified in the load module, its address.

During loading several error conditions may arise which cause the load to be aborted. Such errors are reported to the user by one of the following messages which are typed on the teletype.

L@aaaaaa - memory limit violation, i.e., a word is to be loaded into the indicated memory location which is outside the currently defined memory limits.

M@aaaaaa - memory failure, i.e., a word has been stored into the indicated memory location and cannot be correctly read back. Possibly due to a non existant memory cell, a ROM memory cell or a defective memory cell.

R - user has specified relocation for an absolute load module.

T - user is attempting to load an object module or non binary tape, i.e., source tape.

C - an error has been detected while reading the last tape record. As each record is read, a check sum is computed which is compared to the tape record check sum.

E - end of medium or device error has been detected on the high speed paper tape reader.

2.1.4 MEMORY DUMP

The resident memory dump routine is used to punch a memory image tape of a specific area of memory. The tape is formatted as an absolute load module which can be loaded back into the same memory area using the resident loader. The following command is used to initiate punch activity:

Command	Function
$Pd, l, h↵	Punch the contents of addresses l to h inclusive on paper tape in absolute format using device d (L = low speed punch, i.e., TTY; H - high speed punch).

The paper tape dump may be cancelled by typing "CTRL/C" on the teletype keyboard. If the high speed paper tape punch runs out of tape during the dump, "E" is typed to so indicate and the dump is aborted.

2.1.5 TAPE DUPLICATION

Paper tapes of any format may be duplicated using the high speed paper tape reader and punch by entering the following command:

$K↵	Copy tape on high speed tape reader and punch. Copying continues until end of tape is sensed or until the user types "CTRL/C" on the teletype keyboard.

If an error or end of medium on the reader or out of tape on the punch is detected an "E" is typed and the tape copy is terminated.

2.1.6 UTILITY ROUTINES

The GIC1600 resident firmware contains several general purpose utility subroutines which may be utilized by user programs via subroutine calls (JSR R5 instructions). The following list describes each subroutine function, its entry address and its general characteristics.

BINOCT - Binary to six digit ASCII octal conversion.
Entry address: 175000
Inputs: R0 = binary quantity to be converted.
R1 = buffer base address.
Output: Six octal digits stored in the low byte of each buffer word. The most significant digit is stored in the first (base address) buffer word.
Registers 0-4 unchanged.

OCTBIN - Six digit ASCII octal to binary conversion.
Entry address: 175002
Inputs: One or more ASCII octal digits stored contiguously in the low byte of each buffer word, most significant digit stored in base address of buffer.
R1 = buffer base address.
Outputs: R0 = binary quantity.
R1 = address of character at which conversion terminated. Leading spaces are ignored and conversion is terminated on the first non octal digit (0-7) or when six digits have been converted.
Registers 2-4 unchanged.

TYPSTR - Type character string on teletype.
Entry address: 175004
Inputs: R4=character string base address.
ASCII characters to be typed are stored in the low byte of each word with a zero byte terminating the string.
Outputs: None
Registers 0-3 unchanged.

TYPR2 - Type ASCII character on teletype
Entry address: 175006
Input: R2 = character
Outputs: None
Registers 0-4 unchanged.

TYPOCT - Convert binary quantity to six digit octal and type on teletype.
Entry address: 175010
Input: R0 = binary quantity to be converted.
Outputs: None
Registers 0-4 unchanged.

TYCRLF - Type carriage return and line feed on teletype.
Entry address: 175012
Inputs: None
Outputs: None
Registers 0-4 unchanged.

SELDEV - Select paper tape device.
Entry address: 175014
Input: R0 = 0 - low speed device (TTY)
R0 = 1 - high speed device (HSP)
Outputs: None
Registers 0-4 unchanged.

PUNLDR - Punch 100 frame paper tape leader on selected device.
Entry address: 175016
Inputs: None
Outputs: None
Registers 0-4 unchanged.

PUNWRD - Punch word on selected device.
Entry address: 175020
Input: R2 = data word
Outputs: R0 - check sum accumulation.
Registers 1-4 unchanged.
Note that each word results in two data frames, low byte, high byte.

PUNR2 - Punch byte on selected device
Entry address: 175022
Input: R2 = data byte (low byte)
Output: None
Registers 0-4 unchanged.

RDFRM - Read byte from selected device.
Entry address: 175024
Inputs: None
Outputs: R1 = data byte (low byte)
Registers 0, 2-5 unchanged.

INURSP - Type ":" on teletype and then input character
tring from teletype.
Entry address: 175026
Inputs: None
Outputs: R1 = buffer base address
R2 = first character
Input characters in low byte of each buffer word.
Registers 0, 3-4 unchanged.
Note that input is via the resident monitor which provides character

echo, character delete (rubout) and line delete (←) facilities. The character string is terminated by a carriage return (15 returned in buffer) to which the monitor responds with a line feed.

CHKUSR - Sample teletype input register for CTRL/C character.
Entry address: 175030
Inputs: None
Outputs: Status word C bit set if CTRL/C detected, cleared otherwise.
Registers 0-4 unchanged.

INSTR - Input character string from teletype.
Entry address: 175032
Inputs: No register inputs. Calling sequence as follows:

```
JSR R5, INSTR
BYTE    buffer base address
WORD    number of buffer words
WORD    'c' (prompt character)
```

Outputs: Input characters in low byte of each buffer word.
Registers 0-4 unchanged.
Note that input is via the resident monitor which provides character echo, character delete (rubout) and line delete (←) facilities. The character string is terminated by a carriage return (15 returned in buffer) to which the monitor responds with a line feed.

2.2 ON-LINE SOFTWARE

The On-Line software supplied with the GIC1600 Microcomputer System provides the user with a sophisticated program preparation facility which is especially oriented toward developing CP1600 programs using modular techniques. The Symbolic Assembler (S16AL) allows source program tapes to be segmented in order to simplify editing, recognizes global symbols and external symbol references and generates relocatable object modules. The Relocating Linking Loader (S16RLL) relocates, links and loads object modules produced by either S16AL or the Series 1600 Cross Assembler (S16XAL). The text editor (S16TXE) provides source tape editing facilities at both line and character levels in order to simplify source program tape preparation and correction. The system diagnostic program (S16DGS) is used to test and establish confidence in the GIC1600 Microcomputer System.

2.2.1 S16AL SYMBOLIC ASSEMBLER

The Series 1600 Symbolic Assembler (S16AL) is a program preparation aid which supports General Instrument's family of 16-bit microprocessors. It translates ASCII coded alphanumeric source programs into several different types of binary machine coded paper tape object modules.

2.2.2 FEATURES

- Symbolic representation of all CP1600 instructions
- Literal representations in Octal, Decimal, Hexadecimal, Binary and Character notation
- Arithmetic evaluation of operand expressions
- Assembly directives for
  - Controlling memory allocation
  - Defining character strings
  - Specifying input/output options
  - Establishing conditional assemblies
  - Declaring global and external symbols
- Assembly in three forms
  - Absolute
  - Relocatable
  - Relocatable/Linkable
- Program listings
- Error diagnosis

2.2.3 OPERATION

The S16AL Symbolic Assembler converts symbolic source programs into binary machine code in a two pass process. During the first pass through the source program, all user specified symbols are placed in a symbol table containing the symbol, its value, and several other attributes. During the second pass through the source program, symbolic instruction mnemonics are translated, symbol references resolved, errors diagnosed, binary machine code generated, and a program listing produced.

The binary code produced by the S16AL assembler can be formatted in several ways depending upon the use of assembly directives. If the source program specified an absolute assembly, the binary code is formatted as an absolute load module which can be loaded but not relocated by the resident loader and subsequently executed. If the source program contains global symbol definitions and/or external symbol references, the binary code is formatted as a relocatable object module. Relocatable object modules must be linked together by the relocating linking loader (S16RLL) prior to execution. If the source program does not contain globals or externals, the binary code is formatted as a relocatable load module which can be relocated and loaded by the resident loader and then executed.

2.2.4 SOURCE PROGRAM FORMAT

A S16AL source program is composed of a sequence of statements with each statement contained on a single line terminated by a carriage return character. A statement may contain up to four fields which are identified by their appearance from left to right. The general format of a S16AL statement is: Label, Operator, Operand, Comment. The label and comment are optional, while the operator is always required. The presence and nature of the operand depends upon individual operators. It is recommended that statements be limited to approximately 50 characters so that assembled programs can be printed on teletype or other terminals.

Label

A label is a user defined character string, used to symbolically reference a specific location within a program. If a statement contains a label, the label must begin in the first position of the statement. Labels may contain up to six characters, the first of which must be a letter (A-Z), a currency symbol ($), a question mark (?), or an ampersand (&). The remaining five optional characters may be any valid ASCII character except a blank space, since a space serves as the label terminator. Labels containing more than six characters cause a diagnostic to be issued and are truncated after the sixth character. Labels must be unique in the first six characters, i.e., a specific character string cannot be used in the label field of a statement more than once in a program. Multiple use of a label causes a diagnostic to be issued and the subsequent definitions of the label to be ignored.

An operator follows the label field in a statement. A statement operator contains up to four characters and may be an instruction mnemonic or an assembly directive. Instruction mnemonics are symbolic character strings which represent the various Series 1600 microprocessor instructions. Assembly directives are symbolic character strings used to represent certain functions or actions performed by the assembler during the assembly process. If a statement does not contain a label, the operator must be preceded by at least one blank space. If the operator is the last field in a statement, it is followed by a carriage return, otherwise it is followed by a blank space.

Operand

An operand follows the statement operator separated by at least one blank space. The operand represents an item or items to be operated upon by the statement operator. Operands may be symbols, literals or expressions. When multiple operands are used, they are separated by commas. If an operand is the last field in a statement it is followed by a carriage return, otherwise it is followed by the comment field.

Comment

The comment field is optional in all statements and must be preceded by a semicolon (;). The contents of the comment field are printed on the program listing but have no effect on the assembled program. Entire lines may serve as comments if the first non blank character is a semicolon. Blank lines are printed on the program listing but otherwise ignored so that statements may be separated in order to enhance program readability. The liberal use of commentary is strongly recommended so that the function and operation of programs is evident from the program listing.

2.2.5 SYMBOLS

A symbol is a character string which appears in an operand and is used to represent the value assigned to the symbol by the assembler. A symbol is given a value either by direct assignment via an assembly directive or by appearing in the label field of a statement. Instruction labels are given the value of the assembly location counter assigned to the associated instruction. The assembler recognizes the exclamation mark (!) as a special symbol for the current value of the program counter.

2.2.6 LITERALS

Literals are character strings which serve as sources of data, i.e., cannot be changed and are interpreted by the assembler as constants. The assembler accepts literals expressed as octal, decimal, hexadecimal, binary and character. Literals may be preceded by a plus or minus to signify sign; plus is assumed unless a minus is present.

Octal

soooooo - s = optional + or –, + assumed
o = 0-7
oooooo = 0 to 177777

Decimal

s.ddddd - s = optional + or –, + assumed
. = leading character
d = 0-9
ddddd = –32768 to 32767

Hexadecimal

X'hhhh' - s = optional + or –, + assumed
X' = leading characters, ' = trailing character
h = 0 9, A-F
hhhh = 0 to FFFF

Binary

sB'bbbbbbbbbbbbbbbb' - s = optional + or –, + assumed
b' = leading characters, ' = trailing character
b = 0, 1
bbbbbbbbbbbbbbbb = 0 to 1111111111111111

Character

s"cc" or 'c' - s = optional + or –, + assumed
"or' = delimiter
c = any ASCII character

One or two characters may be packed into each 16-bit word. If one character is specified ("c" or 'c') it is placed in the low order byte of the word with zeros in the high order byte. If two characters are specified ("ab" or 'ab') the first (a) is placed in the low order byte and the second (b) is placed in the high order byte.

2.2.7 EXPRESSIONS

Arithmetic operators (+ and –) may be used to form operand expressions. An element of an expression may be: a user defined symbol, the current assembly location counter symbol (!) or a literal. Expressions may contain up to six elements separated by either + or – operators. The total expression may be terminated by a comma, a carriage return or a semicolon. Expressions are always evaluated from left to right with no parenthetical groupings allowed.

2.2.8 ASSEMBLY DIRECTIVES

Assembly directives are used to control the assembly process and in some cases cause data to be generated. In the following assembly directive descriptions optional elements are enclosed in [ ]. Comments may be used with all assembly directives.

LABEL	OP	OPRND	ACTION
	PAGE		Advance program listing to the top of the next page. Sixty lines are normally printed on each page.
	HEAD	'ccc...c'	Use the character string specified as the operand as a page heading. The first character ('or") is used as the string delimiter.
	REL	[name]	Assemble a relocatable module and use the six character name as the object module identifier. If no name is specified, an unnamed relocatable object module is generated. The module name is used by the Series 1600 Relocating Linking Loader (S16RLL) to identify object modules on its load map. The REL directive must be encountered by the assembler before any data generating operators are processed. If this is not the case, a questionable use diagnostic is issued and an unnamed relocatable object module generated. If no REL or ABS directive is specified, an unnamed relocatable object module is generated; i.e., REL is the default assembly mode.
	ABS		Assemble an absolute load module, which cannot be relocated or linked when loaded. The ABS directive must be encountered by the assembler before any data generating operators are processed. If this is not the case, a questionable use diagnostic is issued and an unnamed relocatable object module is generated.
	ENTR		Define the program entry point; i.e., the location at which execution is to begin after the program is loaded.

LABEL	OP	OPRND	ACTION
	GLOB	Sym[, Sym. . . , Sym]	Declare the symbol(s) as global. Global symbols must be defined as labels in the current program unit but can be referenced from other program units.
	EXT	Sym[, Sym. . . , Sym]	Declare the symbol(s) as external. External symbols reference global symbols in other program units. Both external and global symbol references are resolved by the Series 1600 Relocating Linking Loader (S16RLL).
	ORG	expr	Set the assembly location counter to the value of expr. Program Assembly starts at zero by default.
SYMBOL	EQU	operand	Assign the value of the operand V to the symbol. The operand may be a symbol, a literal or the assembly location counter symbol (!). If ! is specified it may be followed by + or − and a literal. Note that only one level of forward symbol reference is allowed.
[LABEL]	RES	expr	Reserve a block of storage whose length is specified by expr. The contents of individual storage locations is undefined. If a label is specified, it is assigned a value equal to the address of the first word in the block
[LABEL]	ZERO	expr	Zero a block of storage whose length is specified by expr. If a label is specified, it is assigned a value equal to the address of the first word in the block.
	BITS	expr	Use the value of expr to specify the number of bits in a memory word, i.e., word size. The word size is used to check generated data for magnitude exceeding word size during assembly. The default is 16 bits.
	MEML	expr1[, expr2]	Specify lower and upper memory address limits as the values of expr1 and expr2. If only expr1 is specified, its value will be used as the upper memory address limit and

LABEL	OP	OPRND	ACTION
			the lower limit will be set to zero. These limits are used by the assembler to check the validity of addresses assigned to generate code. The defaults are 0 and 177777.
[LABEL]	WORD	expr[, expr, . . . , expr]	Generate a data word for each operand expression. The contents of each word is set equal to the value of the respective expr. If a label is specified it is assigned a value equal to the address of the first word generated.
[LABEL]	BYTE	expr[, expr, . . . , expr]	Generate two data bytes for each operand expressions. The operation is the same as with WORD but 8 bit data is generated for use with double byte addressing in 10-bit memory.
[LABEL]	TEXT	p, 'cc. . . . cc'	Generate a word or words of data which contain the seven bit ASCII code for each character between the delimiters 'or". If p=1, one character is placed in the low byte with zeros in the high byte of each word. If p=2, two characters are packed in each word, low byte to high byte. Incompletely filled words contain a space character in the high byte. If p is not specified, two characters are packed per word. If a label is specified, it is assigned a value equal to the address of the first word generated.
	END		End of the program, the assembly is terminated on the previous statement.
	EOT		End of tape indicator, used to separate a source program into several tape segments.
	NLST		Disable the program listing. The assembly proceeds normally but with the listing suppressed. This directive is used, for example, to avoid printing a length ZERO block.
	LST		Enable the program listing. This directive is used to cause a listing to again be produced after a NLST directive.
	IFEQ	expr	Start conditional assembly. The statements

LABEL	OP	OPRND	ACTION
			that follow will be assembled if expr is equal to zero. If expr is not equal to zero, the statements will be listed but not assembled. Conditional assemblies are useful when a program has statements which are to be assembled only under certain conditions. For example, statements which are to be assembled only during debugging of the program.
	IFNE	expr	Start conditional assembly. The following statements will be assembled if expr is not equal to zero, the following statements will be listed but not assembled.
	ENDC		End conditional assembly, i.e., resume normal assembly.

2.2.9 PROGRAM LISTING

The S16AL Symbolic Assembler produces a listing of the assembled program containing the following fields: line number; address; contents; the statement label, operator, operand and comments. The address and contents fields each contain six octal digits and the operator, operand and comments are tabulated to enhance program readability. If the assembled word is subject to modification when the program is loaded at an address different than that of assembly; i.e., relocated, the contents are followed by the letter "R". If the assembled word references an external symbol, the contents are followed by the letter "X".

Each page of listing contains sixty lines and begins with a one or two line heading. The first heading line contains the module name, the version of the assembler in use and the page number. If the user has specified a heading via the HEAD directive, it follows on the next line. The program listing follows, separated from the page heading by a blank line.

At the end of the program listing, all user defined symbols and the number of diagnostics issued are summarized. The symbol summary lists each symbol in alphabetical order, its octal value, and its attributes. The following codes are used to indicate symbol attributes:

U - symbol is undefined
A - symbol is absolute
R - symbol is relocatable
X - symbol is external

IN - symbol is an instruction label
EQ - symbol is defined by an EQU statement
RS - symbol is a RES or ZERO statement label
DT - symbol is a WORD BYTE, or TEXT statement label

G - symbol is global

E - symbol is entry point

DD - symbol is doubly defined

UR - symbol is unreferenced

2.2.10 DIAGNOSTICS

The S16AL Assembler performs extensive diagnostic checking of each statement during program assembly and prints diagnostic codes on the program listing immediately preceding any statement in error. The diagnostic code characters and their meanings follow.

Diagnostic	Meaning
A	Address outside memory limits
B	Double byte data sequence error
D	Doubly defined symbol
E	Expression longer than 6 elements
L	Label missing or illegal
M	Doubly defined symbol referenced
N	Numeric value of literal illegal
O	Operator unrecognized
P	Pass 2 symbol value different than in pass 1
Q	Questionable syntax
R	Register designator illegal
S	Syntax illegal
T	Truncation of statement field
U	Undefined symbol referenced
V	Value of operand illegal
W	Word size exceeded
X	Expression contains more than one external symbol
?	Questionable use of directive
↑	Previous statement incomplete
+	Symbol table overflow

2.2.11 USING S16AL

The Series 1600 On-Line Assembler requires a GIC1600 Microcomputer system with 6144 words of RAM and a teletype terminal for execution. A high speed paper tape reader and punch may be used if available. Upon initial start-up, S16AL identifies the version in use, indicates assembly pass 1 and requests source input device identification by printing "SRC?:" The user responds with "L" (low speed tape reader, i.e., teletype) or "H" (high speed tape reader). Note that all user responses are terminated by a carriage return. If only a carriage return is entered in response to "SRC?:", control is returned to the resident system monitor. Next, the assembler requests the user to select a pass 1 listing option by printing "LST?:". The user may select no listing, diagnositcs only listed or a full listing by entering "N", "D" or "Y". A pass 1 program listing is complete except for forward symbol references and undefined symbol diagnostics. During initial stages of program development it may be suitable to work with a pass 1 listing until all errors are corrected. If a diagnostic listing is selected, only diagnostics and the corresponding statement are listed. When the listing option is entered, pass 1 of the assembly process begins. If a source tape ends with an "EOT" directive, the assembler prints "END?:" to which the user responds with "N" when the next tape segment is mounted and ready in the tape reader. If the "EOT" is to be treated as an "END", the user responds with "Y". At the end of pass 1, "PASS 2?:" is printed allowing the user to proceed with assembly pass 2 ("Y" entered) or return to the resident system monitor ("N" entered).

At the beginning of assembly pass 2, "LST?:" is again printed allowing the user to select no listing ("N"), a diagnostic listing ("D") or a full program listing ("Y"). Next, the device upon which the object module is to be punched is requested by "OBJ?:" being printed. If the high speed tape punch is to be used, "H" is entered; if the low speed tape punch is to be used, "L" is entered. If no object module is to be generated during pass 2, "N" is entered. If a pass 2 listing and object output on the low speed punch are selected, "DEV CNFLCT" is printed and the pass 2 options again requested because the listing and object code cannot be mixed on the teletype during the same assembly pass. If object output is to be pun ched on the low speed reader, the user must manually enable the teletype punch before entering the "L" response. Since pass 2 starts when the object option is entered, the source tape must be repositioned on the appropriate reader before the option is entered. If the source program tape is segmeneted, the segments must be mounted in the same order as in pass 1. Pass 2 may be rerun by entering "Y" in response to the message "PASS 2 AGAIN?:" which is printed at the end of pass 2. If "N" is entered, pass 1 is reinitiated.

A program assembly can be aborted by depressing the teletype "CTRL" key while simultaneously striking the "C" key. This causes the assembly to be cancelled (voiding any binary output) and control to be returned to the resident monitor.

2.3 TEXT EDITOR

The Series 1600 Text Editor (S16TXE) is used to prepare, correct and modify source programs or other text using the GIC1600 Microcomputer System. S16TXE provides both line and character editing facilities with deletion recall via simple single character commands.

2.3.1 OPERATION

S16TXE communicates interactively with the user via a teletype terminal, accepting single character commands when a command prompt "*:" is printed. Text to be edited is read into memory from paper tape or entered directly via the teletype keyboard. If more paper tape text is to be edited than available memory can hold, the text is automatically segmented when space remains for approximately 256 additional characters. When all editing operations are complete on the text currently in memory, it is written out on paper tape and additional text segments are read in for subsequent editing. S16TXE accepts paper tapes punched in eight bit ASCII code as input, ignoring nulls, rubouts and non eight bit characters. Interactive user dialogue is conducted using seven bit ASCII code to prevent possible corruption of text tapes when using a teletype for output. Each line of text must be terminated by a carriage return and should not exceed 72 characters in length. The user can specify input text segments explicitly by punching form feed characters (CTRL/L) at appropriate points in the text tape. Form feeds, however, are not read into memory and if output text segmentation is desired the user must explicitly inject form feeds using the appropriate command. Text editing is performed at either the line or character level by moving a character pointer through the text. Lines of text may be exchanged, inserted or deleted while character strings within a line may be changed or deleted. Text which has been deleted remain in memory in an inactive state until the text is written out or purged. Consequently, deleted text may be recalled if required.

2.3.2 COMMANDS

S16TXE recognizes a set of single character commands which may be grouped according to the following functions:

- Input/output of text
- Searching for character strings
- Moving and marking the character pointer
- Inserting, exchanging and deleting lines
- Changing and deleting character strings

The user enters one of the following command characters and any additional data required followed by a carriage return in response to the command prompt "*:". If a command requires additional text, S16TXE enters the text mode, prompting the user with a colon ":" on the next line. The user then enters the appropriate text terminated by a carriage return. When processing of a command is completed, another command prompt is printed. If a command is unrecognized or contains an error, a question mark "?" is printed followed by

another command prompt. All user input via the teletype is under resident monitor control which provides character deletion and line deletion facilities using "Rubout" and "←".

In the following command descriptions "*" and ":" are printed by S16TXE, required user entries are underlined and a carriage return is represented by "↲".

*:An↲	Advance the character pointer +or - n lines. A0 moves the character pointer to the beginning of the current line, A moves the character pointer ahead to the next line.
*:B↲	Move the character pointer back to the beginning of text, i.e., the first line.
*:Cn↲ :c...c↲	Change n characters in a line beginning with the current character to the specified character string. C changes the current character.
*:Dn↲	Delete n characters in a line beginning with the current character. Note that deleted characters are deactivated and may be yanked (recalled) until the text is written out or purged.
*:E↲	End, i.e., write out current text followed by a form feed, i.e., text segment indicator and approximately 100 frames of blank tape.
*:F↲	Write out a form feed, i.e., text segment indicator followed by approximately 100 frames of blank tape.
*:Gn↲	Get the nth occurrence of the specified character string from the current character position. G gets the first occurrence. When the specified character string is found, the character pointer is positioned at the next character. If the specified character string is not found, "NONE!" is printed and the character pointer is positioned at the end of text.
*:I↲ :c...c↲ :c....c↲	Insert lines prior to the current line. One or more lines are inserted as entered at the text level until a null line, i.e., only a carriage return is entered.
*:Jn↲	Jump the character pointer + or n character positions. J0 moves the character pointer to the beginning of the current line, J moves the character pointer ahead to the next character.
*:Kn↲	Kill (delete) n lines beginning with the current line. K kills the current line. Note that the killed lines are deactivated and may be yanked (recalled) until the text is written out or purged.

*:Ln↓ List n lines beginning with the current line. L. lists the current line.

*:Ma↓ Mark the current character position. M. sets mark, M0 clears mark.

*:N↓ Read next text segment. The text currently in memory is written out and the next text segment is read into memory.

*:O↓ Open output tape, i.e., punch blank leader.

*:P↓ Purge the text currently in memory. All deactive text resulting from delete or kill operations is permanently removed and associated storage released.

*:Qn↓ Quote, i.e., repeat or copy text from one position to another. Text is copied from the marked character position to the current character position. Qn copies n, Q copies one line.

*: R↓ Read text into memory until a n text segment indicator, i.e., form feed is detected or until space for approximately 256 characters remains. The input text is read into memory starting at the current position of the character pointer and consequently may be appended to text already in memory.

*:S↓ Stop, i.e., return to the resident monitor

*:Tn↓ Tab set, i.e., specify tabulation positions. When tab characters (CTRL/I) are detected either in input text or text entered at the text level sufficient space characters are injected to advance to the next tabulation position. Tn, n=2-71 sets tab radix to n. T0 resets the tab radix to the default, 8. T prints the current tab radix.

*:V↓ Verify the current character position by listing the current line from the current character position to the end. VM↓ verifies the currently marked character position.

*:Wa↓ Write out the text currently in memory. W writes all text, W-1 writes from the marked position up to the character prior to the current character. W0 writes the current line.

*:X↓
:cc....c↓
:cc....c↓

Exchange the current line, i.e., the line currently containing the character pointer with the specified line of text. Additional lines may be then inserted following the line exchanged until a null line is entered, i.e., only a carriage return.

*:Ya↓	Yank inactive text, i.e., recall deleted characters and/or killed lines. Y0 yanks the current line, Y-1 yanks from the character pointer to the marked character, y yanks the all text in memory.
*:ZE↓	Zero buffer, i.e., reset pointers.
*:#↓	Print the number of the current line, i.e., the line currently containing the character pointer.
*:=↓	Print the number of character positions currently available for use in memory.
*:7↓	Accept text punched in 7-bit ASCII code as input.
*:8↓	Accept text punched in 8-bit ASCII code as input.

2.3.3 USING S16TXE

S16TXE requires a GIC1600 Microcomputer with 4096 words of RAM and a teletype terminal for operation. This configuration provides buffer space for approximately 2200 text characters while additional available memory increases the buffer size one character per word. A high speed paper tape reader and punch may be used for input and output of text tape if available.

Upon initial startup, S16TXE identifies the version in use and them prints the text buffer base address. Next, "BUFFER END ADDR?:" is printed, requesting the user to enter the octal address to be used as the text buffer upper limit. The address entered, which must be greater than the buffer starting address, allows the user to protect data or program code that may be in memory. If no address is entered, i.e., only a carriage return, all of available memory is used as the text buffer. After the buffer limit has been determined, a command prompt "*:" is printed.

When an input command is entered, the user is requested to select the device by the message "DEV?:". The user must then respond with either "L" (low speed TTY paper tape reader) or "H" (high speed paper tape reader). If the low speed reader is selected, "RDR ON!" is printed to remind the user to manually enable the TTY tape reader. Likewise, when an output command is entered, "DEV?:" is printed to which the user must respond with "L" or "H". If the low speed punch is selected, "PNCH ON!" is printed to remind the user to manually enable the TTY tape punch. When the punch has been enabled the user must enter a carriage return which indicates to S16TXE that the punch is ready. When input or output via either of the low speed TTY devices is complete, indicated by a command prompt, the user must manually disable the device. Accidental corruption of output text tapes is prevented if the punch is not turned off since seven bit ASCII code is for user dialogue and eight bit ASCII code for all text information.

If the user attempts to insert additional text when the text buffer is full, "BUFFER FULL!" is printed and the command is rejected. Note that a purge releases memory space occupied by text placed in an inactive state by delete or kill operations.

S16TXE can be restarted at any time with no loss of text by executing at its entry address +2. Initial startup, however, should always begin at the entry address.

I/O operations can be cancelled by depressing the teletype "CTRL" key while simultaneously striking the "C" key. S16TXE then returns to the command level, i.e., a command prompt "*" is printed.

2.4 S16RLL RELOCATING LINKING LOADER

The Series 1600 Relocating Linking Loader (S16RLL) is used to load and link together one or more object modules produced by the Series 1600 Cross or On-Line Assemblers (S16XAL) or (S16AL). S16RLL enables the user of a GIC1600 Prototyping Microcomputer System to utilize efficient modular program development techniques by providing the following facilities:

- Relocates and loads program object modules
- Resolves global/external symbol linkages.
- Produces a load map containing module names, origins, sizes and global symbol allocations.
- Detects and reports: unsatisfied external symbols; multiple global symbols and program entries.

2.4.1 OPERATION

Upon initial start-up, S16RLL identifies the version in use and makes the following requests for information from the user (↵ represents a carriage return).

ADR?:a↵ - specify initial or base load address (a = one to six digit octal address).

MAP?:o↵ - load map desired? (o = Y or N).

DEV?:d↵ - specify load device (d = H, L - high or low speed tape reader)

Loading begins after the user responds to the "DEV?:" request. If a load map had been requested, the object module origin address, its name, any global symbols and the module size are printed. When the end of the object module tape is reached, another tape is requested by the "DEV?:" message. If another object module is to be loaded it is mounted and a "Y " response is typed. The load process repeats for each module until the user indicates that the last module has been loaded by typing only a carriage return in response to the "DEV?:" message. S16RLL then resolves any global linkages that may be outstanding, reporting as unsatisfied externals, references to global symbols which are undefined. Finally, a load summary is printed, indicating the initial and final load addresses and the entry address if an entry was specified in one of the loaded object modules. Control is then returned to the resident monitor.

During the load/link process a table of global symbols and temporarily unresolved external references is constructed. The table extends downward in address from the initial address occupied by S16RLL toward the program being loaded. Table size may be minimized by loading modules with global symbol definitions first so that loading of subsequent modules do not result in temporarily unresolved externals. Since the table extends downward, S16RLL should be loaded as high as possible in the available storage area to provide maximum free storage for the program being loaded and the linkage table. S16RLL requires approximately 1180 (2234 octal) word of storage for execution with the global/external symbol table varying in size depending upon the number of global/unresolved externals (four words per entry).

2.4.2 ERROR MESSAGES

S16RLL detects several error conditions during the loading process which are reported to the user by the following messages:

REL?	-	The current object module is not relocatable, load aborted.
MLT GLOB:s	-	Multiple definition of the indicated global symbol has been detected, the first definition is retained, load continues.
MLT ENTR@a	-	Multiple program entry point of the indicated address has been detected, the first entry point is retained, load continues.
UNSAT EXTS	-	The following references to undefined global symbols at the indicated address have been detected, zero value is supplied, load continues.
OVERFLO	-	The global symbol/unsatisfied external reference table has overlapped the program being loaded, load aborted.
E?	-	End of tape (medium) or reader error, load aborted.
C?	-	Checksum error, load aborted.
L@a?	-	Memory limit violation at address a, load aborted
M@a?	-	Memory failure at address a, load aborted.

2.5 S16DGS - DIAGNOSTICS

The Series 1600 diagnostic program package consists of a combined CPU-memory test and an interrupt system test. The primary purpose of the diagnostic programs is to establish operational confidence in the major microcomputer system components. If the diagnostic programs run without error and the resident monitor functions properly, the chance that the Microcomputer is not in proper working order is remote.

2.5.1 CPU-MEMORY TEST

The CPU-memory test exercies the GIC1600 CPU, memory and bus by storing and reading various data patterns throughout a user specified area of memory. The data patterns consist of all zeros, all ones, alternate zeros and ones, shifting ones and shifting zeros, etc. Address decoding is also checked by storing a data pattern throughout the memory area and then storing a different pattern in each subsequent memory location while checking that all other locations are unchanged.

Upon initial start up the user is requested to specify the address limits of the area to be tested and also the number (octal) of test passes to be run. If the specified test area overlaps the program, a question mark "?" is printed and the memory limits are requested again. When the test is complete, the number of errors detected are printed and control is returned to the resident monitor. Any errors detected are reported by descriptive printouts on the TTY as the test proceeds. Since the program is supplied in RLM (relocatable load module) format, any area of memory can be checked by relocating the program when it is loaded.

2.5.2 INTERRUPT TEST

The interrupt test exercies the Microcomputer interrupt facilities by driving the teletype and high speed tape reader and punch. Upon initial start up the user is instructed as to which keyboard characters are used as controls. The test starts by pun ching characters on the high speed reader which are entered on the teletype keyboard. These characters may also be output to the teletype. The second part of the test reads the tape output in the first part on the high speed reader and punches on the high speed punch. Note that the test does not check the devices for data integrity, but such tests can easily be implemented as required utilizing the resident utility routines. The characters used to control the interrupt test are as follows:

CTRL/C	-	Cancel, i.e., stop and return to resident monitor
CTRL/G	-	Go, i.e., start tape output
CTRL/S	-	Restart test
CTRL/R	-	Start output to TTY
CTRL/T	-	Stop output to TTY
CTRL/D	-	End tape output, start high speed reader

Appendix A. 1

INSTRUCTION SET

REGISTER - REGISTER

MNEMONIC	OPERAND	CYCLES	INSTRUCTION			DESCRIPTION	STATUS CHANGE
MOVR	SSS, DDD	6 *	0010	SSS	DDD	MOVe contents of Register SSS to register DDD. *If DDD is 6 or 7 add 1 to Cycles.	S, Z
TSTR	SSS	6 *	0010	SSS	SSS	TeST contents of Register SSS. *If SSS is 6 or 7 add 1 to Cycles.	S, Z
JR	SSS	7	0010	SSS	111	Jump to address in Register SSS. (Move address to Register 7).	S, Z
ADDR	SSS, DDD	6	0011	SSS	DDD	ADD contents of Register SSS to contents of register DDD. Results to DDD	S, Z, C, OV
SUBR	SSS, DDD	6	0100	SSS	DDD	SUBtract of Register SSS from contents of register DDD. Results to DDD	S, Z, C, OV
CMPR	SSS, DDD	6	0101	SSS	DDD	CoMPare Register SSS with register DDD by subtraction. Results not stored.	S, Z, C, OV
ANDR	SSS, DDD	6	0110	SSS	DDD	logical AND contents of Register SSS with contents of register DDD.Results to DDD	S, Z
XORR	SSS, DDD	6	0111	SSS	DDD	eXclusive OR contents of Register SSS with contents of register DDD.Results to DDD	S, Z
CLRR	DDD	6	0111	DDD	DDD	CLeaR Register to zero.	S, Z
INCR	DDD	6	0000	001	DDD	INCrement contents of Register DDD. Results to DDD	S, Z
DECR	DDD	6	0000	010	DDD	DECrement contents of Register DDD. Results to DDD	S, Z
COMR	DDD	6	0000	011	DDD	one's COMplement contents of Register DDD. Results to DDD	S, Z
NEGR	DDD	6	0000	100	DDD	Two's complement contents of Register DDD. Results to DDD	S, Z, C, OV
ADCR	DDD	6	0000	101	DDD	ADd Carry bit to contents of Register DDD. Results to DDD	S, Z, C, OV

REGISTER SHIFT

Executable only with Register 0, 1, 2, 3.
Shift Right instructions set the S flip-flop with Bit 7 of the result after the instruction.
Add 2 cycles if shift is 2 bits or two bytes.
Shifts are not interruptable.

SWAP	RR<n>	6	0001	000	NRR	N = 0, SWAP bytes of register RR. S equals Bit 7 of results of SWAP.	S, Z
		8				N = 1, SWAP bytes of register RR, then swap them back to original form.	S, Z
SLL	RR<n>	6	0001	001	NRR	N = 0. Shift Logical Left one bit, zero to low bit.	S, Z
		8				N = 1, Shift Logical Left two bits, zero to low 2 bits.	S, Z
RLC	RR<n>	6	0001	010	NRR	N = 0, Rotate Left one bit using Carry bit as bit 16.	S, Z, C
		8				N = 1, Rotate Left two bits using C as bit 17 and OV as bit 16.	S, Z, C, OV
SLLC	RR<n>	6	0001	011	NRR	N = 0, Shift Logical Left one bit using C as bit 16, zero to low bit.	S, Z, C
		8				N = 1, Shift Logical Left two bits using C as bit 17, OV as bit 16, zero to low 2 bits.	S, Z, C, OV
SLR	RR<n>	6	0001	100	NRR	N = 0, Shift Logical Right one bit, zero to high bit.	S, Z
		8				N = 1, Shift Logical Right two bits, zero to high two bits.	S, Z
SAR	RR<n>	6	0001	101	NRR	N = 0, Shift Arithmetic Right one bit, sign bit copied to high bit.	S, Z
		8				N = 1, Shift Arithmetic Right two bits, sign bit copied to high bits.	S, Z
RRC	RR<n>	6	0001	110	NRR	N = 0, Rotate Right one bit using Carry as bit 16.	S, Z, C
		8				N = 1, Rotate Right two bits using C as bit 16, OV as bit 17.	S, Z, C, OV
SARC	RR<n>	6	0001	111	NRR	N = 0, Shift Arithmetic Right one bit, thru Carry, sign bit copied to high bit.	S, Z, C
		8				N = 1, Shift Arithmetic Right two bits, thru Carry and OV, sign bit copied to high 2 bits	S, Z, C, OV

INSTRUCTION SET (continued)

BRANCHES The Branch instructions are Program Counter Relative, i.e., the Effective Address = PC±Displacement. PPPPPPPPPP is the Displacement and S is 0 for +, 1 for −. If Memory is greater than 10 bits then the appropriate number of lead bits pppppp will be a part of the Displacement. For a forward branch an addition is performed; for a backward branch a ones complement subtraction is performed. Computation performed on PC+2.

MNEMONIC	OPERAND	CYCLES	INSTRUCTION	DESCRIPTION	STATUS CHANGE
B	DA	7/9	1000 S0 0000 pppppp PPPP PP PPPP	Branch unconditional, Program Counter Relative (+1025 to -1024)	
NOPP		7	1000 S0 1000 pppppp PPPP PP PPPP	NoOPeration, two words	
BC	DA	7/9	1000 S0 0001 pppppp PPPP PP PPPP	Branch on Carry. C = 1	
BLGT	DA			Branch if Logical Greater Than. C = 1	
BNC	DA	7/9	1000 S0 1001 pppppp PPPP PP PPPP	Branch on No Carry. C = 0	
BLLT	DA			Branch if Logical Less Than. C = 0	
BOV	DA	7/9	1000 S0 0010 pppppp PPPP PP PPPP	Branch on OVerflow. OV = 1	
BNOV	DA	7/9	1000 S0 1010 pppppp PPPP PP PPPP	Branch on No OVerflow. OV = 0	
BPL	DA	7/9	1000 S0 0011 pppppp PPPP PP PPPP	Branch on PLus. S = 0	
BMI	DA	7/9	1000 S0 1011 pppppp PPPP PP PPPP	Branch on MInus. S - 1	
BZE	DA	7/9	1000 S0 0100 pppppp PPPP PP PPPP	Branch on ZEro. Z = 1	
BEQ	DA			Branch if EQual. Z - 1	
BNZE	DA	7/9	1000 S0 1100 pppppp PPPP PP PPPP	Branch on No ZEro. Z = 0	
BNEQ	DA			Branch if Not EQual. Z = 0	
BLT	DA	7/9	1000 S0 0101 pppppp PPPP PP PPPP	Branch if Less Than. S ⊻ OV = 1	
BGE	DA	7/9	1000 S0 1101 pppppp PPPP PP PPPP	Branch if Greater than or Equal. S ⊻ OV = 0	
BLE	DA	7/9	1000 S0 0110 pppppp PPPP PP PPPP	Branch if Less than or Equal. Z V (S ⊻ OV) = 1	
BGT	DA	7/9	1000 S0 1110 pppppp PPPP PP PPPP	Branch if Greater Than. Z V (S ⊻ OV) = 0	
BUSC	DA	7/9	1000 S0 0111 pppppp PPPP PP PPPP	Branch if Unequal Sign and Carry C ⊻ S = 1	
BESC	DA	7/9	1000 S0 1111 pppppp PPPP PP PPPP	Branch if Equal Sign and Carry C ⊻ S = 0	
BEXT	DA, E	7/9	1000 S1 EEEE pppppp PPPP PP PPPP	Branch if EXternal condition is True. Field E is externally decoded to select 1 of 16 conditions. Response is tested for true condition.	

INSTRUCTION SET (continued)

CONTROL

MNEMONIC	OPERAND	CYCLES	INSTRUCTION			DESCRIPTION	STATUS CHANGE
GSWD	DD	6	0000	110	0DD	Get Status WorD in register DD. Bits 0-3, 8-11 set to 0. Bits 4, 12 = C; 5, 13 = OV; 6, 14 = Z; 7, 15 = S.	
NOP	<n>	6	0000	110	10N	No operation.	
SIN	<n>	6	0000	110	11N	Software Interrupt; pulse to PCIT * pin	
RSWD	SSS	6	0000	111	SSS	Restore Status Word from register SSS; Bit 4 to C. Bit 5 to OV, Bit 6 to Z, Bit 7 to S.	S, Z, C, OV
HLT		4	0000	000	000	HaLT after next instruction is executed. Resume on control start.	
EIS		4	0000	000	010	Enable Interrupt System. Not Interruptable.	
DIS		4	0000	000	011	Disable Interrupt System. Not Interruptable.	
TCI		4	0000	000	101	Terminate Current Interrupt. Not Interruptable.	
CLRC		4	0000	000	110	CLeaR Carry to zero. Not Interruptable.	C
SETC		4	0000	000	111	SET Carry to one. Not Interruptable.	C

JUMP

J	DA	12	0000 11AA AAAA	000 AAA AAA	100 A00 AAA	Jump to address. Program counter is set to 16 bits of A's.	
JE	DA	12	0000 11AA AAAA	000 AAA AAA	100 A01 AAA	Jump to address. Enable interrupt system. Program counter is set to 16 bits of A's.	
JD	DA	12	0000 11AA AAAA	000 AAA AAA	100 A10 AAA	Jump to address. Disable interrupt system. Program counter is set to 16 bits of A's.	
JSR	BB, DA	12	0000 BBAA AAAA	000 AAA AAA	100 A00 AAA	Jump and Save Return address (PC+3) in register designated by 1BB. Program counter is set to 16 bits of A's. BB≠11	
JSRE	BB, DA	12	0000 BBAA AAAA	000 AAA AAA	100 A01 AAA	Jump and Save Return and Enable interrupt system. Return (PC+3) is saved in register 1BB. Program counter is set to 16 bits of A's. BB≠11	
JSRD	BB, DA	12	0000 BBAA AAAA	000 AAA AAA	100 A10 AAA	Jump and Save Return and Disable interrupt system. Return (PC+3) is saved in register 1BB. Program counter is set to 16 bits of A's. BB≠11	

DIRECT ADDRESSED DATA - MEMORY

Field aaa aaa is dependent on the width of memory. 16 bits is maximum for aaa aaa AAAAAAAAAA.

MNEMONIC	OPERAND	CYCLES	INSTRUCTION	DESCRIPTION	STATUS CHANGE
MVO	SSS, A	11	1001 000 SSS aaa aaa AAAA AAA AAA	MoV Out data from register SSS to address A - A.	
MVI	A, DDD	10	1010 000 DDD aaa aaa AAAA AAA AAA	MoVe In data from address A - A to register DDD.	
ADD	A, DDD	10	1011 000 DDD aaa aaa AAAA AAA AAA	ADD data from address A - A to register DDD. Results to DDD.	S, Z, C, OV
SUB	A, DDD	10	1100 000 DDD aaa aaa AAAA AAA AAA	SUBtract data from address A - A from register DDD. Results to DDD.	S, Z, C, OV
CMP	A, SSS	10	1101 000 SSS aaa aaa AAAA AAA AAA	CoMPare data from address A - A with register SSS by subtraction. Results not stored.	S, Z, C, OV
AND	A, DDD	10	1110 000 DDD aaa aaa AAAA AAA AAA	logical AND data from address A - A with register DDD. Results to DDD.	S, Z
XOR	A, DDD	10	1111 000 DDD aaa aaa AAAA AAA AAA	eXclusive OR data from address A - A with register DDD. Results to DDD.	S, Z

INDIRECT ADDRESSED DATA - REGISTER

MMM Source data is located at the address contained in Register.
MMM = 4, 5 post increment R4 or R5.
MMM = 6 - MVO instruction - post increment R6. PUSH data from Register SSS to the Stack.
Other instructions - pre-decrement R6. PULL data from the Stack to be used as the first operand.

MVO@	SSS, MMM	9	1001 MMM SSS	MoVe Out data from register SSS to the address in register MMM. Note: SSS = MMM = 4, 5, 6 or 7 not supported.	
PSHR	SSS	9	1001 110 SSS	PuSH data from Register SSS to the stack.	
MVI@	MMM, DDD	8 *	1010 MMM DDD	MoVe In data to register DDD from address in register MMM.	
PULR	DDD	11	1010 110 DDD	PULl data from the stack to Register DDD.	
ADD@	MMM, DDD	8 *	1011 MMM DDD	ADD data located at address in register MMM to the contents of register DDD. Results to DDD.	S, Z, C, OV
SUB@	MMM, DDD	8 *	1100 MMM DDD	SUBtract data located at address in Register MMM from contents of register DDD. Results to DDD.	S, Z, C, OV
CMP@	MMM, DDD	8 *	1101 MMM SSS	CoMPare data located at address in Register MMM with contents of register SSS, by subtraction. Results not stored.	S, Z, C, OV
AND@	MMM, DDD	8 *	1110 MMM DDD	logical AND contents of register DDD with data located at address in register MMM. Results to DDD.	S, Z
XOR@	MMM, DDD	8 *	1111 MMM DDD	eXclusive OR contents of register DDD with data located at address in register MMM. Results to DDD.	S, Z

*Add 3 to number of cycles if MMM=6.

IMMEDIATE DATA - REGISTER The number of iiiiii bits depends on the memory width, 16 bits is maximum.

MNEMONIC	OPERAND	CYCLES	INSTRUCTION	DESCRIPTION	STATUS CHANGE
MVOI	SSS,I	9	1001 111 SSS iiiiii IIII III III	MoVe Out Immediate data from register SSS to PC+1 (field)	
MVII	I,DDD	8	1010 111 DDD iiiiii IIII III III	MoVe In Immediate data to register DDD from PC+1 (field).	
ADDI	I,DDD	8	1011 111 DDD iiiiii IIII III III	ADD Immediate data to contents of register DDD. Results to DDD.	S, Z, C, OV
SUBI	I,DDD	8	1100 111 DDD iiiiii IIII III III	SUBtract Immediate data from contents of register DDD. Results to DDD.	S, Z. C. OV
CMPI	I,SSS	8	1101 111 SSS iiiiii IIII III III	CoMPare Immediate data from contents of register SSS by subtraction. Results not stored.	S, Z, C, OV
ANDI	I,DDD	8	1110 111 DDD iiiiii IIII III III	logical AND Immediate data with contents of register DDD. Results to DDD.	S, Z
XORI	I,DDD	8	1111 111 DDD iiiiii IIII III III	eXclusive OR Immediate data with contents of register DDD. Results to DDD.	S, Z
SDBD		4	0000 000 001 This instruction is normally supplied by the assembler as required to properly generate machine code.	Set Double Byte Data for the next instruction which must be an external reference instruction. The effective address of the external reference instruction will address the low order data byte; the address of the high order data byte will be EA+1 if register 4, 5 or 7 is used. If register 1-3 is used the EA will access the same byte twice resulting in both bytes of data being the same. Use of modes 0 and 6 are not supported by this instruction.	

INDIRECT ADDRESSED DOUBLE BYTE DATA - REGISTER

SDBD		4	0000	000	001	MoVe In double byte data from the address in register MMM to register DDD.	
MVI@	MMM, DDD	10	1010	MMM	DDD		
SDBD		4	000	000	001	ADD double byte data from the address in register MMM to the content of register DDD. Results to DDD.	S, Z, C, OV
ADD@	MMM, DDD	10	1011	MMM	DDD		
SDBD		4	0000	000	001	SUBtract double byte data located at address MMM from the content of register DDD. Results to DDD.	S, Z, C, OV
SUB@	MMM, DDD	10	1100	MMM	DDD		
SDBD		4	0000	000	001	CoMPare double byte data located at address in register MMM with the content of register SSS by subtraction. Results is not stored.	S, Z, C, OV
CMP@	MMM, DDD	10	1101	MMM	SSS		
SDBD		4	0000	000	001	logical AND double byte data located at address in register MMM with the content of register DDD. Results to DDD.	S, Z
AND@	MMM, DDD	10	1110	MMM	DDD		
SDBD		4	0000	000	001	eXclusive OR double byte data located at address in register MMM with the content of register DDD. Results to DDD.	S, Z
XOR@	MMM, DDD	10	111	MMM	DDD		

IMMEDIATE DOUBLE BYTE DATA - REGISTER

Note: The SDBD command is provided by the assembler when the immediate data is greater than the memory width and requires two bytes.

MVII	I,DDD	14	0000 1010 XXLL XXUU	000 111 LLL UUU	001 DDD LLL UUU	MoVe In Immediate double byte data to register DDD. L's will be low byte and U's upper byte. XX = don't care.	
ADDI	I,DDD	14	0000 1011 XXLL XXUU	000 111 LLL UUU	001 DDD LLL UUU	ADD Immediate double byte data to contents of register DDD. Results to DDD. L's indicate low byte of literal, U's upper byte.	S, Z, C, OV
SUBI	I,DDD	14	0000 1100 XXLL XXUU	000 111 LLL UUU	001 DDD LLL UUU	SUBtract Immediate double byte data from contents of register DDD. Results to DDD. L's indicate low byte of literal, U's upper byte.	S, Z, C, OV
CMPI	I,SSS	14	0000 1101 XXLL XXUU	000 111 LLL UUU	001 SSS LLL UUU	CoMPare Immediate double byte data with contents of register SSS by subtraction. Results not stored. L's indicate low byte of literal, U's upper byte.	S, Z, C, OV
ANDI	I,DDD	14	0000 1110 XXLL XXUU	000 111 LLL UUU	001 DDD LLL UUU	logical AND Immediate double byte data with the contents of Register DDD. Results to register DDD. L's indicate low byte of literal, U's upper byte.	S, Z
XORI	I,DDD	14	0000 1111 XXLL XXUU	000 111 LLL UUU	001 DDD LLL UUU	eXclusive OR Immediate double byte data with the contents of register DDD. Results to Register DDD. L's indicate low byte of literal, U's upper byte.	S, Z

GLOSSARY OF TERMS

SSS - Source Register
DDD - Destination Register
n - Number of Shifts
RR - Register to Shift (only 0-3 allowed)
AAAAAA Memory address for Jump.
AAAAAAAAAA (new Program Counter)
BB - Register to save old PC in for Jump. (Reg = 1BB, 4, 5, or 6)
S - Sign of address displacement for Branch (PC relative).
pppppp PPPPPPPPPP - Address displacement for Branch pppppp is dependent on the memory word size.
aaaaaa AAAAAAAAAA - Direct address of data word. aaaaaa is dependent on the memory word size.
iiiiii IIIIIIIIII - Immediate data word. iiiiii is dependent on memory word size.
LLLLLLLL Lower 8 bits of double byte data.
UUUUUUUU Upper 8 bits of double byte data.

MMM - Address Mode
- 000 - direct address in location following instruction.
- 001 - indirect address for Register 1
- 010 - indirect address for Register 2
- 011 - indirect address for Register 3
- 100 - indirect address for Register 4, post increment
- 101 - indirect address for Register 5, post increment
- 110 - indirect address for Register 6, post increment for MVO only
  indirect address for Register 6, pre decrement for all instructions except MVO.
- 111 - indirect address for Register 7, post increment. (Immediate data in location following instruction.)

APPENDIX A.2

MONITOR COMMANDS

A	-	Inspect/modify address
B	-	Set breakpoint
C	-	Continue from breakpoint
DA	-	Display contents of addresses.
DB	-	Display breakpoints
DO	-	Display Origins
DR	-	Display contents of registers
DT	-	Display trap address
E	-	Execute
IA	-	Initialize contents of addresses
K	-	Copy tape
L	-	Load tape
MB	-	Modify branch destination
MJ	-	Modify jump destination
ML	-	Set memory limits
O	-	Set module origin
P	-	Punch tape
R	-	Inspect/modify register
SA	-	Search addresses
SW	-	Inspect/modify status word
T	-	Trap to address when SIN executed

APPENDIX A.3

SAMPLE MONITOR DIALOGUE

In the following examples "↵" represents a carriage return character.

Initial start up

```
S16ODM V01B          - Identifying message
$                    - Command prompt
```

Comments

```
$ ; commentary↵
$
```

Display registers

```
$DR↵
SR=000066
R0=000066  R1=000066  R2=000066  R3=000066
R4=000066  R5=000066  R6=000066  R7=170000
$
```

Inspect register

```
$R2↵
003201:↵
$
```

Modify register

```
$R2↵
002053:25↵
$
```

Inspect status word

```
$SW↵
000160:↵
$
```

Modify status word

```
$SW↵
000160:120↵
$
```

Display addresses

```
$DA202,213↵
000200  177777  177760  011064  125715  007417  007417  170360  170360
000210  125252  052525  052525  125252  000000  000001  000002  000003
$
```

Inspect address

```
$A2300↵
002300=100502:↵
$
```

Inspect addresses sequentially

```
$A000302↵
000302=000001:/
000303=050021:/
000304=027770:-
000303=050021:-
000302=000001:-
000301=012345:↵
```

- note that "/" causes the next addresses to be displayed.
- note that "-" causes the previous address to be displayed.
- note that carriage return terminates activity.

Modify address

```
$A5037↵
005037=177777:0↵
$
```

Modify addresses sequentially

```
$A010↵
000010=000201:0/
000011=005007:0/
000012=000222:0/
000013=001000:2-
000012=000000:1↵
$
```

Search addresses

```
$SA205,321,50,70↵
000212=120357
000272=000050
000307=155555
$
```

- note that any address containing a 5 in bits $2^5, 2^4, 2^3$ is displayed.
  70 is used to AND out appropriate bits for comparison.

Initialize addresses

```
$IA5230,5501,2,7↵
$
```

- addresses 5230-5501 will have the lower 3 bits (i.e., mask of 7) set to 010 (i.e., value of 2).

Load tape from low speed reader

```
$LL↵
INIT ADDR 000000
FINL ADDR 000020
$
```

- load summary

Load and relocate tape from high speed reader

```
$LH350↵
INIT ADDR 000350
FINL ADDR 001050
$
```

- load summary

Note: If an error or an invalid tape is detected the load is aborted and an "E" or "T" is typed instead of the load summary.

Punch on low speed punch

```
$PL250,300↵
PNCH ON!↵
```

- note that the user must manually turn the punch on and then respond with a carriage return. During punching binary the teletype printer remains on and may print erroneous data because of the binary data being punched.

```
PNCH OFF
$
```

- note that the user must turn the punch off or all subsequent dialogue will be punched on tape.

Punch on high speed punch

```
$PH505,3000↵
$
```

Step program one instruction at a time

```
$S300↵
$S↵
$S↵
$
```

- start stepping at address 300
- step next instruction
- step next instruction

Note that break points are deactivated during program stepping.

Execute program

$E1003↵ - begin execution at address 1003.

Modify branch instruction destination

$MB200,502↵
$ - Modify branch instruction at address 200 & 201 for destination address 502. If the instruction is not a branch, ODP responds with "?" and no modification takes place.

Modify jump instruction destination

$MJ3210,200↵
$ - Modify jump instruction at address 3210, 3211 & 3212 for destination address 200. If the instruction is not a jump, ODP responds with "?" and no modification takes place.

Set break points

$B2,500↵ - Set break point 2 at address 500.
$B,202↵ - Set first available break point at address 202.
$

Display breakpoints

$DB↵
B0:000202
B2:000500
$

Remove break points

$B0↵ - Remove breakpoint 0
$B↵ - Remove all breakpoints
$

ODP entry via a break point

B3@001025 - Indicates break point reached
$ - Command prompt

Continue from a break point

$C↵

Set program relocation origins

$O3, 752↵	Set origin 3 to address 752.
$O, 15305↵	Set first available origin to address 15305.

Display program relocation origins

```
$DO↵
O0:015305
O3:000752
```

Remove program relocation origins

$O0↵	Remove origin 0.
$O↵	Remove all origins.

Addressing using origins

$AO3+4↵ 000757=xxxxxx:	Inspect the third location in the program module starting at address 752.

Deletion of input lines

```
$DW200, 300← $R2↵
005000:
$A20↵
000020:135←:105↵
```

- note that "←" deletes the entire line, i.e., $ prompt again

Deletion of characters

```
$R5•\5\2↵
000025:137•\7•3\27↵
```

- note that • represents a rubout and ODP responds with the character deleted enclosed in "\".

APPENDIX A.4

RESIDENT UTILITY ROUTINES

ODM	@	171000	-	On-line Debug Monitor entry
BINOCT	@	175000	-	Binary to octal conversion
OCTBIN	@	175002	-	Octal to binary conversion
TYP STR	@	175004	-	Type character string
TYPR2	@	175006	-	Type character in R2
TYPOCT	@	175010	-	Type octal value in R0
TYCRLF	@	175012	-	Type carriage return and line feed
SELDEV	@	175014	-	Select tape device
PUNLDR	@	175016	-	Punch tape leader on selected device
PUNWORD	@	175020	-	Punch word on selected device
PUNR2	@	175022	-	Punch byte in R2 on selected device
RDFRM	@	175024	-	Read one frame from selected device
INURSP	@	175026	-	Prompt user with ":" and input character string
CHKUSR	@	175030	-	Check for user CTRL/C (cancel).
INSTR	@	175032	-	Input character string

```
INASC     GI  S16AL V01A      PAGE   1

                          REL  INASC
                    ;;;;;;
                    ;
                    ;  ASCII TO BINARY CONVERSION ROUTINE
                    ;
                    ;   HEXBIN - HEXADECIMAL ASCII TO BINARY
                    ;   DECBIN - DECIMAL INTEGER ASCII TO BINARY
                    ;   OCTBIN - OCTAL ASCII TO BINARY
                    ;   BINBIN - BINARY ASCII TO BINARY
                    ;
                    ;  CALLING SEQUENCE:
                    ;   R1 = INPUT FIELD BASE ADDRESS
                    ;   R2 = # CHARACTERS TO BE CONVERTED
                    ;   JSR R5,NAME
                    ;  OUTPUTS:
                    ;   R0 = CONVERTED BINARY VALUE
                    ;   R1 = POINTER TO END OF CONVERSION
                    ;   R2 - R4 DESTROYED
                    ;
                    ;  CONVERSION TERMINATES ON FIRST NON NUMERIC
                    ;  CHARARACTER ENCOUNTERED.  LEADING SPACES ARE
                    ;  IGNORED.  LEADING + OR - ARE HANDLED.
                    ;
                    ;;;;;;

                    R0      EQU  0
                    R1      EQU  1
                    R2      EQU  2
                    R3      EQU  3
                    R4      EQU  4
                    R5      EQU  5
                    SP      EQU  6
                    PC      EQU  7

                          GLOB HEXBIN,DECBIN,OCTBIN,BINBIN

 000000 001274      HEXBIN MVII .16,R4     ;RADIX 16
 000001 000020
 000002 001000             B    ASC1
 000003 000012
 000004 001274      DECBIN MVII .10,R4     ;RADIX 10
 000005 000012
 000006 001000             B    ASC1
 000007 000006
 000010 001274      OCTBIN MVII .8,R4      ;RADIX 8
 000011 000010
 000012 001000             B    ASC1
 000013 000002
 000014 001274      BINBIN MVII 2,R4       ;RADIX 2
 000015 000002
 000016 000223      ASC1   MOVR R2,R3      ;# CHRS
 000017 001165             PSHR R5         ;SAVE RETURN
 000020 000700             CLRR R0         ;INIT BIN ACCUM
 000021 000755             CLRR R5         ;INIT STR STRT FLG
 000022 001212      ASC2   MVI@ R1,R2      ;PICK UP CHR
 000023 000255             TSTR R5         ;STR STRT YET ?
 000024 001014             BNZE ASC4       ;YES, NO LDNG CHRS
 000025 000020
 000026 001572             CMPI ' ',R2     ;SPC ?
```

```
INASC      GI   S16AL V01A       PAGE   2

     000027 000040
  52 000030 001004          BEQ  ASC7       ;YES, BYPASS
     000031 000063
  53 000032 001572          CMPI '-',R2     ;MINUS ?
     000033 000055
  54 000034 001014          BNEQ ASC3       ;NO, CHK FOR PLUS
     000035 000003
  55 000036 000025          DECR R5         ;YES, SET MINUS FLG
  56 000037 001000          B    ASC7       ;BYPASS MINUS
     000040 000054
  57 000041 000015  ASC3    INCR R5         ;SET PLUS FLG
  58 000042 001572          CMPI '+',R2     ;PLUS ?
     000043 000053
  59 000044 001004          BEQ  ASC7       ;YES, BYPASS IT
     000045 000047
  60 000046 001472  ASC4    SUBI 060,R2     ;STRIP ASCII MASK
     000047 000060
  61 000050 001013          BMI  ASCFIN     ;NON DIG, TRMN CNVRT
     000051 000047
  62 000052 001572          CMPI 021,R2     ;CHK FOR A-F
     000053 000021
  63 000054 001005          BLT  ASC5       ;NOT
     000055 000006
  64 000056 001572          CMPI 026,R2
     000057 000026
  65 000060 001016          BGT  ASCFIN     ;NON DIG, TRMN CNVRT
     000061 000037
  66 000062 001472          SUBI 07,R2      ;ADJ A-F -> 10-15
     000063 000007
  67 000064 000542  ASC5    CMPR R4,R2      ;CMPR DIG & # BASE
  68 000065 001015          BGE  ASCFIN     ;NON DIG, TRMN CNVRT
     000066 000032
  69 000067 001574          CMPI .10,R4     ;CHK FOR DEC CNVRT
     000070 000012
  70 000071 001004          BEQ  ASC10
     000072 000033
  71 000073 000130          SLLC R0         ;MULT ACCUM BY 2
  72 000074 001001          BC   ASCFIN
     000075 000023
  73 000076 001574          CMPI 2,R4       ;CHK FOR BASE 2
     000077 000002
  74 000100 001004          BEQ  ASC6
     000101 000012
  75 000102 000134          SLLC R0,2       ;MULT ACCUM BY 8
  76 000103 001001          BC   ASCFIN
     000104 000014
  77 000105 001574          CMPI .8,R4      ;CHK FOR BASE 8
     000106 000010
  78 000107 001004          BEQ  ASC6
     000110 000003
  79 000111 000130          SLLC R0         ;MULT ACCUM BY 16
  80 000112 001001          BC   ASCFIN
     000113 000005
  81 000114 000320  ASC6    ADDR R2,R0      ;INSRT CURRNT DIG
  82 000115 000011  ASC7    INCR R1         ;INCR CHR STR PTR
  83 000116 000023          DECR R3         ;CHK FOR ALL CHRS CNVRT
  84 000117 001054          BNZE ASC2       ;NOT, GET NXT CHR
     000120 000076
  85 000121 000255  ASCFIN  TSTR R5         ;CHK SIGN FLG
```

```
INASC      GI  S16AL V01A      PAGE   3

  86 000122 001003          BPL  ASCXIT     ;PLUS
     000123 000001
  87 000124 000040          NEGR R0         ;MINUS
  88 000125 001267  ASCXIT  PULR PC         ;EXIT
  89 000126 001162  ASC10   PSHR R2         ;SAVE CURR DIG
  90 000127 000202          MOVR R0,R2      ;MULT ACCUM BY 10
  91 000130 000134          SLLC R0,2
  92 000131 001041          BC   ASCFIN
     000132 000011
  93 000133 000420          SUBR R2,R0
  94 000134 000130          SLLC R0
  95 000135 001041          BC   ASCFIN
     000136 000015
  96 000137 000420          SUBR R2,R0
  97 000140 000130          SLLC R0
  98 000141 001041          BC   ASCFIN
     000142 000021
  99 000143 001262          PULR R2         ;GET CURR DIG
 100 000144 001040          B    ASC6       ;INSRT DIG
     000145 000031
 101 000145                 END


          ASCFIN 000121  R IN
          ASCXIT 000125  R IN
          ASC1   000016  R IN
          ASC10  000126  R IN
          ASC2   000022  R IN
          ASC3   000041  R IN
          ASC4   000046  R IN
          ASC5   000064  R IN
          ASC6   000114  R IN
          ASC7   000115  R IN
          BINBIN 000014  R IN G   UR
          DECBIN 000004  R IN G   UR
          HEXBIN 000000  R IN G   UR
          OCTBIN 000010  R IN G   UR
          PC     000007  A EQ
          R0     000000  A EQ
          R1     000001  A EQ
          R2     000002  A EQ
          R3     000003  A EQ
          R4     000004  A EQ
          R5     000005  A EQ
          SP     000006  A EQ     UR

     22 SYMBOLS

     NO ERRORS

PASS 2 AGAIN?:
```

APPENDIX A.6

TEXT EDITOR COMMANDS

A	-	Advance character pointer n lines
B	-	Back character pointer to beginning of text
C	-	Change n characters in a line
D	-	Delete n characters in a line
E	-	End, write out text, a form feed and blank trailer
F	-	Output form feed
G	-	Get character string
I	-	Insert lines
J	-	Jump character pointer n positions
K	-	Kill n lines
L	-	List n lines
M	-	Mark character position
N	-	Next segment, output current text, read next segment
O	-	Open, output blank leader
P	-	Purge deleted text
Q	-	Quote text
R	-	Read text into buffer
S	-	Stop
T	-	Tab set
V	-	Verify character position
W	-	Write out current text
X	-	Exchange current line
Y	-	Yank deleted text
ZE	-	Zero buffer
#	-	Print current line number
=	-	Print number of free character positions in buffer
7	-	Accept 7-bit ASCII input
8	-	Accept 8-bit ASCII input

APPENDIX A.7

SAMPLE TEXT EDITOR DIALOGUE

In the following dialogue user input is underlined and "↙" represents a carriage return.

Initial start up

```
S16TXE V01A                  - identifying message
BUFFR LO ADR 003325          - text buffer low address
BUFFR HI ADR?:7777           - specify text buffer limit at 7777
*:                           - command prompt

*:R↙                         - read text
DEV?:H                       - specify high speed tape reader
*: L3↙                       - list first 3 lines
 REL PROG1
R2 EQU 2
PROG1 MVII 5, R2 ;INIT COUNT
*:G↙                         - get first occurrence of character string
:PR                          - "PR"
*:V↙                         - verify character position
OG1
*:C3↙                        - change next 3 characters
:GA↙                         - to "GA"
*:L↙                         - list current line
 REL PRGA
*:A2↙                        - advance two lines
*:V↙                         - verify character position
PROG1 MVII 5, R2 ;INIT COUNT
*:J2↙                        - jump two character positions
*:D↙                         - delete next character
*:J1↙                        - jump one character position
*:C↙                         - change next character
:A↙                          - to "A"
*:L↙                         - list current line
PRGA MVII 5, R2 ;INIT COUNT
*:G↙                         - get next occurrence of
:PROG1↙                      - "PROG1"
*:J-5↙                       - jump back five character positions
*:C5↙                        - change next five characters
:PRGA↙                       - to "PRGA"
*:L↙                         - list current line
 B PRGA ;LOOP AGAIN
*:G↙                         - get next occurence of
:PROG1↙                      - "PROG1"
NONE!                        - no more "PROG1" strings exist
*:L↙                         - list current line
```

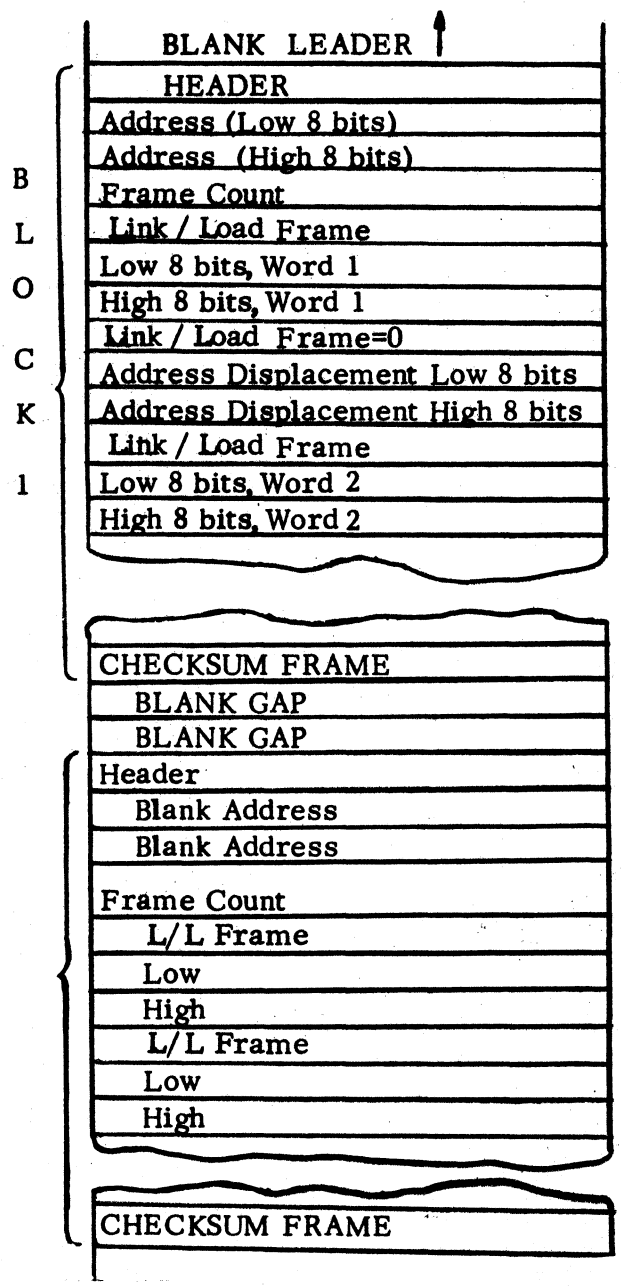
*:I↵	- at end of text, no line listed, insert lines
:K8 WORD .8↵	- line to be inserted
: END↵	- line to be inserted
:↵	- null line, i.e., end of insertion
*:O↵	- open output tape
DEV?:H↵	- specify high speed punch
*:E	- end editing, i.e., write text, form feed, trailer
*:S↵	- stop, i.e., return to resident monitor

APPENDIX A.8

BINARY TAPE FORMATS

RELOCATABLE PAPER TAPE FORMAT

Binary paper tapes produced by S16AL consist of variable length records which contain a four frame header and up to 132 data frames. The first significant frame in all records indicates a relocatable or absolute tape, 001 or 002 respectively (377 or 376 in the last record). The second and third frames in the first record contain the assembly base address or origin (low byte, high byte respectively); in subsequent records these two frames have no significance. The fourth frame contains the number of object data frames in the remainder of the record. The last data frame is followed by a record checksum frame which is used during loading to verify that the record has been read correctly. Object code sequences are the same as in a relocatable binary file except that the link/load code occupies one tape frame and each object data word occupies two tape frames, low byte, high byte respectively. The first record on a tape is preceded by approximately 50 frames of blank leader, the last record is followed by blank trailer of the same length and each record is separated by two blank frames.



HEADER CODES

1 - Relocatable
2 - Absolute
-1 - Last Block of Relocatable
-2 - Last Block of Absolute

RELOCATABLE OBJECT CODE SEQUENCES

The data information in each record of a S16AL object file is grouped into sequences of variable length. The first word in each sequence contains a link/load code which indicates the number and nature of object words following in the sequence.

Code	No. Data Words	Object Word Significance
0	1	address adjustment
1	1	absolute word
2	2	absolute word
3	3	absolute word
4	1	relocatable word
5	2	absolute word, relocatable word
6	3	absolute word, 2 relocatable 8-bit bytes
7	2	2 relocatable 8-bit bytes
8	3	absolute word, 2 relocatable 10-bit bytes
9	1	external reference word
10	2	absolute word, external reference word
11	2	absolute word, external reference displacement
12	3	absolute word, 2 external reference 8-bit bytes
13	2	2 external reference 8-bit bytes
14	3	absolute word, 2 external reference 10-bit bytes
15	1	entry address word
16	2	module name
17	2	global symbol
18	2	external symbol

ABSOLUTE PAPER TAPE FORMAT

In response to the $P command, the contents of memory is punched on paper tape in absolute binary format. The data is punched as eight bit bytes organized in variable length records. Each record begins with a four byte header and ends with a checksum byte. The first byte in the record header contains a code (3, -3 in the last record) which identifies the nature of the tape. The second and third bytes contain the base address of the data in the record. The fourth byte contains the number of data bytes in the record. Each data word is punched as two bytes, low order followed by high order.

Blank Leader
Header
Address, Low 8 bits
Address High 8 bits
Record Frame Count
Data Word Low 8 bits
Data Word High 8 bits
Data Word Low 8 bits
Data Word High 8 bits
Data Word Low 8 bits
Data Word High 8 bits
Data Word Low 8 bits
Data Word High 8 bits
Checksum Frame
Blank Frame
Blank Frame
Header
Address Low 8 bits
Address High 8 bits
Record Frame Count
Data Word Low 8 bits
Data Word High 8 bits
Data Word Low 8 bits
Data Word High 8 bits
Checksum Frame
Blank Trailer

APPENDIX A.9

S16RLL SAMPLE LOAD MAP

```
S16RLL V01A
ADR?:1234
MAP?:Y
DEV?:H
<ORG 001234>
MODUL:CNVRT
 GLOBS
  IOCNVR 001234
<SIZ 000405>
DEV?:H
<ORG 001641>
MODUL:INASC
 GLOBS
  HEXBIN 001641
  INTBIN 001645
  OCTBIN 001651
  BINBIN 001655
<SIZ 000146>
DEV?:H
<ORG 002007>
MODUL:OUTASC
 GLOBS
  HEXASC 002007
  INTASC 002013
  OCTASC 002021
  BINASC 002025
<SIZ 000246>
DEV?:H
<ORG 002255>
MODUL:TTYIN
 GLOBS
  TTYIN  002255
<SIZ 000223>
DEV?:H
<ORG 002500>
MODUL:TTYOUT
 GLOBS
  TTYOUT 002500
  TYPCHR 002523
  TYPR2  002534
<SIZ 000053>
DEV?:
INIT ADDR 001234
FINL ADDR 002552
ENTR ADDR 001234

S16ODP V01A
$
```

APPENDIX A.10

ASCII CHARACTER CODES

Char	7 Bit Octal Code	Char	7 Bit Octal Code
Space	040	@	100
!	041	A	101
"	042	B	102
#	043	C	103
$	044	D	104
%	045	E	105
&	046	F	106
'	047	G	107
(	050	H	110
)	051	I	111
*	052	J	112
+	053	K	113
,	054	L	114
-	055	M	115
.	056	N	116
/	057	O	117
0	060	P	120
1	061	Q	121
2	062	R	122
3	063	S	123
4	064	T	124
5	065	U	125
6	066	V	126
7	067	W	127
8	070	X	130
9	071	Y	131
:	072	Z	132
;	073	[	133
	074	\	134
=	075	]	135
	076	↑	136
?	077	←	137

APPENDIX A.11

CP1600 – INSTRUCTION SET SUMMARY

		MNEMONICS	OPERATION	MICROCYCLES Dir.	Indr.	Imm.	Stack	COMMENTS
External Reference Instructions	Arithmetic & Logic	ADD	ADD	10	8	8	11	
		SUB	SUBtract	10	8	8	11	
		CMP	CoMPare	10	8	8	11	Result not saved
		AND	logical AND	10	8	8	11	
		XOR	eXclusive OR	10	8	8	11	
	I/O	MVO	MoVe Out	11	9	9	9	
		MVI	MoVe In	10	8	8	11	
Internal Register Instructions	Register to Register	ADDR	ADD contents of Registers	6				Add one cycle
		SUBR	SUBtract contents of Register	6				if Register 6 or 7
		CMPR	CoMPare Registers by subtr.	6				Result not saved, except*
		ANDR	logical AND Registers	6				
		XORR	eXclusive OR Registers	6				
		MOVR	MOVe Register	6				
	Single Register	CLRR	CLeaR Register	6				XORR with itself
		TSTR	TeST Register	6				
		JR	Jump to address in Register	7*				PC ← (RRR)
		INCR	INCrement Register	6				
		DECR	DECrement Register	6				
		COMR	COMplement Register	6				One's Complement
		NEGR	NEGate Register	6				Two's Complement
		ADCR	ADd Carry Bit to Register	6				
		GSWD	Get Status WorD	6				
		NOP	No OPeration	6				Two Words
		SIN	Software INterrupt	6				Pulse to PCIT pin
		RSWD	Return Status WorD	6				
		PULR	PULl from stack to Register	11*				PULR = MVI@R6
		PSHR	PuSH Register to stack	9*				PSHR = MVO@R6
	Register Shift	SLL	Shift Logical Left	6				one or two position shift capability. Add two cycles for 2-position shift
		RLC	Rotate Left thru Carry	6				
		SLLC	Shift Logical Left thru Carry	6				
		SLR	Shift Logical Right	6				
		SAR	Shift Arithmetic Right	6				
		RRC	Rotate Right thru Carry	6				
		SARC	Shift Arithmetic Right thru Carry	6				
		SWAP	SWAP 8-bit bytes	6				2-position=SWAP twice
Control Instructions		HLT	HaLT	4				
		SDBD	Set Double Byte Data	4				Must precede external reference to double byte data
		EIS	Enable Interrupt System	4				Not Interruptible
		DIS	Disable Interrupt System	4				
		TCI	Terminate Current Interrupt	4				
		CLRC	CLeaR Carry to zero	4				
		SETC	SET Carry to one	4				
Jump Instructions		J	Jump	12				
		JE	Jump, Enable, interrupt	12				
		JD	Jump, Disable interrupt	12				
		JSR	Jump, Save Return	12				Return Address saved in R4, 5 or 6
		JSRE	Jump, Save Return & Enable	12				
		JSRD	Jump, Save Return & Disable Interrupt	12				
Conditional Branch Instructions		B	unconditional Branch	7				Displacement in PC+1
		BC, BLGE	Branch on Carry, C=1	7				PC ← PC ± Displacement
		BNC, BLLT	Branch on No Carry, C=0	7				Add 2 cycles if test condition is true.
		BOV	Branch on OVerflow, OV=0	7				
		BNOV	Branch on No OVerflow, OV=0	7				
		BPL	Branch on PLus, S=0	7				
		BMI	Branch on MInus, S=1	7				
		BZE, BEQ	Branch on ZEro or EQual	7				Z=1
		BNZE, BNEQ	Branch if Not ZEro or Not EQual	7				Z=0
		BLT	Branch if Less Than	7				S∀OV=1
		BGE	Branch if Greater than or Equal	7				S∀OV=0
		BLE	Branch if Less than or Equal	7				Z V (S∀OV)=1
		BGT	Branch if Greater Than	7				Z V (S∀OV)=0
		BUSC	Branch if Sign ≠ Carry	7				C ∀ S=1
		BESC	Branch if Sign = Carry	7				C ∀ S=0
		BEXT	Branch if External condition is True	7				4 LSB of Instruction are decoded to select 1 of 16 external conditions.

1 MICROCYCLE = 2 CLOCK CYCLES

INSTRUCTION SET

15----------10	9	8 7 6	5 4 3 2	1 0
	0	0 0 1	Mode	R R

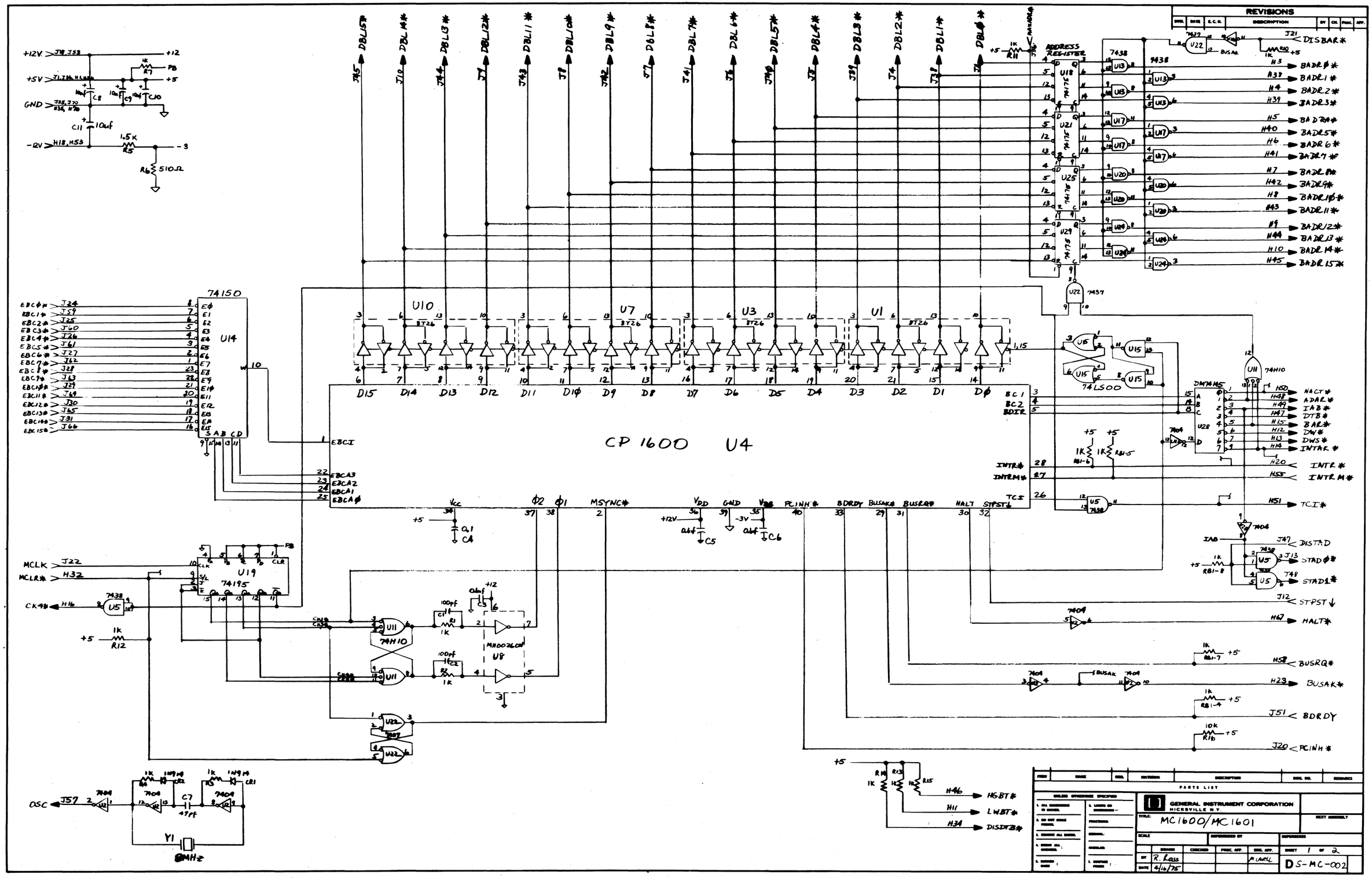
Register R0-R3 only.

SHIFT INSTRUCTIONS:

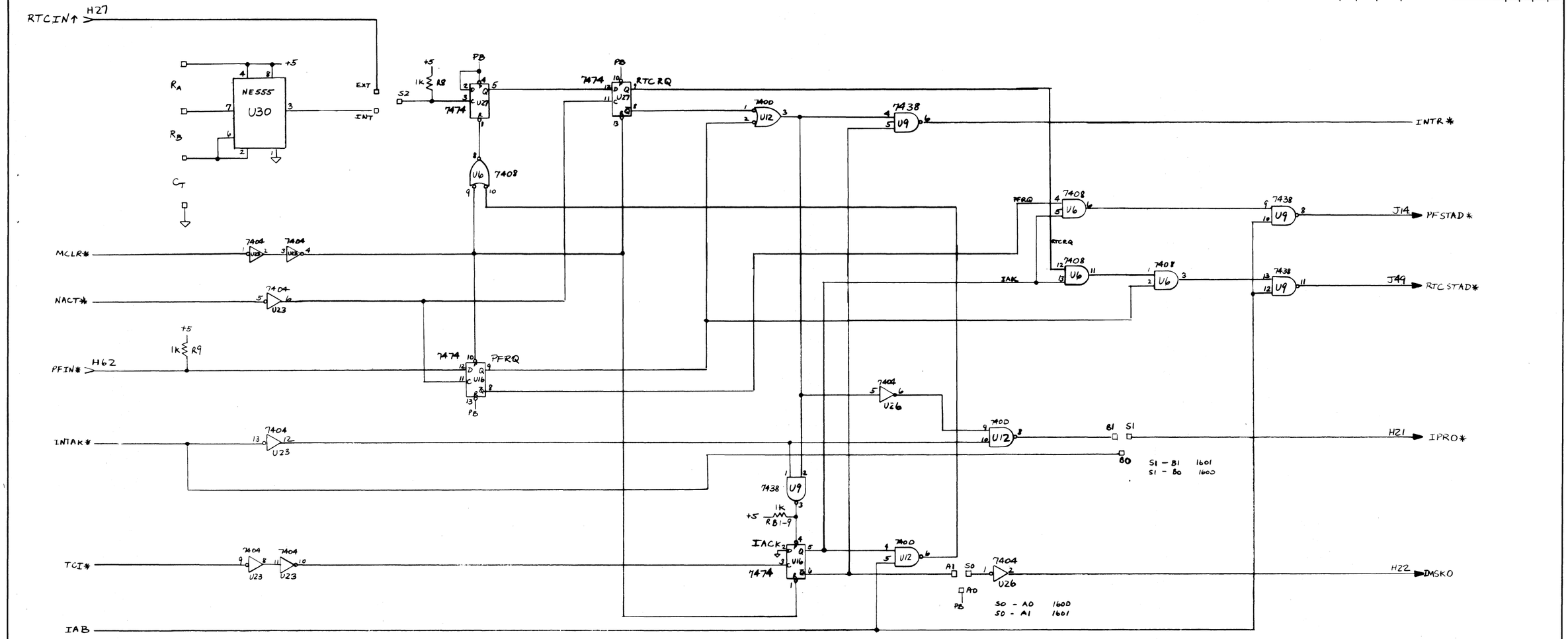
SWAP	0 0 0 0	SWAP 8-bit bytes of Register RR.
	0 0 0 1	SWAP bytes twice
SLL	0 0 1 0	Shift Logical Left (1 bit)
	0 0 1 1	(2 bits)
RLC	0 1 0 0	Rotate Left thru Carry (and Overflow)
	0 1 0 1	
SLLC	0 1 1 0	Shift Logical Left thru Carry (and Overflow)
	0 1 1 1	
SLR	1 0 0 0	Shift Logical Right (1 bit)
	1 0 0 1	(2 bits)
SAR	1 0 1 0	Shift Arithmetic Right (1 bit)
	1 0 1 1	(2 bits)
RRC	1 1 0 0	Rotate Right thru Carry (and Overflow)
	1 1 0 1	
SARC	1 1 1 0	Shift Arithmetic Right thru Carry (and Overflow)
	1 1 1 1	

APPENDIX B

CARD SCHEMATICS


REVISIONS
CP 1600 U4
74150
U14
U10
U7
U3
U1
8T26
ADDRESS REGISTER
74175
7438
9438
7437
74LS00
74H10
DM74LS5
7404
74195
U19
MM0026CN
U8
MC1600/MC1601
GENERAL INSTRUMENT CORPORATION
HICKSVILLE N.Y.
PARTS LIST
R. Lass
4/16/75
DS-MC-002
SHEET 1 OF 2

REVISIONS						
SYM.	DATE	E.C.N.	DESCRIPTION	BY	CH.	PROC.	APP.



SPARES

NOTE:
C12 – C19 are .01uf

1400	1401	REFERENCE	TYPE	+5	GND	+12
	1	U12	7400	14	7	
1		U15	74LS00	14	7	
2	1	U26, U2, U23	7404	14	7	
	1	U6	7408	14	7	
1		U11	7410	14	7	
5	1	U13, U17, U20, U24, U5, U9	7438	14	7	
	2	U16, U27	7474	14	7	
1		U28	DM74145	16	8	
1		U14	74150	24	12	
4		U18, U21, U25, U29	74175	16	8	
1		U19	74195	16	8	
4		U1, U3, U7, U10	8T26	16	8	
1		U8	MH0026CN		3	6
1		U4	CP1600	34	39	36
1		RB1	898-1-1K	16		
	1	U30	NE555	4, 8	1	
1		U22	7437	14	7	
24	7					

ITEM	NAME	REQ.	MATERIAL	DESCRIPTION	DWG. NO.	REMARKS

PARTS LIST

UNLESS OTHERWISE SPECIFIED
1. ALL DIMENSIONS IN INCHES.
2. DO NOT SCALE PRINTS.
3. REMOVE ALL BURRS.
4. BREAK ALL CORNERS.
5. CORNER RADII
6. LIMITS ON DIMENSIONS —
FRACTIONS:
DECIMAL:
ANGULAR:
7. SURFACE FINISH

GENERAL INSTRUMENT CORPORATION
HICKSVILLE N.Y.

TITLE: MC1600/MC1601

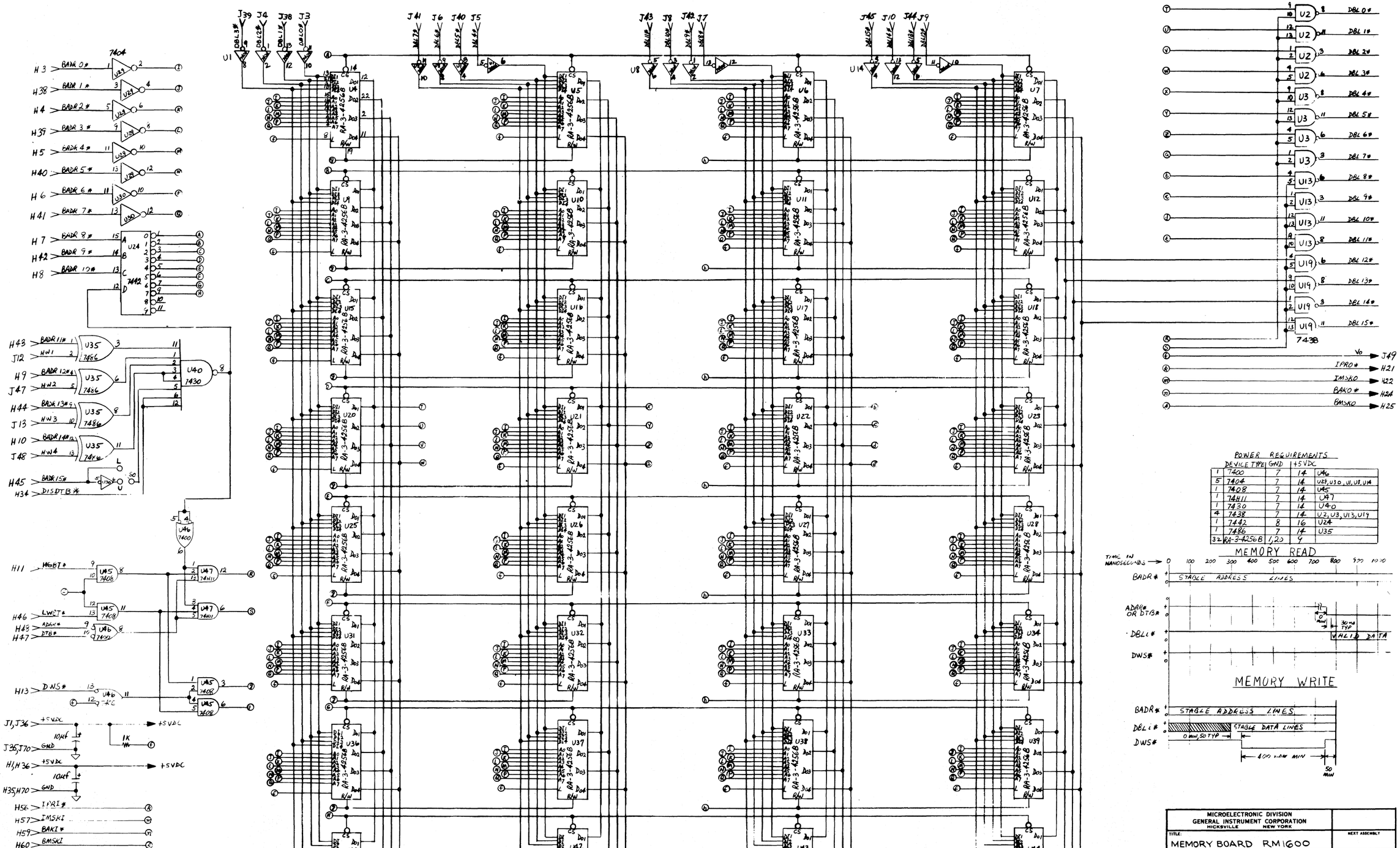
NEXT ASSEMBLY

SCALE | SUPERSEDED BY | SUPERSEDES

	DRAWN	CHECKED	PROC. APP.	ENG. APP.
BY	R. ROSS			M. LAMELL
DATE	4/16/75			

SHEET 2 OF 2

DS-MC-002



POWER REQUIREMENTS

	DEVICE TYPE	GND	+5VDC	
1	7400	7	14	U46
5	7404	7	14	U29, U30, U1, U8, U14
1	7408	7	14	U45
1	74H11	7	14	U47
1	7430	7	14	U40
4	7438	7	14	U2, U3, U13, U19
1	7442	8	16	U24
1	7486	7	14	U35
32	RA-3-4256B	1,20	4	

MEMORY READ

MEMORY WRITE

MICROELECTRONIC DIVISION
GENERAL INSTRUMENT CORPORATION
HICKSVILLE NEW YORK

TITLE: MEMORY BOARD RM1600

NEXT ASSEMBLY

SCALE —

SUPERSEDED BY

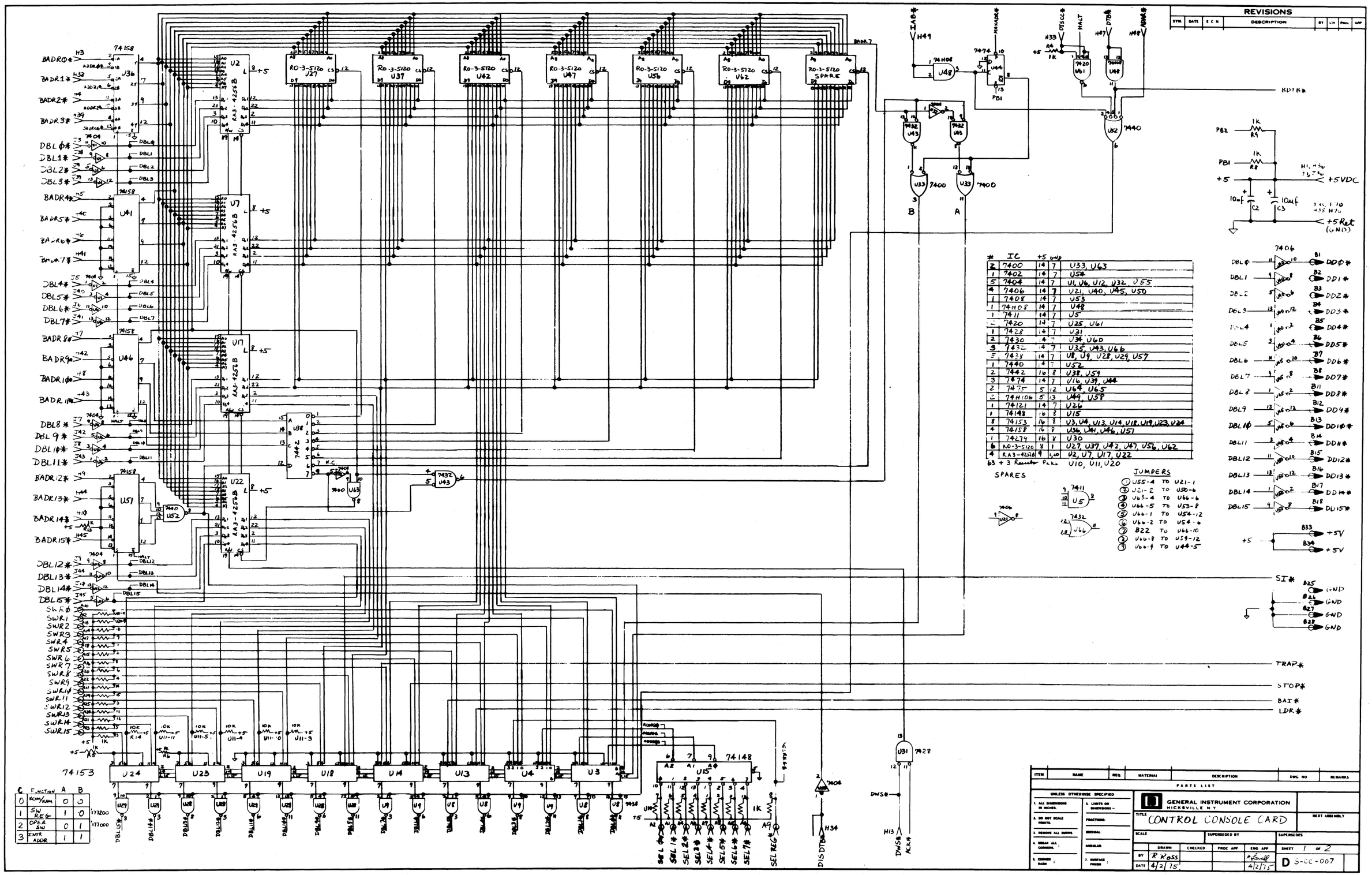
SUPERSEDES

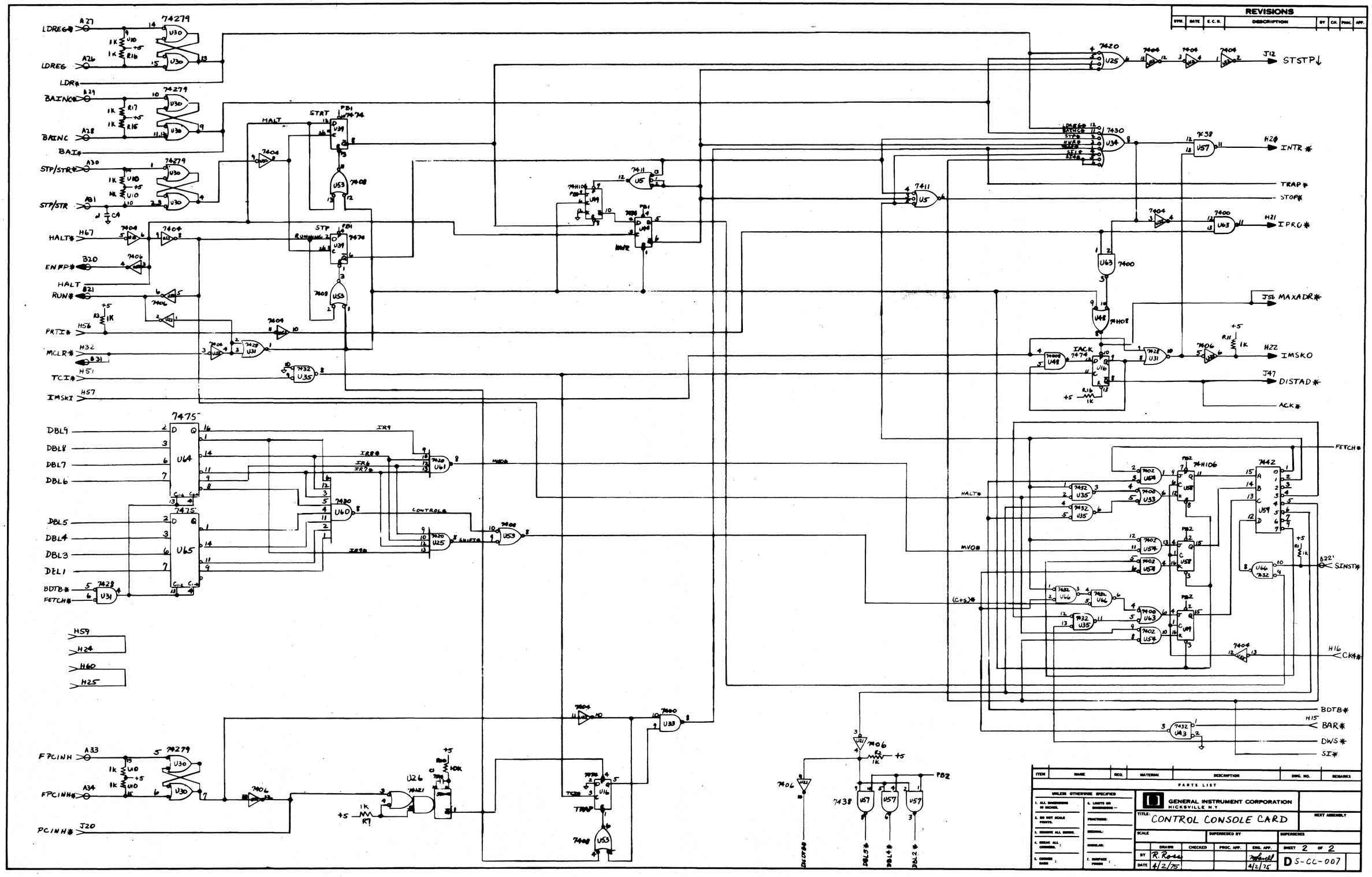
	DRAWN	CHECKED	PROC. APP.	ENG. APP.
BY				
DATE	12/18/79			

SHEET OF

D S-RM-012

S-RM-012

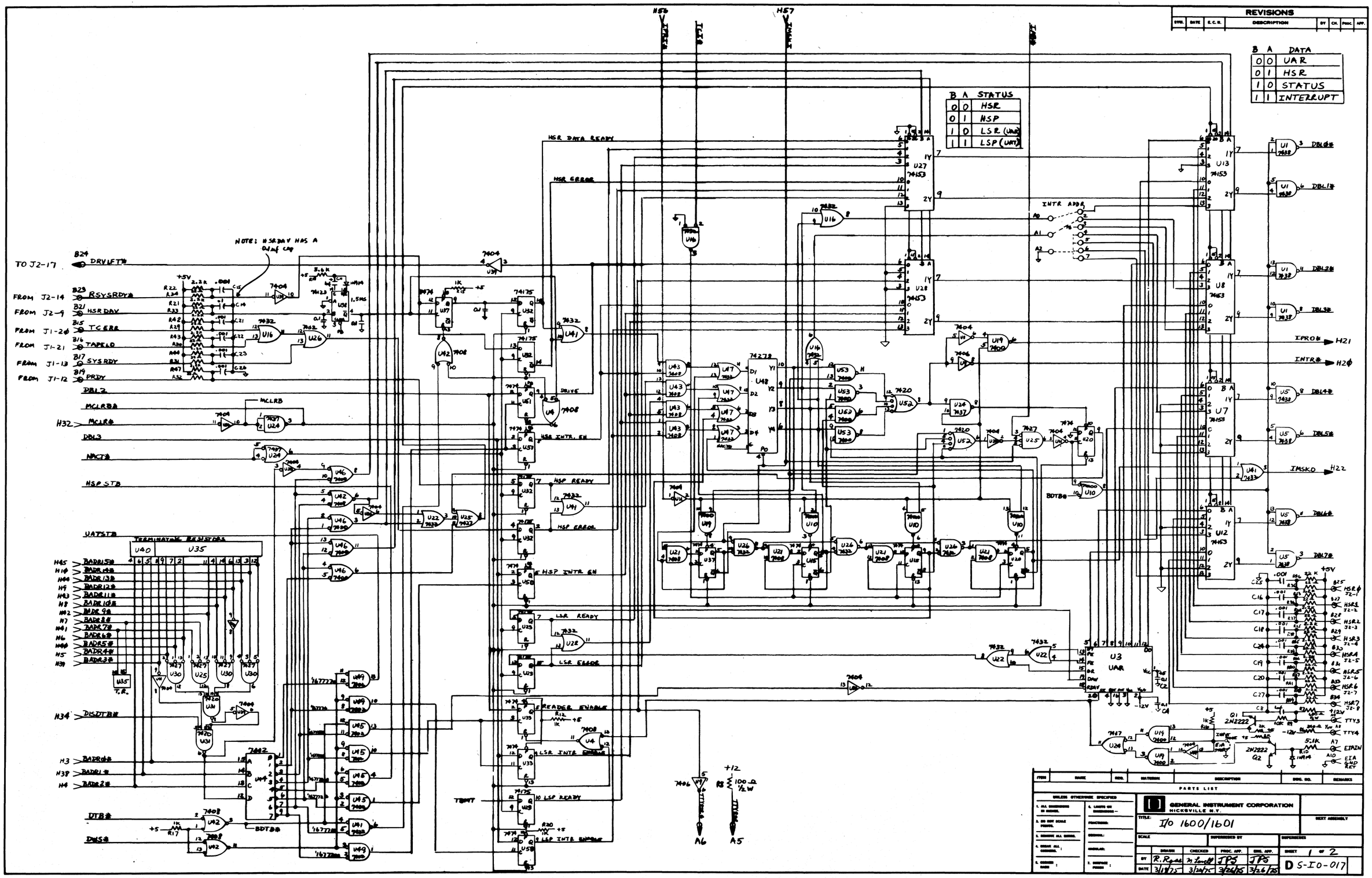

REVISIONS
SYM DATE ECN DESCRIPTION BY CH PROJ APP
BADR0*
BADR1*
BADR2*
BADR3*
BADR4*
BADR5*
BADR6*
BADR7*
BADR8*
BADR9*
BADR10*
BADR11*
BADR12*
BADR13*
BADR14*
BADR15*
DBL0*
DBL1*
DBL2*
DBL3*
DBL4*
DBL5*
DBL6*
DBL7*
DBL8*
DBL9*
DBL10*
DBL11*
DBL12*
DBL13*
DBL14*
DBL15*
SWR0
SWR1
SWR2
SWR3
SWR4
SWR5
SWR6
SWR7
SWR8
SWR9
SWR10
SWR11
SWR12
SWR13
SWR14
SWR15
RO-3-5120
RA3-4256B
74158
74153
74148
7400
7402
7404
7406
7408
7410
7411
7420
7428
7430
7432
7440
7442
7474
74H08
74H10
74H21
74H48
74H53
74H58
74H74
74H75
74H106
74H148
74H158
74H174
74H175
74H195
74H153
74H158
74H174
RO-3-5120
RA3-4256B
SPARE
BDTB*
PB2
PB1
+5VDC
+5Ret (GND)
DD0*
DD1*
DD2*
DD3*
DD4*
DD5*
DD6*
DD7*
DD8*
DD9*
DD10*
DD11*
DD12*
DD13*
DD14*
DD15*
+5V
GND
SI*
TRAP*
STOP*
BAI*
LDR*
SEL 0*
SEL 1*
SEL 2*
SEL 3*
SEL 4*
SEL 5*
SEL 6*
SEL 7*
SEL REG*
DISDTB*
DWS*
ACK*
H34
H13
H49
H33
H47
H48
HALT
DISCC*
DTB*
MANADR*
IAB*
PB1
B
A
C FUNCTION A B
0 ROM/RAM 0 0 177200
1 SW REG 1 0
2 OPER SW 0 1 177000
3 INTR ADDR 1 1
IC +5 GND
2 7400 14 7 U33, U63
1 7402 14 7 U58
5 7404 14 7 U1, U6, U12, U32, U55
4 7406 14 7 U21, U40, U45, U50
1 7408 14 7 U53
1 74H08 14 7 U48
1 7411 14 7 U5
2 7420 14 7 U25, U61
1 7428 14 7 U31
2 7430 14 7 U34, U60
3 7432 14 7 U35, U43, U66
5 7438 14 7 U8, U9, U28, U29, U57
1 7440 14 7 U52
2 7442 16 8 U38, U54
3 7474 14 7 U16, U39, U44
2 7475 5 12 U64, U65
2 74H106 5 13 U49, U59
1 74121 14 7 U26
1 74148 16 8 U15
8 74153 16 8 U3, U4, U13, U14, U18, U19, U23, U24
4 74158 16 8 U36, U41, U46, U51
1 74274 16 8 U30
6 RO-3-5120 8 1 U27, U37, U42, U47, U56, U62
4 RA3-4256B 9 1,10 U2, U7, U17, U22
63 + 3 Resistor Paks U10, U11, U20
SPARES
JUMPERS
1 U55-4 TO U21-1
2 U21-2 TO U50-6
3 U63-4 TO U46-6
4 U66-5 TO U53-8
5 U66-1 TO U54-12
6 U66-2 TO U54-6
7 B22 TO U66-10
8 U66-8 TO U59-12
9 U66-9 TO U44-5
ITEM NAME REQ. MATERIAL DESCRIPTION DWG NO REMARKS
PARTS LIST
UNLESS OTHERWISE SPECIFIED
1. ALL DIMENSIONS IN INCHES.
2. DO NOT SCALE PRINTS.
3. REMOVE ALL BURRS.
4. BREAK ALL CORNERS.
5. CORNER RADII
6. LIMITS ON DIMENSIONS —
FRACTIONS.
DECIMAL.
ANGULAR.
7. SURFACE FINISH
GENERAL INSTRUMENT CORPORATION
HICKSVILLE N.Y.
TITLE CONTROL CONSOLE CARD
NEXT ASSEMBLY
SCALE
SUPERSEDED BY
SUPERSEDES
DRAWN CHECKED PROC APP ENG APP
BY R ROSS
DATE 4/2/75
4/2/75
SHEET 1 OF 2
D 5-CC-007

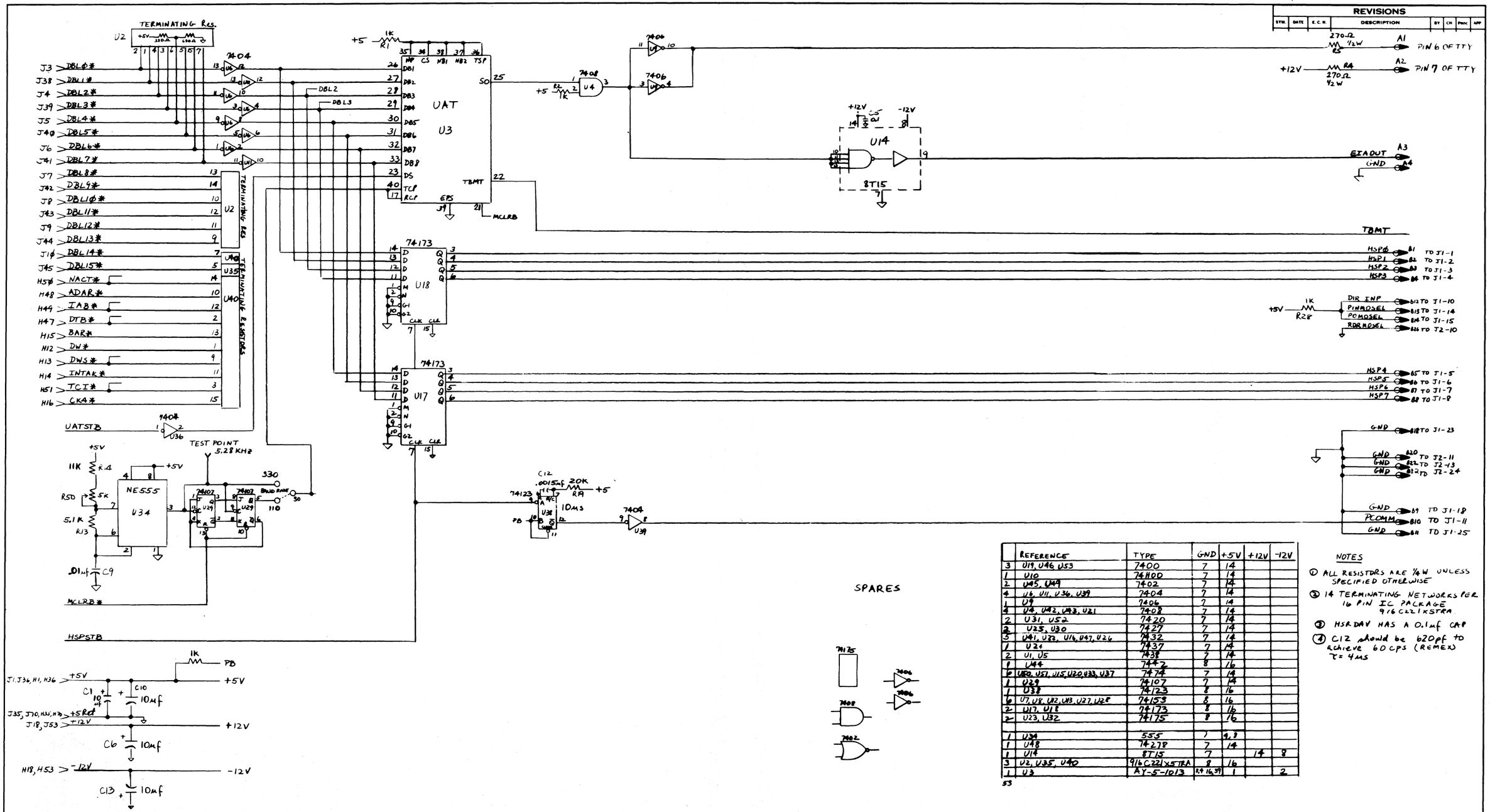

REVISIONS
SYM. DATE E.C.N. DESCRIPTION BY CK. PROC. APP.
LDREG#
LDREG
LDR#
BAINC#
BAINC
BAI#
STP/STR#
STP/STR
HALT#
ENFP#
HALT
RUN#
PRTI#
MCLR#
TCI#
IMSKI
DBL9
DBL8
DBL7
DBL6
DBL5
DBL4
DBL3
DBL1
BDTB#
FETCH#
FPCINH
FPCINH#
PCINH#
74279
7475
7404
7408
7474
74H106
7411
7430
7420
7400
7428
7429
7432
7442
7438
7406
74H08
7402
STRT
STP
HALT
IACK
IMSKO
DISTAD#
ACK#
MAXADR#
STSTP
INTR#
TRAP#
STOP#
IPRO#
FETCH#
SINST#
CK4#
BDTB#
BAR#
DWS#
SI#
CONTROL
SHIFTA
MVOB
MVOA
(C+S)#
DBL5#
DBL4#
DBL2#
DISTAB
PCINH#
GENERAL INSTRUMENT CORPORATION
HICKSVILLE N.Y.
TITLE: CONTROL CONSOLE CARD
NEXT ASSEMBLY
SCALE
SUPERSEDED BY
SUPERSEDES
SHEET 2 OF 2
DRAWN BY R. Rosa 4/2/75
CHECKED
PROC. APP.
ENG. APP. 4/2/75
D S-CC-007
PARTS LIST
ITEM NAME REQ. MATERIAL DESCRIPTION DWG. NO. REMARKS
UNLESS OTHERWISE SPECIFIED

REVISIONS						
SYM.	DATE	E.C.N.	DESCRIPTION	BY	CH.	PROC.	APP.

B	A	DATA
0	0	UAR
0	1	HSR
1	0	STATUS
1	1	INTERRUPT

B	A	STATUS
0	0	HSR
0	1	HSP
1	0	LSR (UAR)
1	1	LSP (UAT)





	REFERENCE	TYPE	GND	+5V	+12V	-12V
3	U19, U46, U53	7400	7	14		
1	U10	74H00	7	14		
2	U45, U49	7402	7	14		
4	U6, U11, U36, U39	7404	7	14		
1	U9	7406	7	14		
4	U4, U42, U43, U21	7408	7	14		
2	U31, U52	7420	7	14		
2	U25, U30	7427	7	14		
5	U41, U22, U16, U47, U26	7432	7	14		
1	U24	7437	7	14		
2	U1, U5	7438	7	14		
1	U44	7442	8	16		
10	U50, U51, U15, U20, U33, U37	7474	7	14		
1	U29	74107	7	14		
1	U38	74123	8	16		
10	U7, U8, U12, U13, U27, U28	74153	8	16		
2	U17, U18	74173	8	16		
2	U23, U32	74175	8	16		
1	U34	555	7	4,8		
1	U48	74278	7	14		
1	U14	8T15	7		14	8
3	U2, U35, U40	9316C221X5TRA	8	16		
1	U3	AY-5-1013	8,9,16,39	1		2
53						

NOTES

1. ALL RESISTORS ARE 1/4 W UNLESS SPECIFIED OTHERWISE
2. 14 TERMINATING NETWORKS PER 16 PIN IC PACKAGE 9316C221KSTRA
3. HSRDAV HAS A 0.1uf CAP
4. C12 should be 620pf to achieve 60 CPS (REMEN) τ = 4ms

ITEM	NAME	REQ.	MATERIAL	DESCRIPTION	DWG. NO.	REMARKS

PARTS LIST

UNLESS OTHERWISE SPECIFIED
1. ALL DIMENSIONS IN INCHES.
2. DO NOT SCALE PRINTS.
3. REMOVE ALL BURRS.
4. BREAK ALL CORNERS.
5. CORNER RADII
6. LIMITS ON DIMENSIONS — FRACTIONAL: DECIMAL: ANGULAR:
7. SURFACE FINISH

GENERAL INSTRUMENT CORPORATION
HICKSVILLE N.Y.

TITLE: I/O 1600/1601

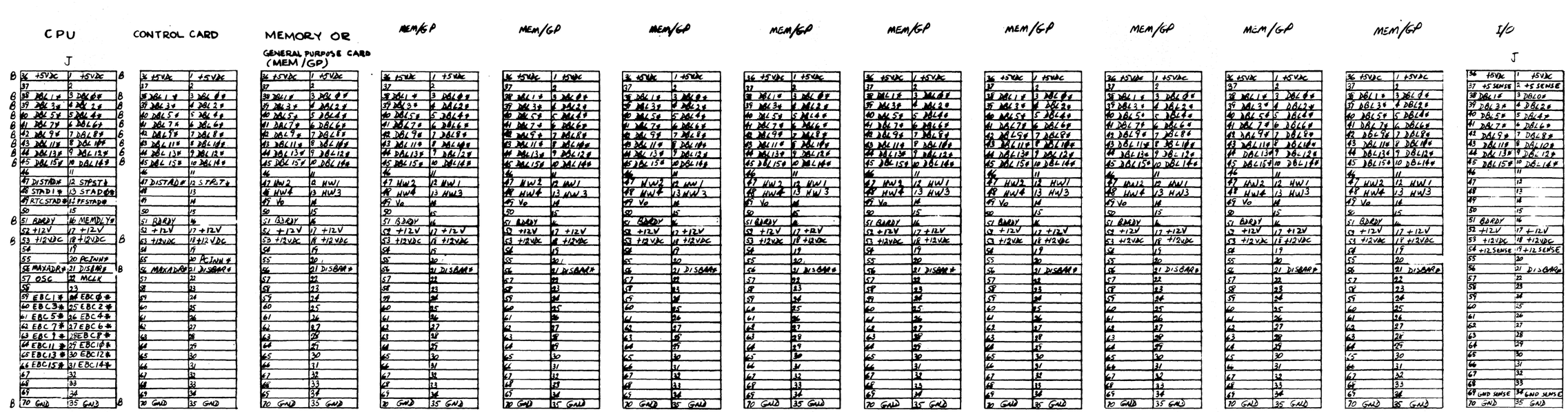
NEXT ASSEMBLY

SCALE | SUPERSEDED BY | SUPERSEDES

	DRAWN	CHECKED	PROC. APP.	ENG. APP.
BY	R Rosa	M Kowell	JTPS	JTPS
DATE	1/19/75	3/20/75	3/26/75	3/24/75

SHEET 2 OF 2

D S-IO-017



FACTORY JUMPERS: J14 (PFSTAD#) TIED TO J38 (DBL1#) 1601 ONLY
J49 (RTCSTAD#) TIED TO J4 (DBL2#) 1601 ONLY
J57 (OSC) TIED TO J22 (MCLK)
SLOT 2 H21 (IPRO#) TIED TO SLOT 13 H56 (IPRI#)
SLOT 2 H22 (IMSKO) TIED TO SLOT 13 H57 (IMSKI)

FOR SYSTEMS WITH FOUR RM1600 CARDS:
SLOT 3 HW1-4: GND
SLOT 4 HW1: V_0 HW2-4: GND
SLOT 5 HW2: V_0 HW1,3,4: GND
SLOT 6 HW3: V_0 HW1,2,4: GND

FOR SYSTEMS WITH ONE RM1601 CARD:
SLOT 3 HW1-3: GND

MICROELECTRONIC DIVISION
GENERAL INSTRUMENT CORPORATION
HICKSVILLE NEW YORK

TITLE: MICRO COMPUTER BACKPLANE

NEXT ASSEMBLY

SCALE —

SUPERSEDED BY

SUPERSEDES

	DRAWN	CHECKED	PROC. APP.	ENG. APP.
BY				
DATE 12/20/75				

SHEET OF

S-BP-026

GENERAL INSTRUMENT CORPORATION
MICROELECTRONICS

EASTERN AREA SALES HEADQUARTERS, 600 W. John St., Hicksville, N.Y. 11802, (516) 733-3107
CENTRAL AREA SALES HEADQUARTERS, 3101 West Pratt Blvd., Chicago, Ill. 60645, (312) 338-9200
WESTERN AREA SALES HEADQUARTERS, 7120 Hayvenhurst Ave., Van Nuys, Calif. 91406, (213) 781-0489

GENERAL INSTRUMENT CANADA LTD., 61 Industry St., Toronto 337, Ontario, Canada, Tel: (416) 763-4133
GENERAL INSTRUMENT MICROELECTRONICS LTD., 57/61 Mortimer St., London, W1N 7TD, England, Tel: 01-636-2022
GENERAL INSTRUMENT EUROPE S.P.A., Piazza Amendola 9, 20149 Milano, Italy, Tel: 469-7751
GENERAL INSTRUMENT FRANCE SA, 11-13 Rue Gandon, 75-Paris-13eme, France, Tel: 588-74-31
GENERAL INSTRUMENT DEUTSCHLAND GMBH, (Mos Produkigruppe) Nordendstrasse, 1A 8000 München 40 Tel: 28-40-31
GENERAL INSTRUMENT INTERNATIONAL CORP., Fukide Building, 17 Fukide-cho, Minato-ku, Tokyo 105, Japan, Tel: (03) 437-0281-5
GENERAL INSTRUMENT OF TAIWAN LTD., P.O. Box 22226 Taipei, Taiwan, Tel: 933861-3

Printed in U.S.A.