SIGNETICS TWIN
TESTWARE INSTRUMENT

SYSTEM REFERENCE

SIGNETICS TWIN
TESTWARE INSTRUMENT
SYSTEM REFERENCE

signetics
a subsidiary of U.S. Philips Corporation

Signetics Corporation
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700

ORDER NUMBER: TW09004000

PRICE: $5.00

TABLE OF CONTENTS

TABLE OF CONTENTS (con't)

TABLE OF CONTENTS (con't)

TABLE OF CONTENTS (con't)

CHAPTER 1

INTRODUCTION

1.0 THE TWIN SYSTEM

The TWIN (TestWare INstrument) system, shown in the block diagram of Figure 1-1, consists of interdependent subsystems, each contributing to the total task of implementing user microprocessor applications from initial concept to actual hardware operation. The system closely resembles a general-purpose minicomputer during the initial stages of product development, and allows source programs to be entered, edited and assembled into object programs. Object programs may be executed simply as programs, or as part of a user's product emulation. When programs have been run and debugged to the user's satisfaction, the TWIN system is capable of programming PROM devices for inclusion in the user's prototype hardware.

The Program Development Computer (PDC) is the principal subsystem of the TWIN. The PDC employs a dual-processor/dual memory architecture operating on a common bus structure. This Master/Slave architectural concept provides the following features:

1. Protected operating system memory and I/O.
2. Complete user access to Slave memory address space and I/O address space.
3. Operating system and Debug software are independent of user programs.
4. Support of future Slave processors by adding boards and supplying additional software.

The Program Development Computer is configured using plug-in modules, each dedicated to a specific task within the system. Each module interacts with the others by use of the common bus structure. The major buses include a data bus, an address bus, and a control bus. The PDC acts as the controller for the entire system, and performs the necessary functions of transferring data to and from system peripherals.

To facilitate system operation, a dual-drive floppy disk subsystem is provided in all TWIN configurations. This subsystem, with integral controller, stores user program files and allows retrieval of these files for operation by the

editor and assembler. The Signetics Disk Operating System (SDOS) software is loaded from the disk system into the development computer. SDOS provides complete control over all portions of the TWIN system.

A CRT console is the standard device for entering user programs. It is equipped with a full ASCII keyboard for data entry and a CRT display to allow the operator to view the results of his program manipulations during editing, assembly, and debug operations. The display and keyboard can be separated for user convenience. An interface is provided for an ASR-33 teletypewriter, which can be used in place of the CRT console. An optional line printer provides hard copy output.

Other system I/O devices can be added to the system to enhance its capabilities. Physical interface to the system bus is accomplished by use of the optional General-Purpose I/O card. Addition of a peripheral device requires addition of its software driver to the TWIN operating system (see Appendix D).

TWICE (TestWare In-Circuit Emulation) denotes hardware/software elements of the TWIN system that support integration and check-out of the user's prototype product. A number of different TWICE operation modes support product development, integration, and production. These are:

0. Using Slave memory and GP I/O, the TWIN system Slave CPU executes the user's "paper" program and allows him to determine whether or not the program does what he wants in terms of manipulating his data. The system console allows a display of the contents of memory, registers, and I/O ports to aid in locating bugs in the program. This is shown in Figure 1-2A.

1. Using the TWIN Slave memory for program storage and the TWICE interface cable to connect the user's prototype I/O, the system allows the Slave CPU to take the place of the user's CPU and run and debug programs. This is shown in Figure 1-2B.

2. Using the Slave CPU and user memory, I/O and other hardware, the system allows user programs to be run and debugged in the actual user environment. This is shown in Figure 1-2C.

The TWIN PROM programming capability lets the user program his memory with the object programs created by the system in the earlier phase of development, thus simulating 99% of the final hardware. The Debug software package lets the user trace program execution, examining contents of RAM at selected locations and checking CPU status and I/O operations. Thus the complete range of user needs is met, beginning with a user program on paper and ending with final execution in hardware.

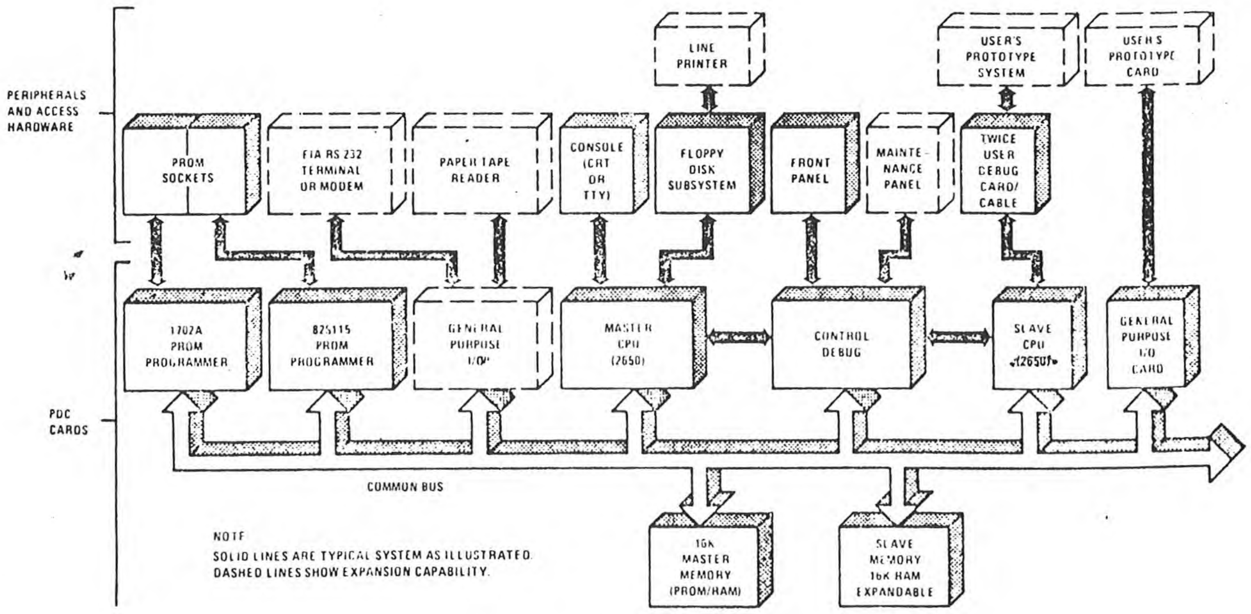


FIGURE 1-1
THE TWIN SYSTEM

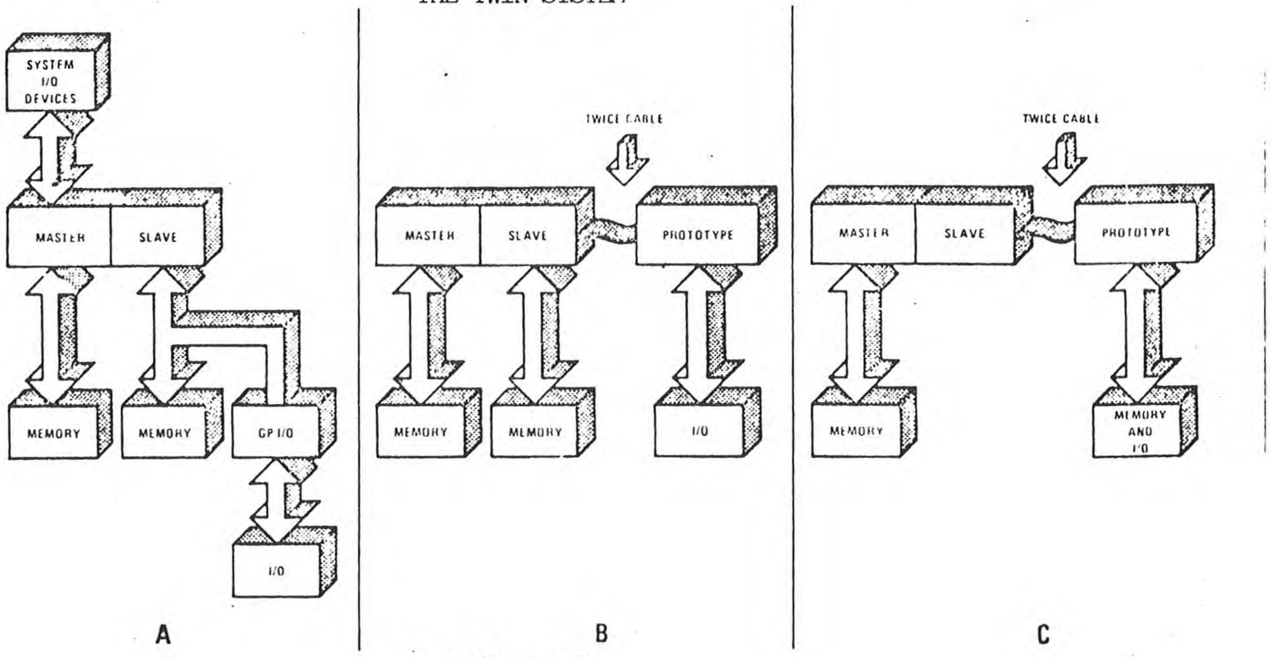


FIGURE 1-2
TWICE OPERATING MODES

1.1 TWIN DEVELOPMENT SOFTWARE

System software provided with the Prototype Development System includes the Signetics Disk Operating System (SDOS), Text Editor, Assembler, and Debug Package.

SDOS

SDOS relieves the user of the necessity of understanding detailed internal operation of the TWIN. It provides complete control over operation of all portions of the Prototype Development System. All functions relating to file handling, loading and execution are monitored and controlled by SDOS, including in-circuit emulation and PROM programming functions. SDOS resides in a dedicated memory consisting of a 256 byte PROM and 16K RAM running under the Master CPU.

The SDOS software allows the user to create, edit, and assemble files; obtain object and listing outputs; load and execute programs; and, through the debug system, check out programs in an efficient manner.

SDOS provides a powerful command file capability that enables the user to create customized operating system commands.

SDOS controls the multi-drive floppy disk subsystem, a line printer, and the CRT console, which may be an ASR-33 TTY or an RS-232 compatible device. Software drivers are provided within SDOS for these I/O devices. The SDOS software provides a flexible input/output system that enables the user to dynamically assign any logical channel to any physical device or file within the system. In this way, system I/O devices may be dynamically assigned using SDOS commands either from the console or from within a user's program. Thus, the user may write his own driver for other peripheral devices and link them into the SDOS system by use of the optional General-Purpose I/O Card.

Text Editor

The Text Editor is a comprehensive software package that allows the user to enter and modify text files. The Text Editor is line oriented and accepts inputs from the terminal or a disk file, performs modifications in a work space, and outputs the revised text to a disk file.

Assembler

The Prototype Development System Resident Assembler translates symbolic assembly language instructions into appropriate machine language code.

The Assembler generates absolute object code. This code is in hexadecimal format and may be loaded into the system for direct execution or may be converted by an SDOS command to SMS format for PROM or ROM programming.

Debug

The Debug system is a software program which provides the user with real-time program debug capabilities within both a software and hardware environment. It uses special hardware features built into the program development system to control execution of the user's program. User programs operating under the Debug system have dynamic program trace, breakpoint capabilities, and memory modification capabilities. Status reporting on the memory, the program, and the processor is also provided.

1.2 PURPOSE OF THIS MANUAL

This manual is intended to acquaint an experienced logic designer with the TWIN system, both in overall concept and architecture, and in detail. The descriptions of functions and the detailed descriptions of each board in the system should be sufficient to allow troubleshooting to the board level in case of failure. It is assumed that the reader has a knowledge of microcomputer development systems and a knowledge of the 2650 microprocessor. For a detailed description of system operation, refer to the TWIN Operator's Guide.

1.3 MANUAL CONTENTS

Chapter 2 of this manual provides a description of the system in general terms, and describes its architecture. The rest of the manual is devoted to describing each board used in the system, with functional descriptions and a theory of operation at the block diagram level. System peripherals are described briefly in Chapter 2, but detailed descriptions and operating theory are not provided. The reader is referred to the manuals for each peripheral device for operation and maintenance.

CHAPTER 2

SYSTEM DESCRIPTION

2.0 SYSTEM ARCHITECTURE

The TWIN Program Development Computer (PDC) is based on a dual processor concept, as shown in Figure 2-1. The Master CPU runs the SDOS operating system and interfaces to all system I/O devices, while the Slave CPU runs user programs as well as the Text Editor and Assembler. The Master CPU has completely separate memory and I/O, protected from user programs. This provides a relatively crash-proof operating system and gives the user almost total use of the address space in Slave memory.*

The Slave CPU can utilize the system I/O structure through "supervisor calls" (SVCs) to the Master. SVCs are described in Appendix B of this manual.

Both Master and Slave CPU's are currently 2650 microprocessor devices. However, the dual processor architecture permits different Slave CPU's to be supported by the system without requiring a complete re-write of the operating system. Only those portions of the system software unique to the Slave CPU being supported (the Assembler and portions of the Debugger) must be updated. Hardware provisions within the TWIN permit support of Slave CPU's of up to 16-bit word length and up to 64K address space.

Interaction of Master and Slave CPU's is controlled by the Debug module under Master CPU control. Both CPU's share the basic bus structure, with only one processor active at any one time. Control is transferred to the Slave by a Master halt. Control is returned to the Master by any system interrupt or a Debug interrupt such as a breakpoint.

The bus structure is essentially universal for all board locations in the card cage, with minor exceptions. The Debug board splits the bus into Master and Slave halves, with a separate interrupt structure for each half. The PROM Programmer boards have dedicated locations because of the special AC voltages required for programmer operation. Memory or I/O boards may be placed in any open position (see Figure 2-3).

* A 100 byte segment of Slave memory is required for Master/Slave interface when running the Slave CPU under Debug control.

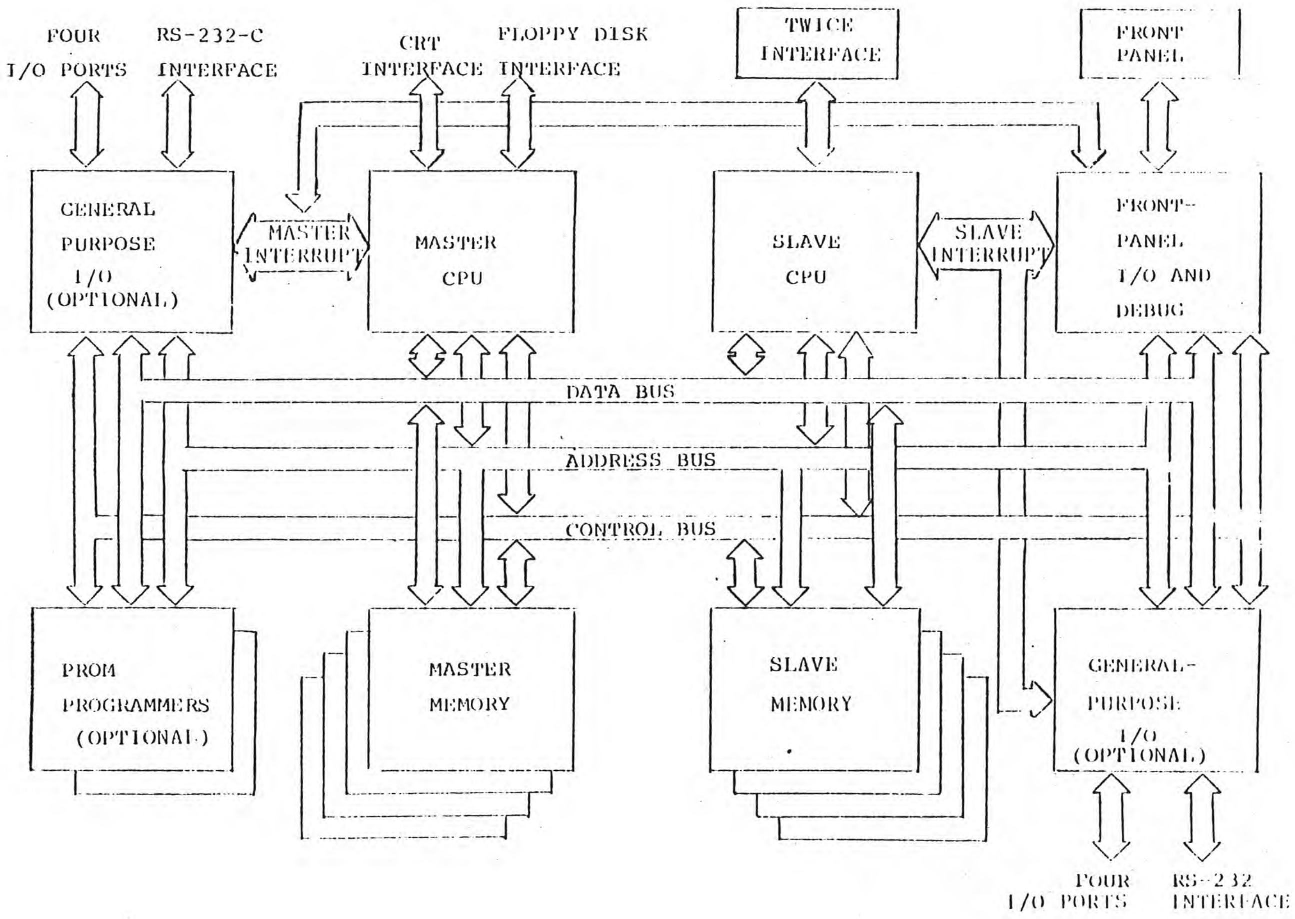


FIGURE 2-1

PROGRAM DEVELOPMENT COMPUTER BLOCK DIAGRAM

The TWICE cable assembly provides in-circuit emulation of user hardware. The cable connects to the Slave CPU on one end and essentially replaces the CPU in user hardware on the other end. This gives total debugging capability in an actual hardware environment.

Basic system I/O consists of the floppy disk interface and the CRT terminal or TTY interface, both handled by the Master CPU. General-Purpose I/O modules provide a limited RS-232-C interface and parallel I/O ports to interface user peripherals. The GPI/O card can also be used in the Master side to add additional system I/O devices. Addition of a system peripheral requires addition of its software driver to the operating system. This procedure is described in Appendix D of this manual.

2.1 MEMORY ORGANIZATION

Master Memory consists of 16K bytes of RAM, with the first 256 bytes overridden by a PROM which contains the system bootstrap program. When the system is initialized, the Master accesses the bootstrap program to load the operating system from the system diskette into Master Memory.

Slave Memory contains 16K bytes of RAM in the standard system, expandable to 64K bytes maximum. The Slave has access to Slave Memory only, while the Master has access to both memories. This is accomplished by assigning, under program control, a 16K byte portion of Slave Memory to the Master's 16K-32K address space. This is shown in Figure 2-2. The ability of the Master to "bank switch" Slave Memory into its address space allows communication for I/O service requests and Debug trace operation.

2.2 BUS STRUCTURE

The PDC bus structure is shown in Figure 2-3. The bus is split into Master and Slave sections by the Debug and I/O Logic board. The bus is essentially universal in that data, address, and control lines are paralleled to all boards. The exceptions to this structure are Slave Debug control lines, Master Debug control lines, and special PROM power lines. The bus structure allows complete freedom of memory placement as well as freedom within each half of the bus for CPU or I/O board placement. I/O and memory address assignment is located on the individual boards.

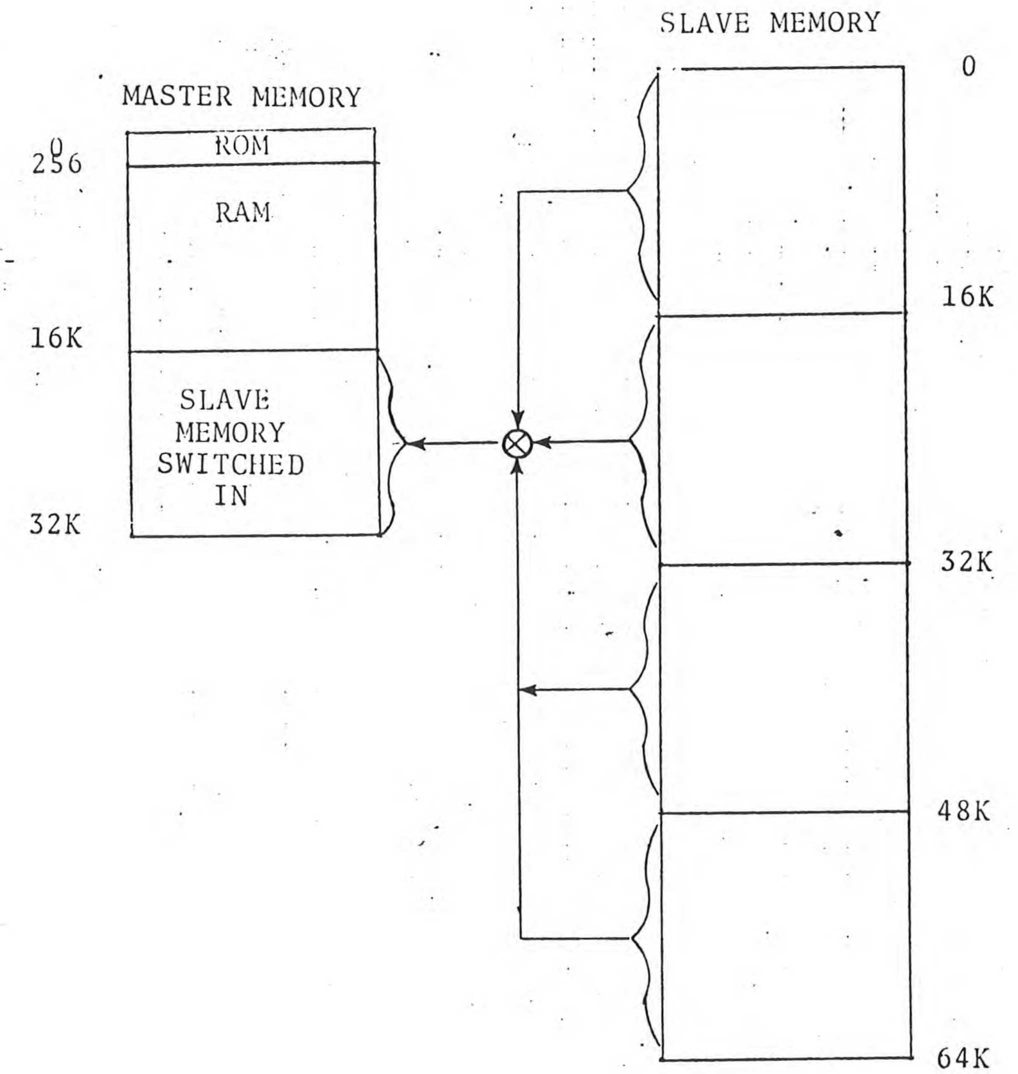


FIGURE 2-2

MEMORY ORGANIZATION

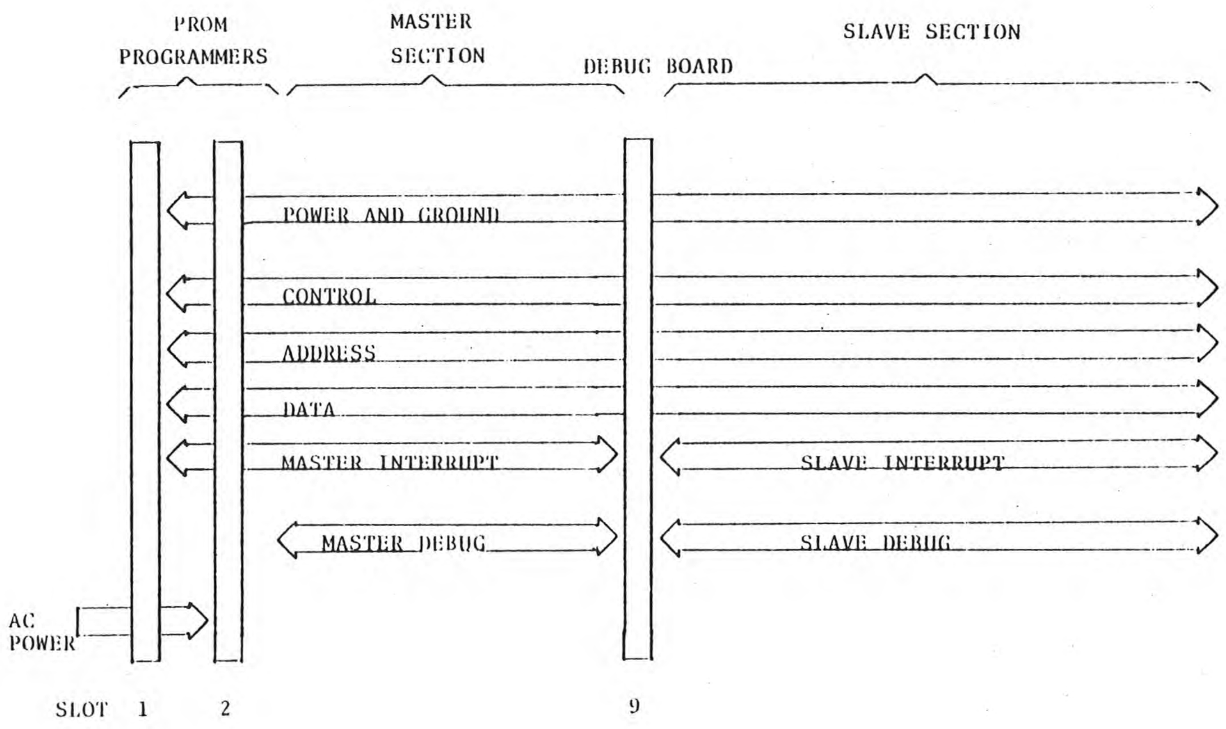


FIGURE 2-3

BUS STRUCTURE

The bus consists of 100 lines made up of the following signals:

COMMON BUS

16 address lines
16 data lines
21 control lines:
 9 memory and I/O control lines
 7 system control lines
 3 Debug lines
 2 sense/flag lines
20 power and ground lines

SLAVE SIDE

3 Debug lines
10 interrupt-related lines
8 spare lines

MASTER SIDE

4 Debug lines
17 interrupt-related lines

PROM PROGRAMMER

14 AC voltage lines

Four of the 20 power and ground pins provide an auxiliary power bus which can be connected via rear-panel terminals to an external power supply for user requirements not met by the standard system supplies. The AC voltages required by the PROM programmers extend to the end board positions only and mechanical keying prevents plugging any other boards into those positions. All board connectors are offset to prevent plugging boards in backwards.

2.3 DEBUG HARDWARE

Hardware is contained on both the Debug module and Slave CPU module to facilitate the following major system Debug features:

- Master/Slave interaction
- Breakpoints
- Forced Slave jumps

Master/Slave Interaction

All Debug hardware communication to the Master CPU is by way of interrupts. The Master relinquishes control to the Slave CPU by performing a Halt instruction. Control is regained by the Master when it receives an interrupt. Interrupts may be from system I/O or from the Debug hardware. The Debug module contains its own interrupt priority decoder and vector generator for Debug and supervisor call interrupts.

Breakpoints

Breakpoint registers are contained on the Debug module. The breakpoint addresses are loaded by the Master CPU under command from the user. When a breakpoint address is identical to the current Slave CPU address, the Slave CPU is interrupted before the next instruction fetch. Both the "last" instruction address and the "next" instruction adddress are stored. This allows the software to examine the Slave's program and interpret op-codes for the trace printout. It also allows the system to restore the Slave to its original address after executing a register dump routine.

Forced Slave Jumps

Forced Slave jumps are required for two reasons:

1) To execute the user's program starting at any command location

2) To allow Slave execution of register dump and register restore routines for trace.

The jump address is contained in a register in the Debug module and the forced jump logic is contained on the Slave CPU module.

TWICE Cable

The TWICE cable allows the user to exercise his hardware while still retaining the Debug features of the development system. The cable connects to the Slave CPU on one end and plugs in in place of the user's 2650 in his hardware on the other. The cable contains line drivers and receivers close to the user end to insure signal integrity and minimize user circuit loading.

The Slave CPU essentially acts as the user's 2650. There are two in-circuit modes of operation. In Mode 1, the user's I/O is exercised while still operating out of the TWIN Slave memory. Full Debug and trace operations are available. In Mode 2, both the user's I/O and memory are utilized, still allowing breakpoints, single step, and trace. Multiplexers on the Slave board switch between the cable interface and the computer bus for each mode. The cable can remain installed in the normal Slave CPU mode with no adverse effects.

2.4 I/O PROVISIONS

The CRT/TTY interface is a serial interface providing a limited EIA RS-232-C interface and a 20 mA TTY current loop interface. Baud rates are switch selectable for TTY (110) or CRT (300). Additional jumpers can modify the CRT Baud rate to 150, 300, 600, or 1200. A 25-pin standard EIA connector on the rear panel connects to the top edge of the Master CPU board.

The General Purpose I/O board allows both system I/O expansion and user I/O capability. The board contains a full RS-232-C interface, four sets of 8-bit parallel I/O ports, and eight interrupt flip-flops. This allows a hardware interface to any TTL-compatible peripheral. By inserting the card in the Slave side and setting a control switch, the card responds to Slave commands only and can operate the Slave interrupt structure.

2.5 PROGRAM DEVELOPMENT COMPUTER SUB-ASSEMBLIES

As shown in Figure 2-1, the development computer is divided into functional modules, with each module consisting of a separate printed circuit board. The computer is divided both functionally and physically into two sections: Master and Slave. The Master section includes the Master CPU and I/O board, Master Memory (consisting of four 4K memory boards), and PROM Programmer boards. Common to both Master and Slave is the Front-Panel I/O and Debug board. The Slave section includes the Slave CPU board and Slave Memory (up to eight 4K RAM boards). General-Purpose I/O boards may be included in the Master or Slave sections. All of the boards plug into the Mother board which carries power, address, data, and control buses.

2.5.1 MASTER CPU AND I/O BOARD

The Master CPU and I/O board function as the controller for the system. This board is shown in the shaded portion of Figure 2-4. The board contains clock circuitry for the 2650, which is the Master CPU, as well as for the rest of the system. The 2650 requires auxiliary logic which buffers the address and data lines to drive the system buses without loading the chip. I/O functions include I/O ports for use with the floppy disk system and the CRT terminal or TTY. Interrupt circuits are provided to interrupt the CPU during I/O operations and Debug operations. An interrupt timer, driven by the system clock, is provided to give selectable 10 ms or 100 ms interrupts. The timer can be enabled or disabled under program control.

The CPU board is the major controlling element in the system. It is the primary source of addresses for memory access and I/O operations; its control signals determine the direction of data flow on the data bus and enable the various I/O devices to receive data from or transmit data to the bus. The CPU also outputs data to and from the disk system in accordance with the disk operating system program. The following operations are performed by the CPU.

1) Address output to memory and I/O ports.

2) Data output to memory and I/O ports.

3) Data input from memory and I/O ports.

4) Control outputs to memory, I/O ports, disk, and TTY or terminal.

5) Operations on data input to the CPU (such as addition, logic operations, etc.)

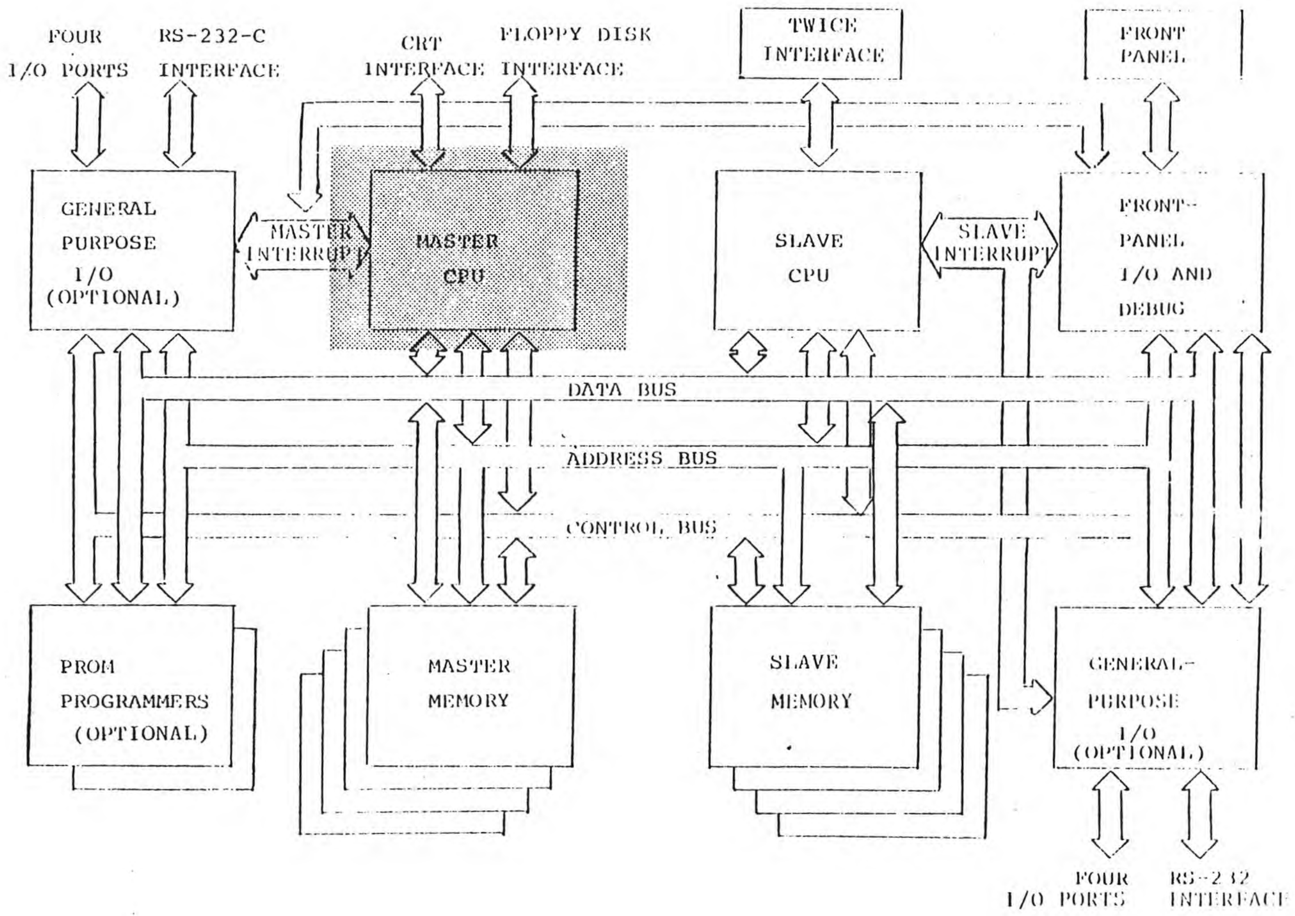


FIGURE 2-4

MASTER CPU AND I/O

2.5.2. MASTER MEMORY

Master Memory, shown in the shaded portion of Figure 2-5, consists of four 4K RAM boards. A base address for each board is selected by switches mounted on the board. A portion (256 bytes) of the Master Memory is implemented as PROM, which contains the system bootstrap program.

The Master Memory is divided into three parts:

* The bootstrap area consists of 256 bytes of PROM which contain a program to initialize the system upon system reset. The bootstrap program also loads the resident SDOS program from the disk.

* The resident SDOS area contains the portion of the operating system which is always required to be resident in the system.

* The overlay area contains programs which are brought in from the disk as required for execution of system commands.

The memory is under the control of the Master CPU, and acts as a result of commands received from the CPU. The following operations are performed.

1) Output to the data bus the contents of memory accessed by the CPU

2) Store in memory the contents of the data bus when specified by the CPU.

3) Return "operation acknowledge" signals when accessed by the CPU.

2.5.3 PROM PROGRAMMER BOARDS

There are two PROM programmer boards in the system, connected to the system bus and to the front panel PROM sockets. These boards are shown in the shaded portion of Figure 2-6. Each board, although differing in detail, performs the same function. On command from the system software, an object file is loaded from the disk into Slave memory. The data contained in the file is transferred, one word at a time, from the memory to registers on the board. Sequencers and timing circuits address the chip to be programmed, and pulse the power supplies on the board to enter the data into the PROM. When the programming sequence is complete, the programmer interrupts the CPU and the next word is programmed following the same sequence. Circuitry is included to read the data stored on the PROM and output it back to the system for error checking.

When PROM programming is required, the selected programmer board interacts with the Master CPU to control the flow of data and to flag errors. The following actions occur:

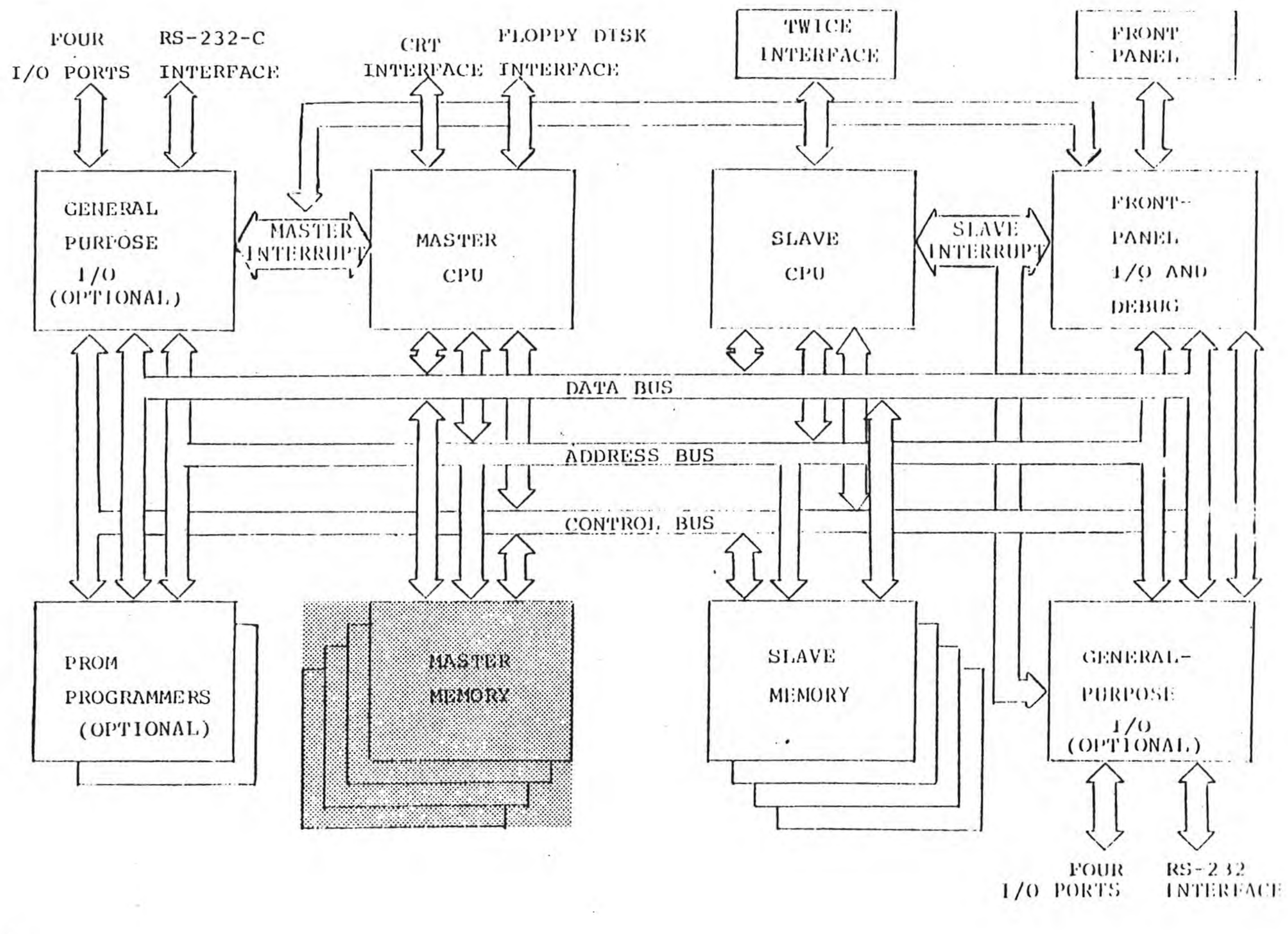


FIGURE 2-5

MASTER MEMORY

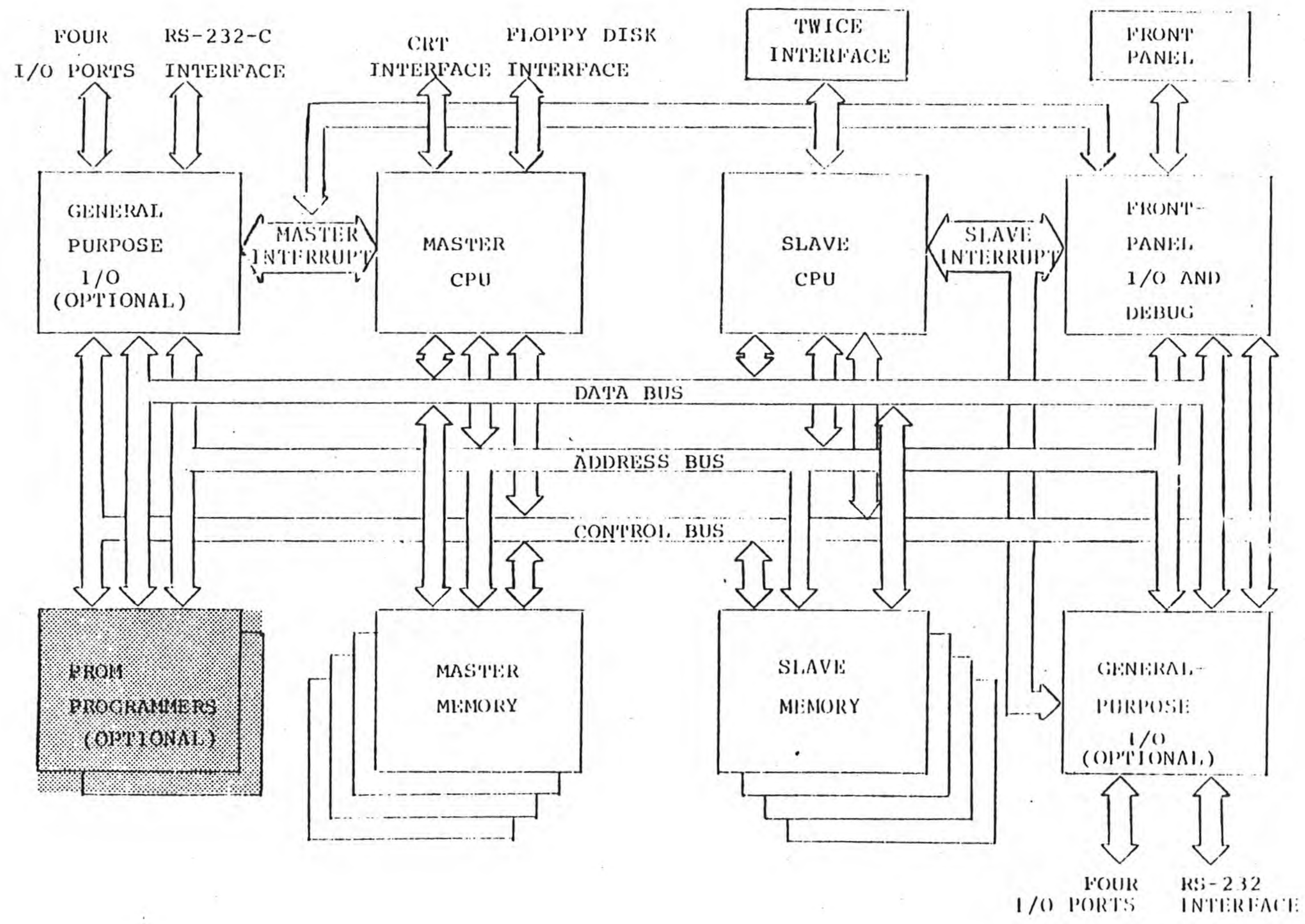


FIGURE 2-6

PROM PROGRAMMERS

1) The programmer accepts data and addresses from the CPU via the data and address buses, and stores each byte in registers for programming the device.

2) The programmer outputs a "busy" signal during the programming cycle, forcing the CPU to loop and look for a "done" bit in the programmer status register.

3) The operating system checks the data programmed against the data read from the PROM after the programming cycle.

4) The programmer outputs a "power fail" bit to the status register when any of the on-board power supplies have failed. A "PROM Power Switch" status bit is also set when the front panel PROM programming switch is not turned on.

Circuitry on the programmer cards also allows the contents of a previously programed PROM to be compared to data in the Slave memory. The contents of a PROM may also be loaded into the memory.

2.5.4 SLAVE CPU AND TWICE INTERFACE

Control of user programs and interface to user hardware is provided by the Slave CPU, shown in the shaded portions of Figure 2-7. The board contains a 2650 microprocessor and supporting logic to drive the system bus and the TWICE interface. Jump control logic is also provided to allow the Slave CPU to jump when commanded by the system. Interrupt logic is also contained on the board to allow interrupts as requested from the Slave interrupt bus (separate from the Master interrupt bus). Priority encoding is accomplished by encoding circuitry on the board. Interface to user hardware is done by control and driver circuits which select one of three Slave CPU modes:

0) The Slave uses the development computer memory and I/O,

1) The Slave uses external user I/O and development computer memory, and

2) The Slave uses external user memory and I/O.

In user mode 1 and 2, the user clock provides timing for the Slave CPU.

The Slave CPU board operates in two distinct modes: under program control from user programs (either in Slave memory or user memory) to control user hardware, and under control by the Front Panel I/O and Debug board. Essentially, the types of operations performed are the same as the Master CPU, except that the Slave CPU can access only Slave Memory or user memory and I/O devices. When the

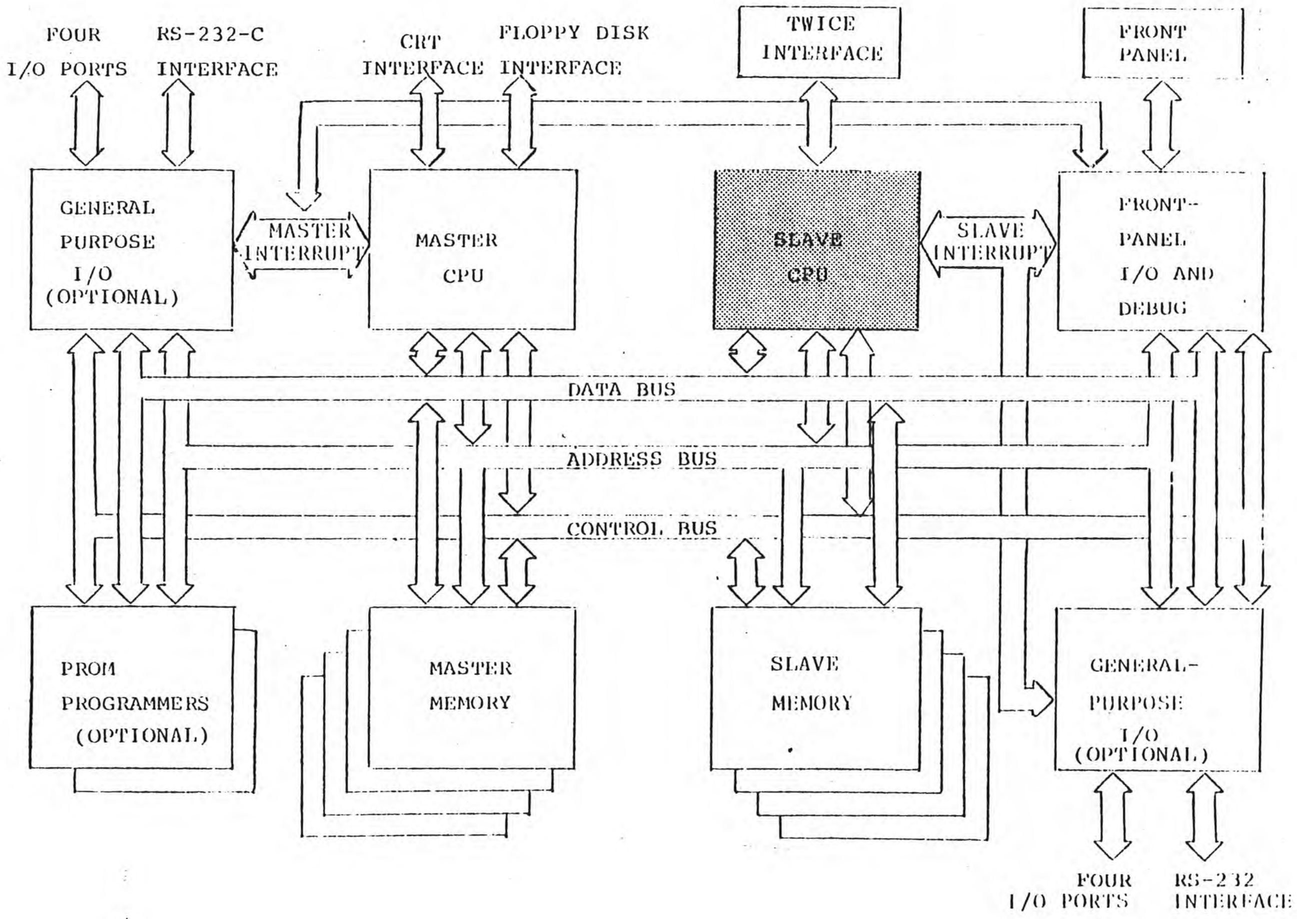


FIGURE 2-7

SLAVE CPU AND TWICE INTERFACE

Master CPU is running, the Slave is halted, waiting for control to be relinquished to continue executing instructions or to jump to an address indicated by the Debug circuitry.

Two system programs, the Assembler and the Editor, run on the Slave CPU.

2.5.5 SLAVE MEMORY

Slave Memory, shown in Figure 2-8, is similar to Master Memory, and consists of up to eight 4K RAM boards. PROM is not included. The Slave Memory holds the user program. It is also used to hold the Assembler and Editor, when these programs are run on the system.

2.5.6 GENERAL-PURPOSE I/O BOARD

Interface to external data devices is provided by the General-Purpose I/O board, shown in the shaded portions of Figure 2-9. The board can be used by either Master or Slave. This board consists of four pairs of input/output ports, selected by an extended read or write operation commanded by the CPU. Each port has a unique address, and can be connected to a variety of external parallel data devices through connectors on the top of the board. A serial interface for either RS-232-C devices or a TTY is provided together with control and status ports for data interchange protocol. A UART is located on the board to perform the necessary serial-to-parallel conversion. Separate drivers are provided for either the RS-232-C or TTY outputs, although only one device can be used at one time. Clock circuits, using the system clock as a reference, provide data rates of 110 Baud for the TTY and 110 to 1200 Baud for the RS-232-C interface.

The General-Purpose I/O board accepts input data from external sources and outputs data to external sources under program control. Essentially the interaction is similar to that of memory, with the exception of the capability to interrupt the CPU when data is ready to be read or ready to be transmitted. The following are part of the interactions of the board:

1) Accept addresses and data from the buses and write to one of the I/O ports, including the UART status and control ports.

2) Output data to the CPU when a read operation is performed.

3) Interrupt the CPU when data is available from the ports or when data is ready to be transmitted.

2.5.7 FRONT-PANEL I/O AND DEBUG LOGIC BOARD

The Front-Panel I/O and Debug Logic is perhaps the most complex board in the system in its interaction with the other system modules.

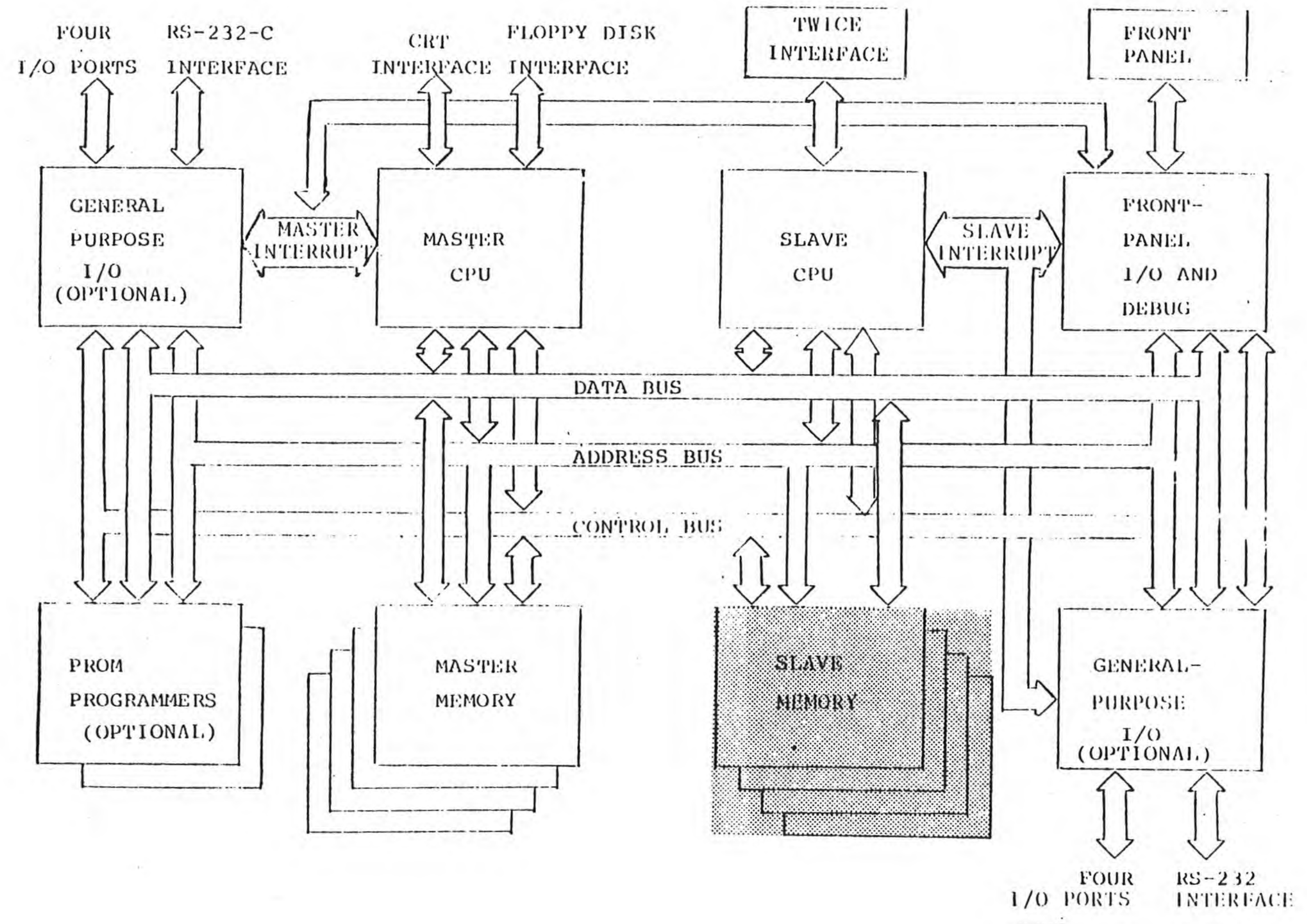


FIGURE 2-8

SLAVE MEMORY

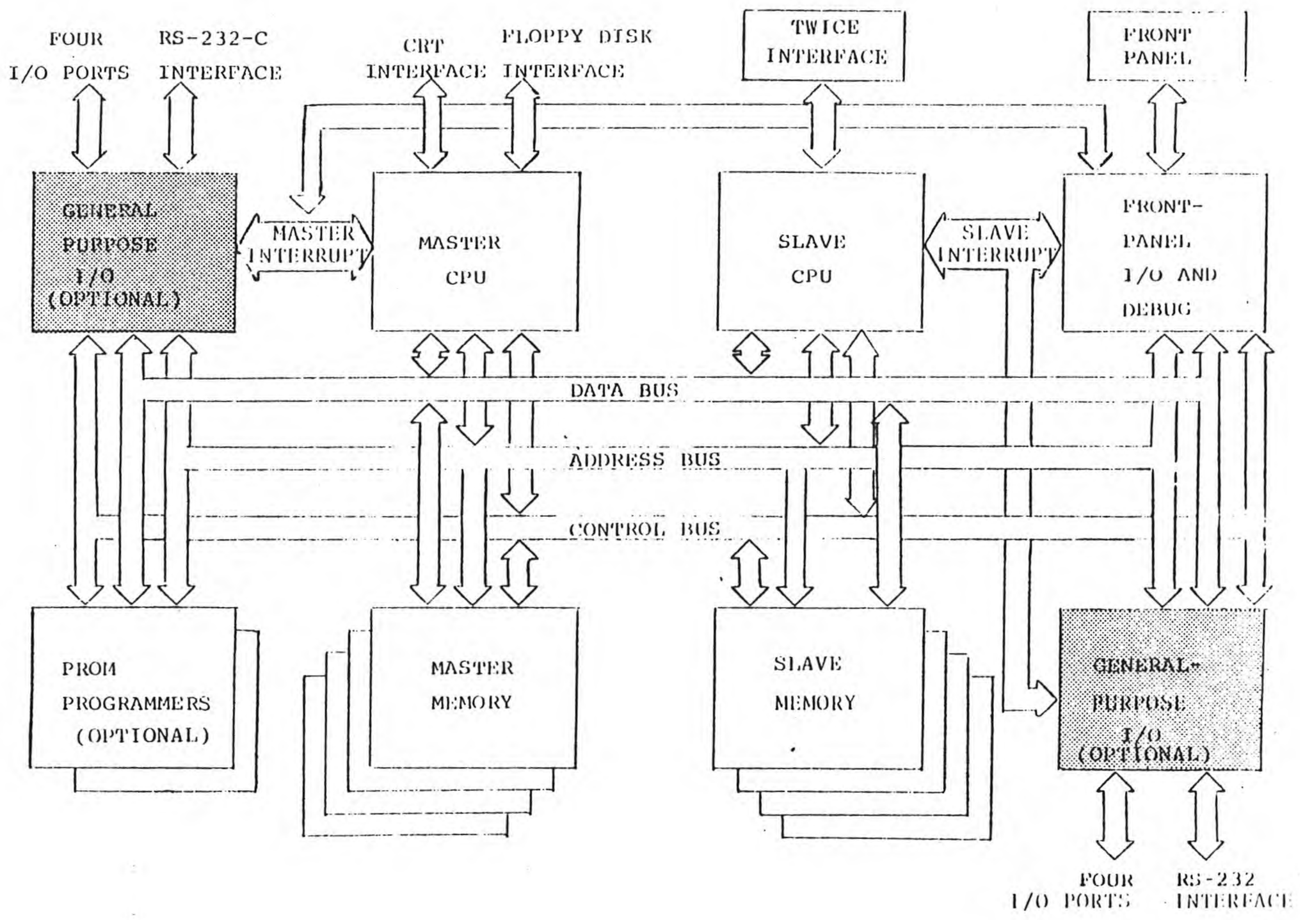


FIGURE 2-9

GENERAL PURPOSE I/O

Interface to the user or maintenance front panels and control of the system during user Debug operations are provided by this board, shown in the shaded portion of Figure 2-10. The board can be broken down into several major sections which control operations independently. The first major section consists of breakpoint logic which monitors the address bus and compares the contents to the contents of the Debug breakpoint registers or to the address set on the maintenance front-panel address switches. When the two addresses are identical, control logic interrupts the Master CPU and halts the Slave CPU. At this point, the contents of memory may be examined. A second major section is Program Counter logic which is used in Debug mode of operation, where the software needs to read the contents of the Slave CPU Program Counter. Jump control constitutes the next major portion, and is activated by the system via the Master CPU. When activated, a jump address is set into the Program Counter Logic for transfer via the data bus to the Slave CPU. Master/Slave control constitutes the final major section of interest. This logic determines which CPU has control of the address, data, and control buses. Since both Master and Slave cannot have control at the same time, the control logic allows one to run and pauses the other.

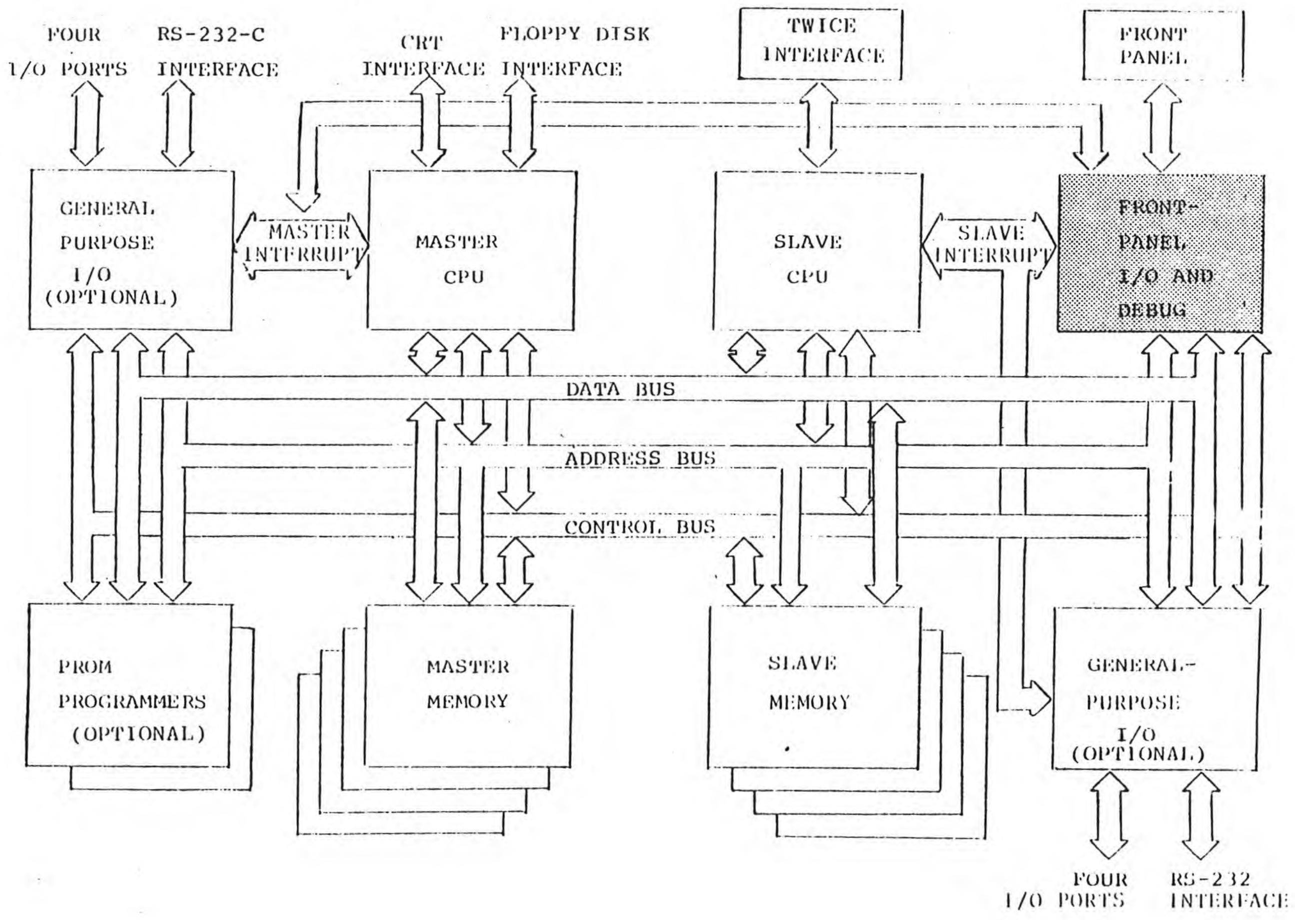


FIGURE 2-10

FRONT-PANEL I/O AND DEBUG

CHAPTER 3

MASTER CPU AND I/O BOARD

3.0 GENERAL DESCRIPTION

Data transfer and processing tasks, as well as overall system control are handled by the Master CPU and I/O board. In system operation, the board controls the three major system buses, and performs I/O transfers with the disk system and console device.

The central processor function is performed by a Signetics 2650 microprocessor and its supporting logic, consisting of crystal-controlled clock and bus drivers for the bi-directional data bus, the address bus, and the control bus.

This chapter outlines the function of the Master CPU and I/O board in the system, and describes the logic contained on the board to implement this function. The 2650 microprocessor itself is not described. Those readers wishing detailed information on this device are referred to the Signetics 2650 Reference Manual.

3.1 OVERALL BLOCK DIAGRAM

A simplified data flow block diagram of the board is shown in Figure 3-1. The board can be divided into four major sections for purposes of discussion. The first major block is the CPU itself and the supporting logic. The second section includes I/O logic for communication with the disk system and the console device. The third section consists of interrupt priority decoding and interrupt vector generation logic. The last major section consists of the Baud rate and interval timer logic.

3.1.1 THE CPU

The 2650 CPU is the controlling element, accessing memory for instructions and executing these instructions.

Logic surrounding the CPU device is intended to provide bus driving capability and timing for the CPU's operations. Secondary functions include Slave memory bank switching and power-on reset.

The clock consists of a 10 MHz crystal-controlled oscillator which runs continuously. The output of the oscillator drives a divide-by-eight circuit to give the required 1.25 MHz clock signal for the CPU. Part of the divider consists of a flip-flop which in turn is controlled by an external "hold" signal which

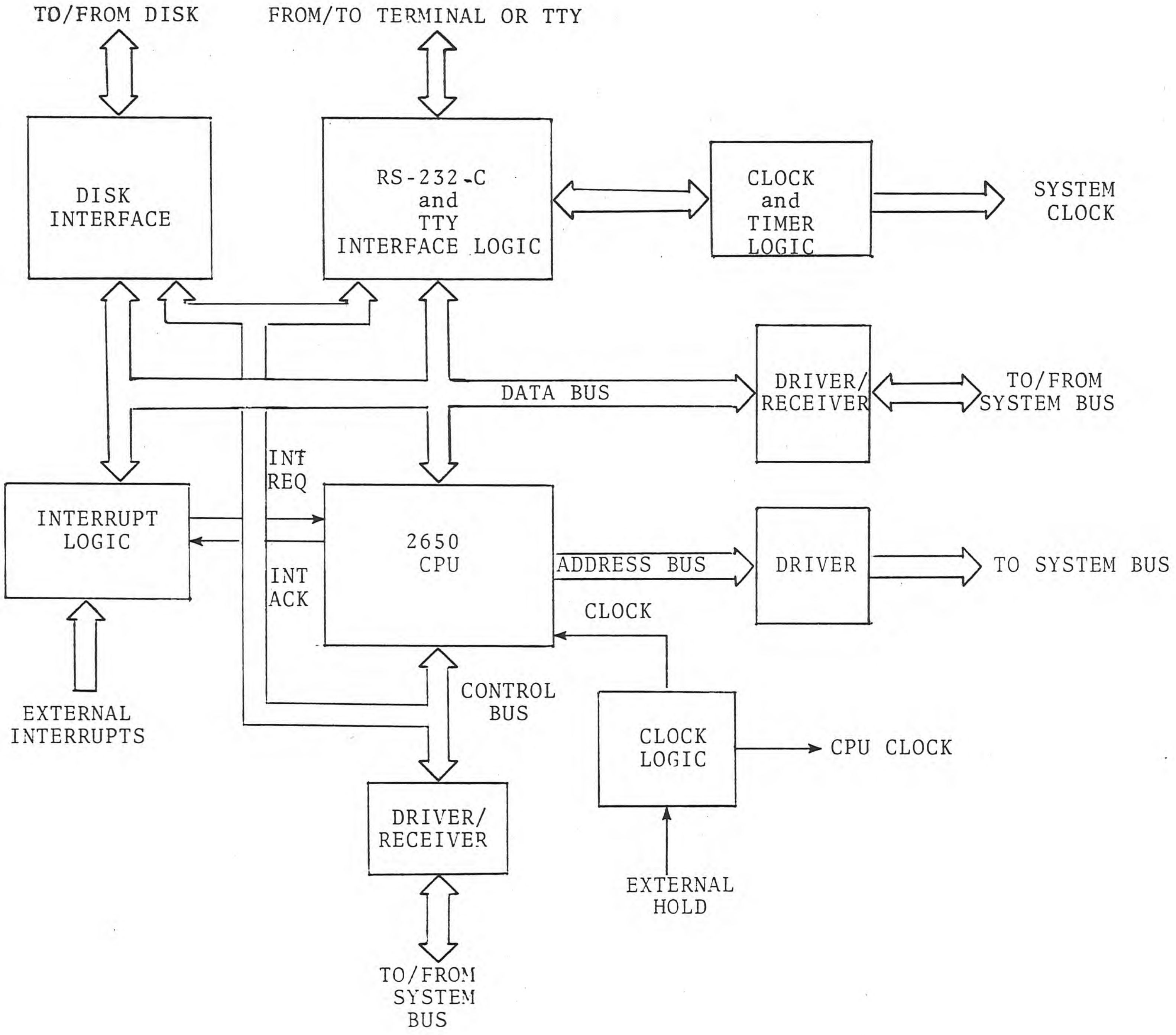


FIGURE 3-1
DATA FLOW BLOCK DIAGRAM

effectively puts the CPU in a hold state during master/slave operation.

The CPU's data lines are buffered by transceivers whose inputs and outputs are connected to the system data bus. Each of the address lines is buffered similarly, using tri-state drivers whose outputs may be floated to allow DMA operation in future applications.

The drivers and receivers provide the necessary system bus driving capability. When the CPU is in a "run" state, the drivers are enabled, and signals are passed to the bus or received from the bus. When the CPU is halted, the drivers are disabled and put into a high-impedance state so the buses can be controlled by external devices, or by the Slave CPU.

Logic is provided to restart the CPU upon power turn-on. When the +5 V supply comes up to its operating level, the power-on detector initiates a "reset" signal which resets the 2650 CPU and also sends a system-wide reset to the rest of the modules. This causes the CPU to fetch an instruction from location 00 in Master memory and causes the bootstrap program to be initiated.

3.1.2 I/O LOGIC

Communications between the system and peripherals such as the floppy disk and the CRT terminal are handled by the CPU board I/O logic. There are two separate interfaces: a serial interface for the console device, either CRT terminal or TTY, and a parallel interface for the disk system. The serial interface incorporates a UART (Universal Asynchronous Receiver Transmitter) to perform the necessary serial-to-parallel and parallel-to-serial conversion for the TTY or terminal. Status and control registers are included to provide the data exchange protocol signals required by the console device and the CPU. Interface to the disk includes lines which allow control of the optional line printer which is interfaced to the disk system.

Clock generation logic provides a crystal-controlled 9.984 MHz system clock, 110 Baud to 1200 Baud TTY and CRT terminal clocks, a 38.4 kHZ I/O clock, and a 10 or 100 ms interval timer.

3.1.3 INTERRUPT LOGIC

Since the TWIN system operates with an interrupt-driven structure, a means must be provided to assign priority to interrupts from devices capable of such interrupts, and to generate interrupt vectors when the CPU has accepted an interrupt. The interrupt logic consists of a series of latches which are set when an interrupt occurs, and reset when the interrupt is acknowledged and the vector is output to the data bus. Priority encoders arbitrate priority among the devices requesting interrupts and enable the interrupt vector generation logic.

3.1.4 DMA LOGIC

Capability for future implementation of DMA boards within the TWIN system is provided by logic on the CPU board which allows the 2650 to be halted and the address and data buses to be floated. In this way the DMA boards can control the address and data buses, allowing an external device to operate directly with system memory, without utilizing the CPU and its I/O functions.

3.2 DETAILED FUNCTIONAL DESCRIPTION

This section describes individual logic blocks within the board.

3.2.1 2650 CPU

The 2650 and its supporting logic, which will be described subsequently, provide control of the system. Acting on instructions accessed from memory, the 2650 manipulates data and provides control using its arithmetic and logic functions.

The CPU clock is generated by a 10 MHz oscillator whose output is divided by eight in two stages. The first stage divides the clock frequency by four. The second stage divides the clock by two and provides a means of stopping the clock in the low signal state for any number of half cycles. The front panel "hold" signal is generated by the maintenance front panel and stops the clock during single step or breakpoint operation. The basic 10 MHz oscillator uses a series resonant crystal as the feedback element and two inverters, which are biased into the linear region, as amplifiers. The 10 MHz output is also buffered onto the system bus for Slave CPU use.

The CPU address, data, and control outputs are buffered by a series of drivers and receiver/transmitters. Bidirectional bus drivers receive data from the system bus and transmit it to the 2650, or receive data from the 2650 and transmit it to the bus. These devices are enabled by ANDing the $\overline{\text{RUN}}$ and R/$\overline{\text{W}}$ outputs from the 2650 through inverters and gates. When the 2650 is not running (i.e., halted), the transeivers are disabled and put into the high impedance state. In this state the data bus is floating and can be driven by other modules in the system such as the Slave CPU or Debug boards.

Unidirectional bus drivers buffer the address outputs from the 2650 and drive the system address bus. Each device is enabled by an inverted RUN output from the 2650.

Control signals such as $\overline{\text{INTACK}}$, $\overline{\text{FLAG}}$, $\overline{\text{WRP}}$, and $\overline{\text{M}}$/IO are brought directly to a tri-state driver. Other signals such as RUN and OPREQ are conditioned by logic which is driven by the 2650 outputs. OPREQ is ANDed with INTACK and delayed 250 ns by an RC network. The resultant signal is squared by a Schmitt trigger and inverted before being transmitted to a driver which is enabled or disabled by the RUN output of the 2650.

SIGNALS AND DESCRIPTIONS

$\overline{\text{ADD0}}$-$\overline{\text{ADD15}}$	Address lines. $\overline{\text{ADD0}}$ is the low-order bit. $\overline{\text{ADD13}}$ and $\overline{\text{ADD14}}$ have special significance for memory operations, $\overline{\text{ADD13}}$ is a normal address bit; for I/O operations it indicates whether I/O is extended or non-extended. $\overline{\text{ADD14}}$ operates normally for memory operations and indicates data or control outputs for I/O operations.
$\overline{\text{DB0}}$-$\overline{\text{DB7}}$	Data lines. An eight-bit bi-directional bus carrying data and instructions into and out of the processor.
$\overline{\text{FLAG}}$	This output indicates the state of the flag bit in the processor status word.
$\overline{\text{INTACK}}$	This output acknowledges that an interrupt vector must be placed on the data bus by the interrupting device.
$\overline{\text{OPREQ}}$	This output signal notifies external devices that an external operation is in progress and the device performing the operation must respond with an $\overline{\text{OPACK}}$ signal (see below).
$\overline{\text{HOLD}}$	This is an input from an external device indicating that an external operation is in progress. It is input to the $\overline{\text{OPACK}}$ input of the processor to hold it in a "wait" state until the operation is complete.

SIGNALS AND DESCRIPTIONS (continued)

$\overline{\text{M}}$/IO	This indicates whether the operation in progress is a memory or I/O operation.
R/$\overline{\text{W}}$	Indicates whether the operation in progress is a read or write.
$\overline{\text{WRP}}$	This is a strobe output to signal external devices that data and address buses are stable.
$\overline{\text{MSTR RUN}}$	This is an output derived from the processor RUN output that indicates a processor run condition (not halted or paused).
$\overline{\text{RUN}}$	same as $\overline{\text{MSTR RUN}}$.
$\overline{\text{RESET}}$	This is an output for system reset, occurring when +5V power is turned on.
$\overline{\text{SENSE}}$	This is an input to the processor SENSE line to set the status word SENSE bit.
$\overline{\text{INT1}}$-$\overline{\text{INT15}}$	These are interrupt inputs from external devices. They are priority encoded and result in an interrupt request to the processor and an interrupt vector being placed on the data bus.
$\overline{\text{DBG INT}}$	An input from the Debug module requesting an interrupt.
$\overline{\text{DBG VEN}}$	An output from the interrupt circuitry enabling the Debug vector generation circuitry.
$\overline{\text{MST INT 0}}$	An output indicating that a processor interrupt has been requested.
FP HOLD	An input putting the processor in a hold state by shutting off the clock.

SIGNALS AND DESCRIPTIONS (continued)

CPU CLK	A 10 MHz output from the processor clock oscillator.
$\overline{\text{DMA PAUSE}}$	An input from external DMA devices, pausing the processor and floating the address, data, and control buses.
CMEM	Indicates that Master CPU is accessing Slave (common) memory.
SYS CLK	9.984 MHz output for system synchronization.
I/O CLK	Approximately 38 KHz output for I/O device synchronization.

When the address or control drivers are disabled, they are put in the high impedance mode and control of the buses is given to other modules in the system.

HOLD and SENSE control inputs are inverted and drive the $\overline{\text{OPACK}}$ and $\overline{\text{SENSE}}$ inputs of the 2650.

The power-on reset circuit consists of an RC network which charges during power on and a Schmitt trigger which provides a 50 ms pulse to reset system logic. The reset signal is ORed with the front panel reset signal.

For the Master CPU to access Slave memory, any 16K section of Slave memory is mapped into the Master 16K-32K address space. The 16K section is specified by two control bits in a Master I/O command. The bit formats are described in the paragraphs concerning I/O operations. When the Master address is 16K or above, address bit 14 is on. This outputs the bank switching bits to the address bus as bits 14 and 15.

Basic CPU timing is shown in Figure 3-2. Timing for Read, Write, and Interrupt operations is shown for reference. The system bus uses a HOLD signal rather than an operation acknowledge ($\overline{\text{OPACK}}$). If the memory or I/O device cannot return or take data within 200 ns of OPREQ, a hold is generated by the device until the data to be read is valid, or the write data has been received. In the case of the 2102 device used in the Master and Slave memory boards, the hold duration is approximately 650 ns.

3.2.2 I/O LOGIC

The I/O logic on the Master CPU board provides the following functions:

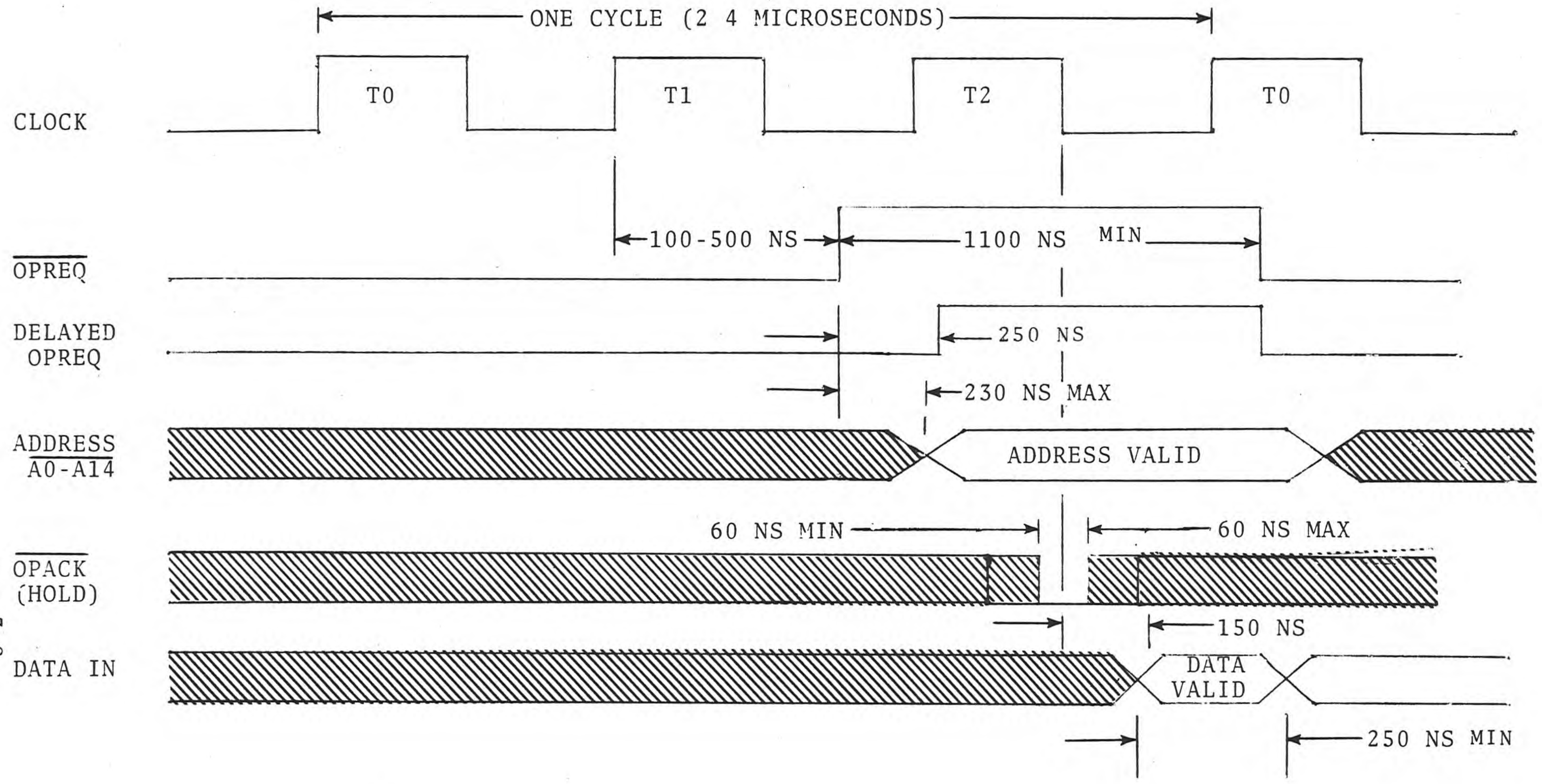
1) TTY/CRT I/O
2) Disk I/O
3) I/O decoding

This section has a separate bi-directional I/O data bus and driver-receiver pairs to the bus. It contains all I/O decoding, I/O output storage registers, and input gating.

The I/O logic can be broken into sections for purposes of discussion. TTY and RS-232-C interfaces share the same logic, while the disk interface requires separate logic.

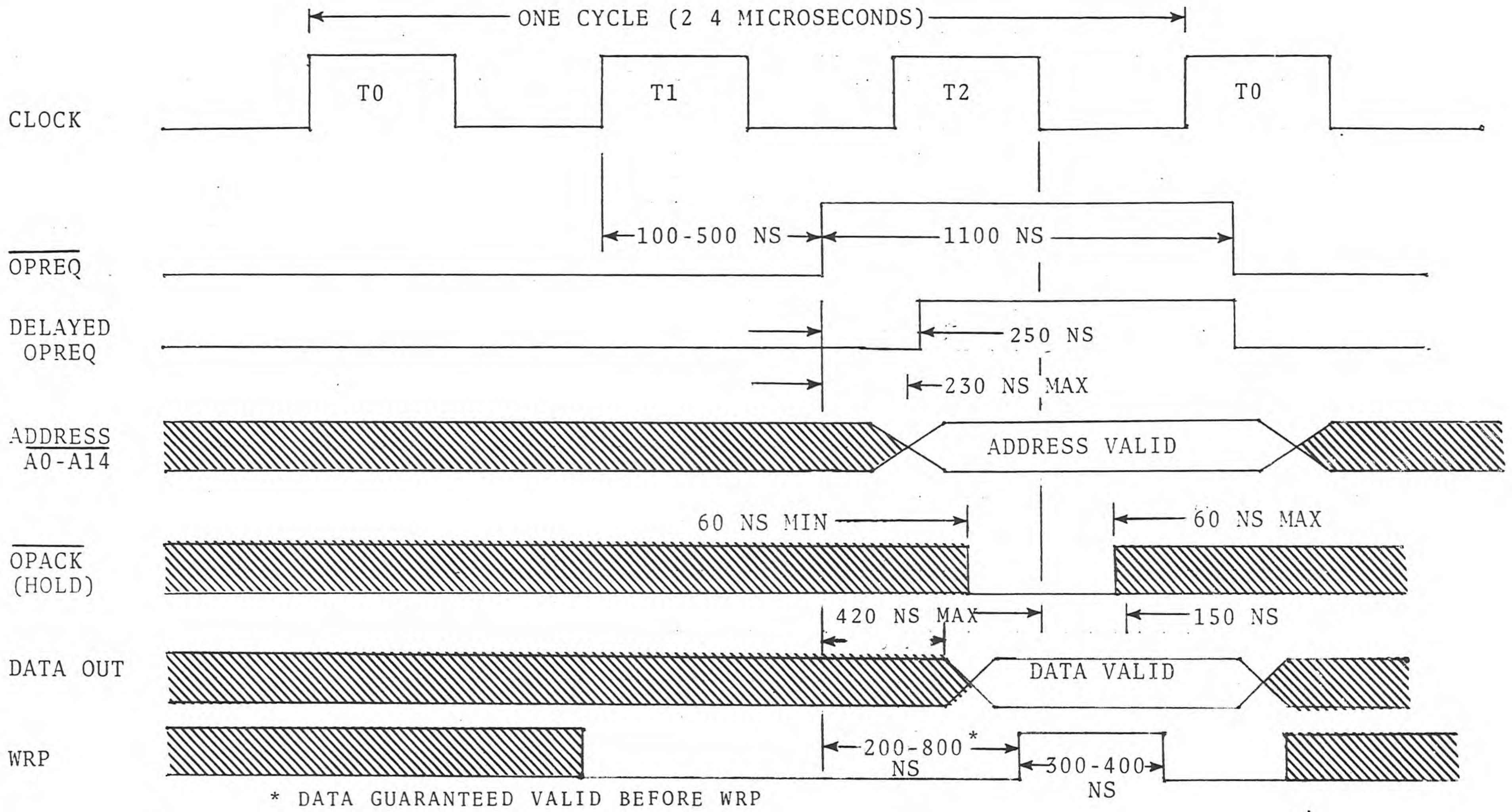
TTY/CRT I/O

This section provides both a limited EIA serial interface and a TTY current loop interface. However, only one terminal can be connected to the board at any time. A UART is the heart of the circuit, providing parallel to serial and serial to



a. MEMORY OR I/O READ TIMING

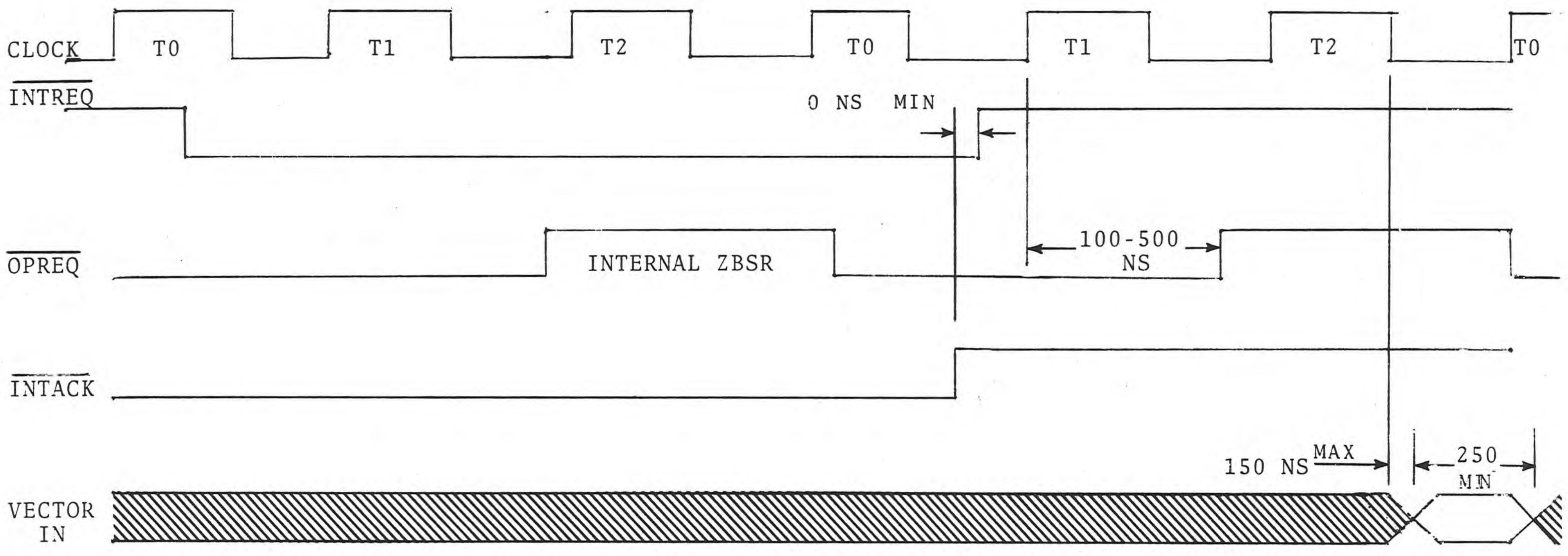
FIGURE 3-2
CPU TIMING



b. MEMORY OR I/O WRITE TIMING

FIGURE 3-2 (continued)
CPU TIMING

LAST CYCLE OF CURRENT INSTRUCTION



c. INTERRUPT TIMING

FIGURE 3-2 (continued)
CPU TIMING

parallel conversion, status, and programmable control functions. The UART clock rate and number of stop bits are switch-selected. One switch position provides 110 Baud and 2 stop bits, while the other position provides 1 stop bit and a rate which is jumper selectable from 150 to 1200 Baud. Parity is selected under program control.

Serial data input received from the EIA receiver or the current loop interface is fed to the UART. The UART shifts the serial data in and sets the data available (DA) status when a complete character has been received. If interrupts are enabled, this triggers the TTY IN interrupt (level 4), or the RS-232 IN interrupt (level 6). The parallel data is enabled out of the UART through tri-state buffers onto the internal data bus is by reading port E8. This also resets DA.

Serial data out is transmitted by performing a write to port E8. The parallel data is presented to the UART from the internal bus and strobed into the UART by applying a pulse to its DS input. This initiates the serial output stream to an open collector driver for the TTY current loop interface and to a bipolar driver for the EIA interface. When the character is transmitted, TBMT (transmit buffer empty) is set by the UART. If interrupts are enabled this sets the TTY OUT interrupt (level 5) or the RS-232 OUT interrupt (level 7).

Status indications consisting of DA, TBMT, OR (overrun), FE (framing error) or PE (parity error) are available by reading port E9. Tri-state buffers enable these signals onto the internal bus from the UART. Formats for the data and control ports are shown in Table 3-1.

A TTY-related function is control of the TTY paper tape reader. A separate reader-on flip-flop and open collector driver provide the interface for this function. The flip-flop is turned on under program control and is reset upon receipt of the start bit, thus reading a character at a time.

The status byte which controls the TTY paper tape reader, the UART parity circuits, and enables the TTY interrupts is output by a write to port E9.

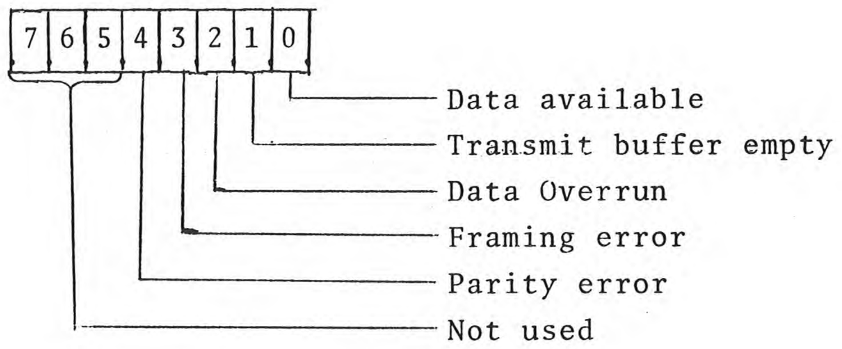
The TTY/CRT connector is a 25 pin connector conforming to EIA RS-232-C standards which is mounted on the development computer rear panel. This connector is connected to connector P2 on the top edge of the Master CPU board by a ribbon cable.

To interface to the CRT or other EIA peripherals, certain signals of the RS-232-C interface are pulled up permanently to a logic 1. These signals are clear to send, data set ready, carrier detected, data terminal ready, and request to send.

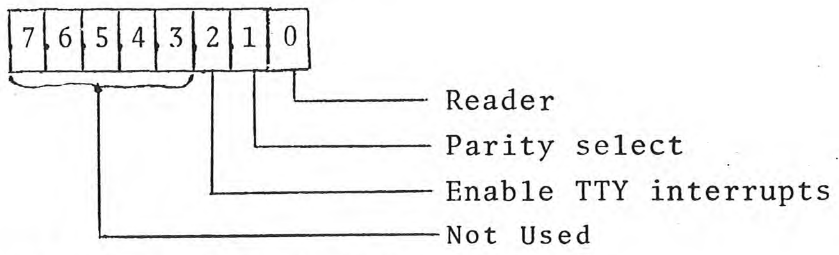
TABLE 3-1

CRT/TTY FORMATS

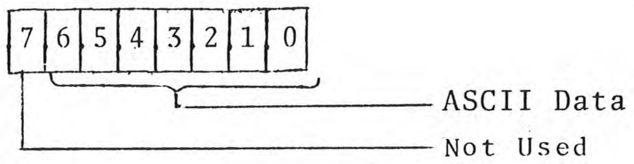
STATUS BYTE (E9 READ)



CONTROL BYTE (E9 WRITE)



DATA BYTE (E8 READ/WRITE)



DISK I/O

The disk I/O logic consists of a data I/O port, a control I/O port, and interrupt flip-flops. The byte formats are shown in Table 3-2. The eight disk input data bits (DIDO-7) are terminated with a pull up-pull down network and received with tri-state buffers. The buffers are gated onto the internal bus by reading port EB. Output data is stored in an eight-bit register by writing to port EB. The outputs are driven with tri-state buffers. Disk status input is enabled to the data bus with tri-state drivers by reading port EA. Disk controller signals PBZY, FLG are filtered with an RC network and shaped by a Schmitt trigger, then fed to interrupt flip-flops. A tape signal is provided only for possible future replacement of the disk drives by a high speed paper tape reader. Disk control is performed by a write to port EA. This outputs a control byte which consists of the three disk control bits, a spare output bit and interrupt enables for the disk and printer. The three output disk control bits are buffered with tri-state drivers.

For a detailed description of the floppy disk command formats and timing refer to Appendix A.

I/O DECODING

The I/O decoding logic recognizes extended I/O commands to addresses E8-EF. The logic provides enable lines for the tri-state input gates and clocks for the output registers. The logic also provides strobes for the bank switch and timer interrupt control discussed in other sections.

3.2.3 INTERRUPT LOGIC

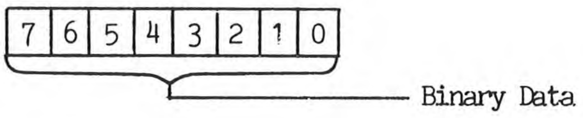
The interrupt logic establishes priority and generates the interrupt routine address vector for 16 system interrupts. It also handles the interaction of the 16 level Debug interrupt generator.

Interrupts are stored in a dedicated interrupt flip-flop. The 16 interrupt lines are continuously sampled at a 10 MHz rate. If one or more interrupts are present, the CPU interrupt line is activated. When the CPU responds with $\overline{\text{INTACK}}$, sampling is stopped and the interrupt address for the highest priority line present is encoded. Refer to Figure 3-2 for timing. The vector is placed on the data bus during $\overline{\text{INTACK}}$ for input to the CPU. I/O circuits can also recognize their vector on the data bus during $\overline{\text{INTACK}}$ to determine when their interrupt request has been acknowledged. On-board interrupts are reset with a vector decoder which is enabled during $\overline{\text{INTACK}}$. The Debug module interrupts have lower priority and are given an enable line ($\overline{\text{DBG VEN}}$) only if no system I/O interrupts are pending. Table 3-3 lists priorities and vector addresses for the interrupts.

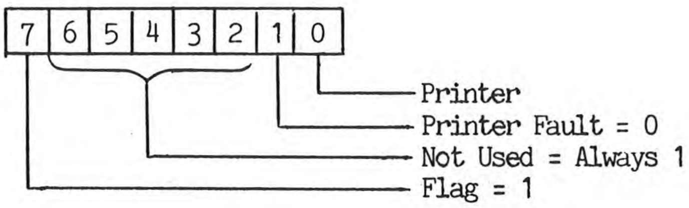
TABLE 3-2

DISK FORMATS

DATA BYTE (EB READ/WRITE)



STATUS-BYTE (EA READ)



CONTROL BYTE (EA WRITE)

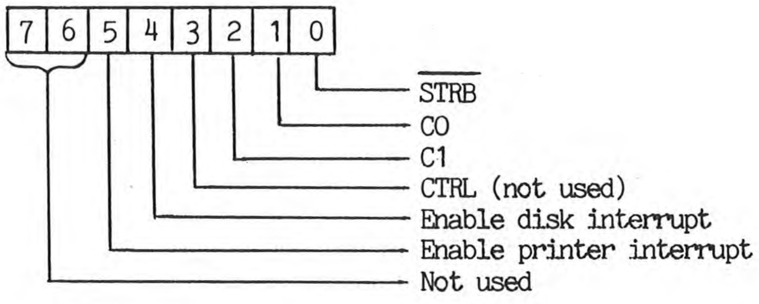


TABLE 3-3

INTERRUPT PRIORITY ASSIGNMENTS AND ADDRESS VECTORS

Priority	Address	Function	
0	0000	Reset	
1	0002	NA	
2	0004	Paper Tape Rdr	
3	0006	NA	
4	0008	TTY In	
5	000A	TTY Out	
6	000C	RS-232 In	Master
7	000E	RS-232 Out	I/O
8	0010	NA	
9	0012	Timer	
10	0014	Printer	
11	0016	Floppy Disk	
12	0018	NA	
13	001A	PROM Program 1	
14	001C	PROM Program 2	
15	001E	NA	
16	0020	Slave SVC 1	
17	0022	Slave SVC 2	
18	0024	Slave SVC 3	
19	0026	Slave SVC 4	Service
20	0028	Slave SVC 5	Requests
21	002A	Slave SVC 6	
22	002C	Debug SVC 1	
23	002E	Debug SVC 2	
24	0030	Breakpoint 1	
25	0032	Breakpoint 2	
26	0034	Single Cycle	
27	0036	Halt	Debug
28	0038	Diag. Control	Hardware
29	003A	NA	
30	003C	NA	
31	003E	NA	

NA = NOT ASSIGNED

3.2.4 DMA LOGIC

DMA operations require that the CPU be paused, and the address and data buses floated for the necessary DMA device access. This is accomplished by the DMA PAUSE signal. $\overline{\text{DMA PAUSE}}$ is ORed with $\overline{\text{MSTR PAUSE}}$ and drives the PAUSE input of the 2650. After completion of the current instruction, the 2650 RUN output will go high. This is used to disable the address and data bus drivers, which float the buses.

3.2.5 BAUD RATE AND TIMER LOGIC

A crystal-controlled oscillator, identical (except for frequency) to that used in the CPU clock logic, is used for the Baud rate and interval timer generators. The 9.984 MHz output of the oscillator is directed through a driver to the rest of the system through the system bus. A second path is directed to a divide-by-130 divider logic which in turn drives a divider to generate the 110 Baud clock. Another divider chain provides the 150 to 1200 Baud clocks. Dividers provide either 10 ms or 100 ms timer intervals for timer interrupts. A jumper on the board selects which interval will be used. See Figure 3-3.

3.3 UTILIZATION OF THE BOARD

The Master CPU and I/O board is normally checked out with the system and ready for use with the CRT terminal. If a TTY or TTY-compatible terminal is used with the board, switch S1 must be placed in the 110 Baud position. If a standard TTY is to be used, the current loop and TTY reader control circuits must be modified. Refer to Appendix C for modification procedures.

Although normal usage will dictate use of all the interrupt capability, jumpers E1 through E12 can be cut to disable any interrupts required.

Although the Master CPU and I/O board was designed specifically for use in the TWIN system, it can be used in products of a general nature. It contains complete clock, I/O and address circuits, and only requires the addition of memory to form a complete CPU. External power is required as listed below. The board is designed to be plugged into a 100-pin printed circuit connector.

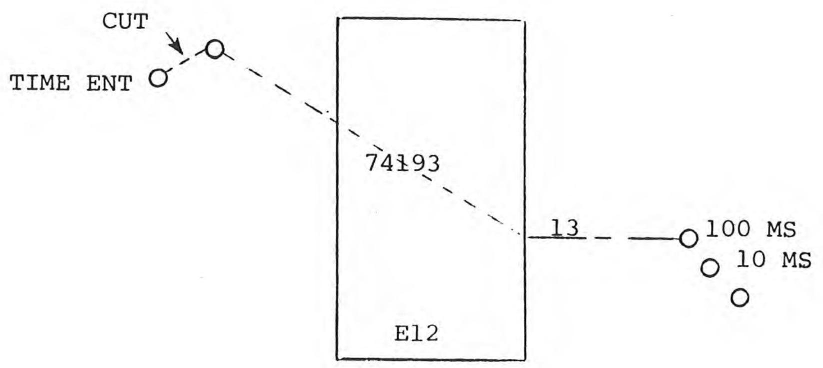
The following supply voltages are required:

1.	+5 V at	1.9	A
2.	+12 V at	50	mA
3.	-12 V at	50	mA

The board should be mounted vertically with side support to minimize vibration. Sufficient space should be left on both sides of the board to permit heat dissipation. Other components radiating large volumes of heat should not be located near the board.

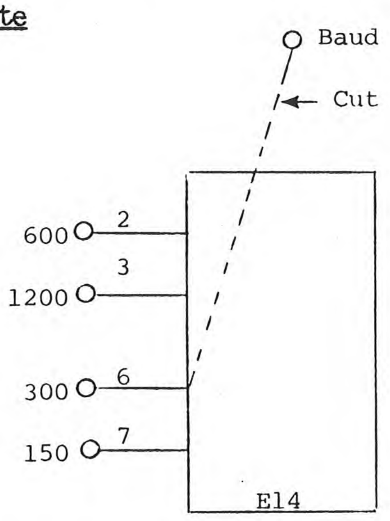
A list of edge connector pin assignments is found in Tables 3-4 through 3-6. A list of internal signals and system bus signals used by the Master CPU and I/O board is given in Table 3-4.

Internal Timer



Cut trace between Time Ent and 100 ms, to remove 100 ms Timer. Add strap from 10 ms to Time Ent for 10 ms Timer.

Baud Rate



Cut trace to normal 300 Baud, add Strap to desired Baud Rate.

Figure 3-3

TIMER STRAPPING DIAGRAMS

TABLE 3-5

PIN LIST, TOP EDGE CONNECTOR P1

Pin	Signal	Description
1	DOD7	Disk Output Bit 7
2	DOD6	Disk Output Bit 6
3	DOD5	Disk Output Bit 5
4	DOD4	Disk Output Bit 4
5	DOD3	Disk Output Bit 3
6	DOD2	Disk Output Bit 2
7	DOD1	Disk Output Bit 0
8	DODO	Disk Output Bit 0
9	not used	
10	not used	
11	CTRL	
12	$\overline{\text{C1}}$	
13	$\overline{\text{CO}}$	
14	not used	
15	not used	
16	$\overline{\text{STRB}}$	
17	gnd	
18	gnd	
19	gnd	
20	not used	
21	FLG	
22	TAPE	
23	not used	
24	not used	
25	not used	
26	not used	
27	$\overline{\text{PFLT}}$	
28	not used	
29	D1D7	Disk Input Bit 7
30	D1D6	Disk Input Bit 6
31	D1D5	Disk Input Bit 5
32	D1D4	Disk Input Bit 4
33	D1D3	Disk Input Bit 3
34	D1D2	Disk Input Bit 2
35	D1D1	Disk Input Bit 1
36	D1DO	Disk Input Bit 0
37	gnd	
38	FLG	
39	$\overline{\text{PBZY}}$	
40	$\overline{\text{PFLT}}$	

TABLE 3-5

PIN LIST, TOP EDGE CONNECTOR P2

Pin	Signal	Description
1	GND	Protective Ground
2	TTY XMIT-	TTY Current Loop Input
3	EIA RCV DATA	RS-232-C Serial Input
4	RDR CTRL+	Tape Reader Current Output
5	EIA XMIT DATA	RS-2332-C Serial Output
6	RDR CTRL-	Tape Reader Control Output
7	REQ TO SND	RS-232-C Request to Send Input
8	TTY RCVR+	TTY Current Loop Input
9	CLR TO SND	RS-232-C Clear to Send Output
10	TTY RCVR-	TTY Current Loop Output
11	DATA SET RDY	RS-232-C Data Set Ready Output
12	SPARE	
13	EIA SIG GND	RS-232-C Signal Ground
14	DATA TERM RDY	RS-232-C Data Terminal Ready
15	SIG DET	RS-232-C Carrier Detected Output
16-24	NOT USED	
25	TTY XMIT+	TTY Current Loop Output

TABLE 3-6

PIN LIST, MASTER CPU BOARD

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
5-8	not used	
9	gnd	
10	gnd	
11	+12 V	Power Input
12	+12 V	Power Input
13	+12 V	Power Input
14	-12 V	Power Input
15	gnd	
16	gnd	
17	$\overline{\text{ADR0}}$	Address Bit 0
18	$\overline{\text{ADR1}}$	Address Bit 1
19	$\overline{\text{ADR2}}$	Address Bit 2
20	$\overline{\text{ADR3}}$	Address Bit 3
21	$\overline{\text{ADR4}}$	Address Bit 4
22	$\overline{\text{ADR5}}$	Address Bit 5
23	$\overline{\text{ADR6}}$	Address Bit 6
24	$\overline{\text{ADR7}}$	Address Bit 7
25	$\overline{\text{ADR8}}$	Address Bit 8
26	$\overline{\text{ADR9}}$	Address Bit 9
27	$\overline{\text{ADR10}}$	Address Bit 10
28	$\overline{\text{ADR11}}$	Address Bit 11
29	$\overline{\text{ADR12}}$	Address Bit 12
30	$\overline{\text{ADR13}}$	Address Bit 13
31	$\overline{\text{ADR14}}$	Address Bit 14
32	$\overline{\text{ADR15}}$	Address Bit 15
33	CMEM	Slave Memory Enable Output
34-35	not used	
36	$\overline{\text{DATA 0}}$	Data Bit 0
37	$\overline{\text{DATA 1}}$	Data Bit 1
38	$\overline{\text{DATA 2}}$	Data Bit 2
39	$\overline{\text{DATA 3}}$	Data Bit 3
40	$\overline{\text{DATA 4}}$	Data Bit 4
41	$\overline{\text{DATA 5}}$	Data Bit 5
42	$\overline{\text{DATA 6}}$	Data Bit 6
43	$\overline{\text{DATA 7}}$	Data Bit 7
44-50	not used	

TABLE 3-7

PIN LIST, MASTER CPU BOARD (continued)

Pin	Signal	Description
51	not used	
52	$\overline{\text{M}}$/IO	Memory/IO Enable Output
53	$\overline{\text{WRP}}$	Write Pulse Output
54	$\overline{\text{OPREQ}}$	Operation Request Output
55	R/$\overline{\text{W}}$	Read/Write Command Output
56	$\overline{\text{HOLD}}$	Operation Acknowledge Input
57	not used	
58	$\overline{\text{RUN}}$	Run condition Output
59	$\overline{\text{RESET}}$	Reset Input
60	not used	
61	$\overline{\text{INTACK}}$	Interrupt Acknowledge Output
62	$\overline{\text{INT 0}}$	Interrupt Level 0 Input
63	$\overline{\text{INT 1}}$	Interrupt Level 1 Input
64	$\overline{\text{INT 2}}$	Interrupt Level 2 Input
65	$\overline{\text{INT 3}}$	Interrupt Level 3 Input
66	$\overline{\text{INT 4}}$	Interrupt Level 4 Input
67	$\overline{\text{INT 5}}$	Interrupt Level 5 Input
68	$\overline{\text{INT 6}}$	Interrupt Level 6 Input
69	$\overline{\text{INT 7}}$	Interrupt Level 7 Input
70	$\overline{\text{MST INT 8}}$	Interrupt Level 8 Input
71	$\overline{\text{MST INT 9}}$	Interrupt Level 9 Input
72	$\overline{\text{MST INT 10}}$	Interrupt Level 10 Input
73	$\overline{\text{MST INT 11}}$	Interrupt Level 11 Input
74	$\overline{\text{MST INT 12}}$	Interrupt Level 12 Input
75	$\overline{\text{MST INT 13}}$	Interrupt Level 13 Input

TABLE 3-7

PIN LIST, MASTER CPU BOARD (continued)

Pin	Signal	Description
76	$\overline{\text{MST INT 14}}$	Interrupt Level 14 Input
77	$\overline{\text{MST INT 15}}$	Interrupt Level 15 Input
78	$\overline{\text{MST PSE}}$	Master CPU Pause Input
79	$\overline{\text{DBG PSE}}$	Debug Interrupt Input
80	$\overline{\text{DBG VEN}}$	Debug Vector Enable Output
81	$\overline{\text{MST INT 0}}$	Interrupt Acknowledge Output
82	not used	
83	$\overline{\text{DMA PAUSE}}$	Pause DMA
84	$\overline{\text{MAST RUN}}$	Master CPU Run Output
85	$\overline{\text{SENSE}}$	CPU Sense Input
86	$\overline{\text{FLAG}}$	CPU Flag Output
87	$\overline{\text{F.P. HOLD}}$	Front-Panel Hold Input
88-93	not used	
94	I/O CLK	Clock Output for I/O Devices
95	CPU CLK	
96	SYS CLK	Master System Clock Output
97	gnd	
98	gnd	
99	gnd	
100	gnd	

CHAPTER 4

PROM/RAM MEMORY

4.0 INTRODUCTION

The PROM/RAM Memory module provides RAM storage for the Master CPU as well as PROM for the system bootstrap loader. Each module is capable of storing up to 4K eight-bit bytes of object code and data for program execution. This code consists of programs intended to service user commands, and to direct data to the CPU to allow it to perform the required function.

The PROM portion of the module is capable of storing up to 2K eight-bit bytes of object code. The bootstrap loader occupies the lower 256 locations in memory. For general use, the PROM array has provision for installing board-mounted switches to allow user selection of a base address. It should be noted that in the TWIN only one PROM/RAM Memory board has PROM installed.

Each module has switch-selected base addresses, allowing unique selection of modules in the system.

Communications with the CPU is accomplished through the system data bus, which is driven by a set of drivers and receivers for the RAM array and a set of drivers for the PROM array. The drivers and receivers are enabled only when the board is accessed.

Memory is enabled when the CPU is executing an instruction requiring a read from, or write to memory. During each CPU cycle, control signals are active to control transfer of data to or from the data bus.

4.1 OVERALL BLOCK DIAGRAM

A block diagram of the board is shown in Figure 4-1. There are five major blocks: RAM, PROM, Address Logic, Operation Control, and Timing Logic. The following paragraphs discuss the function of each of these blocks in typical memory read and write operations.

4.1.1 RAM Read

At the beginning of a CPU read cycle, addresses are output to the system address bus by the CPU. At approximately the same time, control signals become active to determine the direction of data flow on the data bus and to enable modules which will transmit or receive the data.

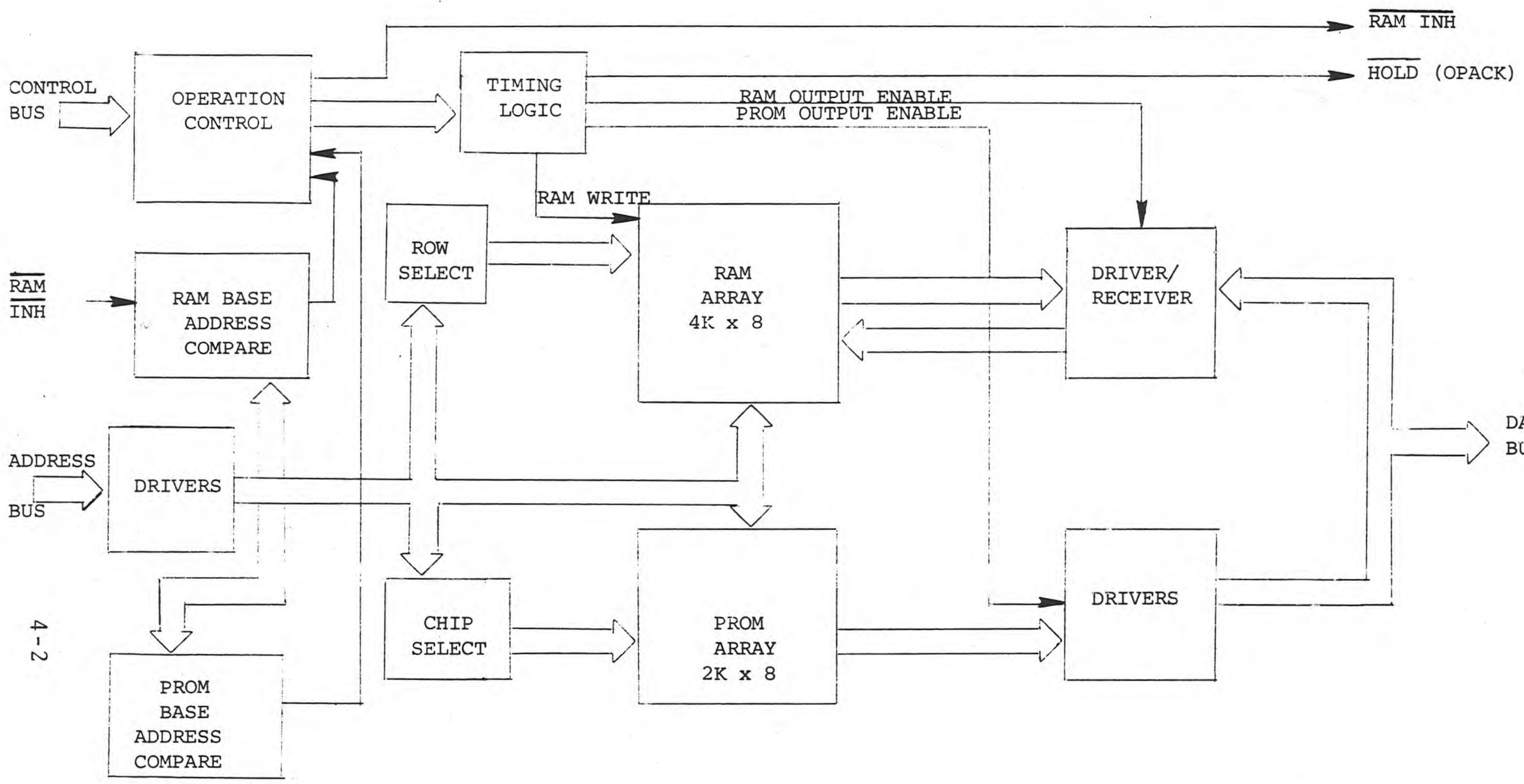


FIGURE 4-1
PROM/RAM
MEMORY BLOCK DIAGRAM

Address lines are input to the PROM/RAM Memory board from the system bus. The upper four address bits (A12-15) are used to select one of the memory boards in the system. This is done by comparing the four address bits with the board address selected by four board mounted switches. If the address compares, RAM MODULE SELECT is generated and the read sequence begins. The other conditions which must be met to output data are that OPREQ must be active, R/$\overline{\text{W}}$ must be high, and $\overline{\text{MEM}}$/IO must be low (i.e., a memory read operation must be specified by the CPU). If all these conditions are met, RAM OUTPUT ENABLE is active, and drivers output the data from the RAM array to the data bus. A signal derived from $\overline{\text{OP REQ}}$ and $\overline{\text{MEM}}$/IO triggers a one-shot, which outputs a 650 nanosecond pulse (RAM HOLD), which is output to the system bus as $\overline{\text{OPACK}}$ and holds the CPU until the data on the bus is stable.

4.1.2 RAM Write

Writing to the RAM array involves the same addressing sequence as a read. In the case of RAM write operation, however, the signal R/$\overline{\text{W}}$ must be low. A write operation triggers a one-shot and generates a 400 ns RAM WRITE pulse to strobe the data into the selected row of devices. At the same time, another one-shot is triggered to generate the $\overline{\text{OPACK}}$ signal to the CPU.

4.1.3 PROM Read

The sequence of reading from the PROM array is similar to a RAM read. If PROM is resident on the board (it will not be on other memory boards in the system), a decoder outputs PROM RESIDENT, which in turn triggers a PROM hold one-shot, and HOLD (OPACK) will be output to the bus. At the same time, PROM OUT ENABLE enables the PROM output drivers which output the data to the data bus. To eliminate interference from RAM occupying the same address space, PROM OUT ENABLE generates $\overline{\text{RAM INH}}$ which prevents any RAM arrays from being enabled.

4.2 DETAILED FUNCTIONAL DESCRIPTION

Referring to Figure 4-1, the following describes the five major functional blocks.

4.2.1 Memory Organization

The RAM array is composed of 32 type 2102-2 memory devices, each capable of storing 1024 bits. These are organized into a four row by eight column array, for a total capacity of 4096 eight-bit words.

The board contains programming provisions to allow the 4K of RAM to be based at a selected 4K segment of the 64K TWIN memory space. A means for disabling portions of RAM when PROM is installed in the same memory space is also included.

Although the basic printed circuit board is designed to support up to eight 1702A PROMs, circuitry is installed to support only the 256 bytes of bootstrap PROM.

4.2.2 Memory Addressing

The board contains a group of four switches which allow the base address of the RAM array to be placed at any multiple of 4K between 0 and 60K. Similarly, switches and a jumper are provided to allow the base address of the PROM array to be placed at any multiple of 2K between 0 and 62K. These are factory wired to location 0.

The sixteen address bus bits (A0-A15) are divided into three groups to address the PROM array. The five most significant bits (A11-A15) are applied to the PROM base address comparator. The other inputs to the comparator are the five base address programming switches, the CMEM bus signal, and a CMEM programming jumper. This jumper allows the board to be used in either Master or Slave memory portions of the system. The comparator output enables the PROM read circuits. The second group of bits, A8-A10, are decoded into eight lines to enable one of the eight PROM devices. These signals are also applied, via jumpers, to the PROM read circuits. These switches are used to enable the PROM read circuits only for those PROM devices that are actually installed on the board. The third group of address lines A0-A7, select one of the 256 bytes of the enabled PROM to be output to the data bus drivers.

The address lines are also divided into three groups to address the RAM array on the board. Address lines A12 to A15 are applied to the RAM base address comparator, whose output enables the RAM read and write circuits. Lines A10 and A11 are decoded to select one of the four "rows" of the array, while A0-A9 select the proper bit within the selected row.

The PROM read enable signal is also applied to the system bus as a RAM inhibit (RAM INH) signal. This signal is used to inhibit the output of the RAM base address comparator. Thus, if PROM is installed in an address space where RAM also exists, the RAM output will be disabled when PROM at the same address is being read.

4.2.3. Timing

Memory is enabled when the CPU is executing an instruction requiring a read or write involving memory. During the CPU cycle, control signals become active to control transfer of data to or from memory. $\overline{\text{OPREQ}}$ alerts memory and I/O units that a data transfer is in the offing. $\overline{\text{MEM}}$/IO signifies that either a memory of I/O unit is to be activated, and R/$\overline{\text{W}}$ determines whether the requested operation is a read or write.

On the board, $\overline{\text{MEM}}$/IO and $\overline{\text{OPREQ}}$ are ANDed to generate the basic memory timing signal. When PROM is addressed, the output of the PROM read enable circuit is used to enable a data bus driver which places the PROM outputs on the data bus. At the same time a one-microsecond $\overline{\text{HOLD}}$ signal is generated which is ultimately applied to the 2650 OPACK input. This provides the delay necessary to ensure that valid PROM data is on the data bus before the CPU proceeds to read the data.

If RAM is addressed (and no PROM occupies the same address space), and a read operation is requested, the RAM data output drivers are enabled to place RAM data on the data bus. If the operation is a write, a 400 ns write pulse is generated and applied to the RAM write circuits to write the data into the RAM array. This write pulse occurs after a 200 ns delay to allow settling of the address and data buses. In either case, a 650 ns $\overline{\text{HOLD}}$ signal is generated to Hold the CPU until the operation is complete.

4.2.4 -9 Volt Regulator

The -12 VDC input from the system bus is applied to a 3 V Zener diode which provides the -9 VDC power for operation of the single PROM.

4.3 UTILIZATION

The PROM/RAM Memory board may be used as a general-purpose RAM or ROM storage unit. Table 2-5 lists the required input lines; all levels are TTL-compatible. The unit is designed to be mounted vertically in a 100-pin printed circuit connector. Guides should be provided on both vertical edges to prevent movement.

Exposure to dust and vibration should be avoided for reliable operation. The board should not be mounted near other components which radiate heat toward the Memory board.

The board requires the following DC voltages:

(1) +5 VDC at 1.2 A
(2) -12 VDC at 60 mA

4.3.1 Switches and Jumpers

The PROM/RAM Memory Board has provision for switches and jumpers which provide user control over the operation of the board. Jumpers disabling RAM and PROM hold circuits must be cut when the board is used with different CPU devices.

4.3.2 RAM Base Address Selection

Switch E8A is used to set the base address of the 4K RAM. This may be set at any 4K multiple between 0 and 60K. The switch weights and sample settings for a base address of 20,480 are as follows:

SWITCH	WEIGHT	SETTING FOR BASE ADDRESS OF 20,480 (4096x5)
E8A-1	4,096	ON
E8A-2	8,192	OFF
E8A-3	16,384	ON
E8A-4	32,768	OFF

4.3.3 PROM Base Address Selection

Switch E8B when installed by the user and jumper J1 select the base address of the 2K PROM at any 2K multiple between 0 and 62K. The switch weights and sample settings for a base address of 12,288 are as follows:

SWITCH	WEIGHT	SETTING FOR BASE ADDRESS OF 12,288(2048x6)
E8B-1	2,048	OFF
E8B-2	4,096	ON
E8B-3	8,192	ON
E8B-4	16,384	OFF
J1	32,768	OPEN

4.3.4 Resident PROM Selection

Switch D14 when installed by the user, designates the address ranges occupied by PROM and for which RAM should be disabled. Jumper J5 can be left open to disable this function. The switch positions, the PROM location, and the corresponding address spaces are as follows, where 'BA' refers to the PROM base address.

SWITCH	PROM	ADDRESS RANGE
D14-1	A10	BA+ 0-BA+ 255
D14-2	B10	BA+ 256-BA+ 511
D14-3	A12	BA+ 512-BA+ 767
D14-4	B12	BA+ 768-BA+1023
D14-5	A13	BA+1024-BA+1279
D14-6	B13	BA+1280-BA+1535
D14-7	A15	BA+1536-BA+1791
D14-8	B15	BA+1792-BA+2047

4.3.5 Other Jumpers

J2 determines whether the RAM is selected when the CMEM bus signal is low or when it is high. J8 performs the same function for the PROM area. J3 can be used to disable the RAM completely and J4 will disable the PROM completely. J6 determines if the one microsecond PROM hold signal is applied to the HOLD bus line; and J7 does the same for the 650 millisecond RAM hold signal. The PROM hold jumper must be cut when the board is used with RAM alone.

TABLE 4-1

PIN LIST, MASTER MEMORY BOARD

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
5-12	not used	
13	-12 V	Power Input
14	-12 V	Power Input
15	gnd	
16	gnd	
17	$\overline{\text{ADR0}}$	Address Bit 0
18	$\overline{\text{ADR1}}$	Address Bit 1
19	$\overline{\text{ADR2}}$	Address Bit 2
20	$\overline{\text{ADR3}}$	Address Bit 3
21	$\overline{\text{ADR4}}$	Address Bit 4
22	$\overline{\text{ADR5}}$	Address Bit 5
23	$\overline{\text{ADR6}}$	Address Bit 6
24	$\overline{\text{ADR7}}$	Address Bit 7
25	$\overline{\text{ADR8}}$	Address Bit 8
26	$\overline{\text{ADR9}}$	Address Bit 9
27	$\overline{\text{ADR10}}$	Address Bit 10
28	$\overline{\text{ADR11}}$	Address Bit 11
29	$\overline{\text{ADR12}}$	Address Bit 12
30	$\overline{\text{ADR13}}$	Address Bit 13
31	$\overline{\text{ADR14}}$	Address Bit 14
32	$\overline{\text{ADR15}}$	Address Bit 15
33	$\overline{\text{CMEM}}$	Slave memory select input
34	$\overline{\text{RAMINH}}$	RAM inhibit output
35	not used	
36	$\overline{\text{DATA 0}}$	Data Bit 0
37	$\overline{\text{DATA 1}}$	Data Bit 1
38	$\overline{\text{DATA 2}}$	Data Bit 2
39	$\overline{\text{DATA 3}}$	Data Bit 3
40	$\overline{\text{DATA 4}}$	Data Bit 4
41	$\overline{\text{DATA 5}}$	Data Bit 5
42	$\overline{\text{DATA 6}}$	Data Bit 6
43	$\overline{\text{DATA 7}}$	Data Bit 7
44-50	not used	

TABLE 4-1

PIN LIST, MASTER MEMORY (CONTINUED)

Pin	Signal	Description
51	not used	
52	$\overline{M}$/IO	Memory/IO Enable
53	$\overline{\text{SPARE}}$	SPARE
54	$\overline{\text{OPREQ}}$	Operation Request
55	R/$\overline{W}$	Read/Write Command
56	$\overline{\text{HOLD}}$	Hold Command to CPU
57	not used	
58	not used	
59	$\overline{\text{RESET}}$	System Reset Input
60-96	not used	
97	gnd	
98	gnd	
99	gnd	
100	gnd	

CHAPTER 5

SLAVE CPU AND TWICE INTERFACE

5.0 GENERAL DESCRIPTION

The Slave CPU executes and supports debugging of user programs both within the development computer and externally through the TWICE in-circuit emulation assembly. The heart of the Slave CPU is a special version of the 2650 microprocessor (CT354). Operation is essentially identical to the 2650 contained on the Master CPU module. Interface and multiplex circuitry connect the CPU to the system bus or to the TWICE cable, depending on the mode of user operation. Other logic includes a forced jump sequencer, memory protect gating and instruction fetch logic. Forced jump logic enables the system to start the user program at any location, or jump the Slave to a trace routine where register contents are dumped. Memory protect gating is programmed by the system to confine a Slave to a particular 16K or 32K section of slave memory. Fetch decode circuitry provides a prediction of the next instruction fetch. This is used to allow the debug logic to know when an instruction is being fetched for storage and later retrieval during a trace operation, and to time the start of the forced jump sequence. This circuitry is replaced in later units by use of a special 2650 IC (CT354) which has the fetch signal multiplexed with the RUN line.

5.1 OVERALL BLOCK DIAGRAM

A simplified block diagram is shown in Figure 5-1. The Slave CPU can be divided into the following sections shown in the detailed block diagram Figure 5-2:

1) TWICE (user) and CPU support logic

2) 2650 address, data paths

3) Forced Jump logic

4) Priority interrupt logic

5) Memory protect logic

6) Fetch decode logic

The TWICE control logic channels 2650 data, address and control signals to the bus or to the TWICE cable depending on the user mode and the type of operation. An internal bi-directional data bus connects directly to the 2650 and handles user data, bus data, and forced jump bytes.

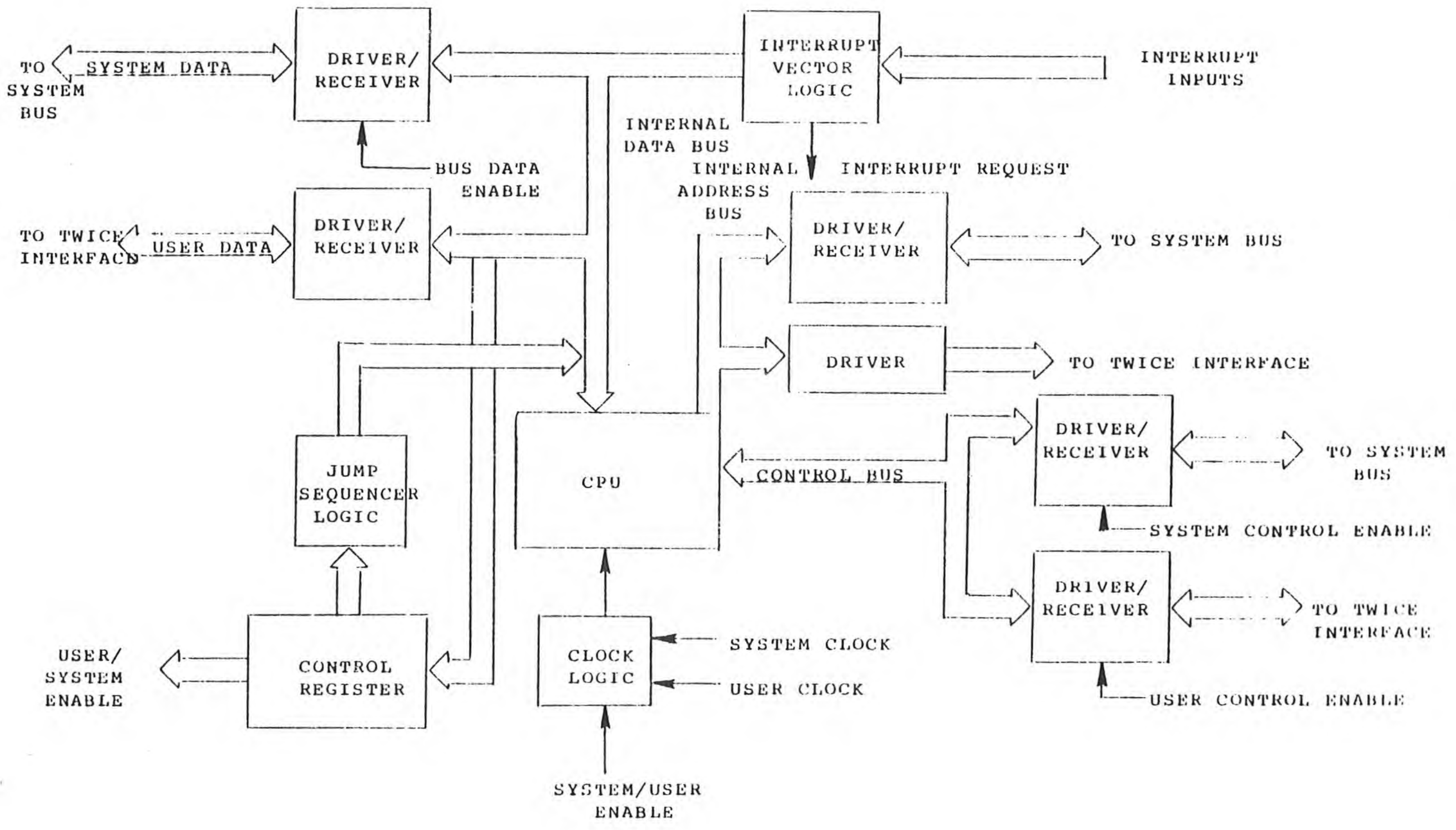


FIGURE 5-1 SLAVE CPU AND TWICE INTERFACE

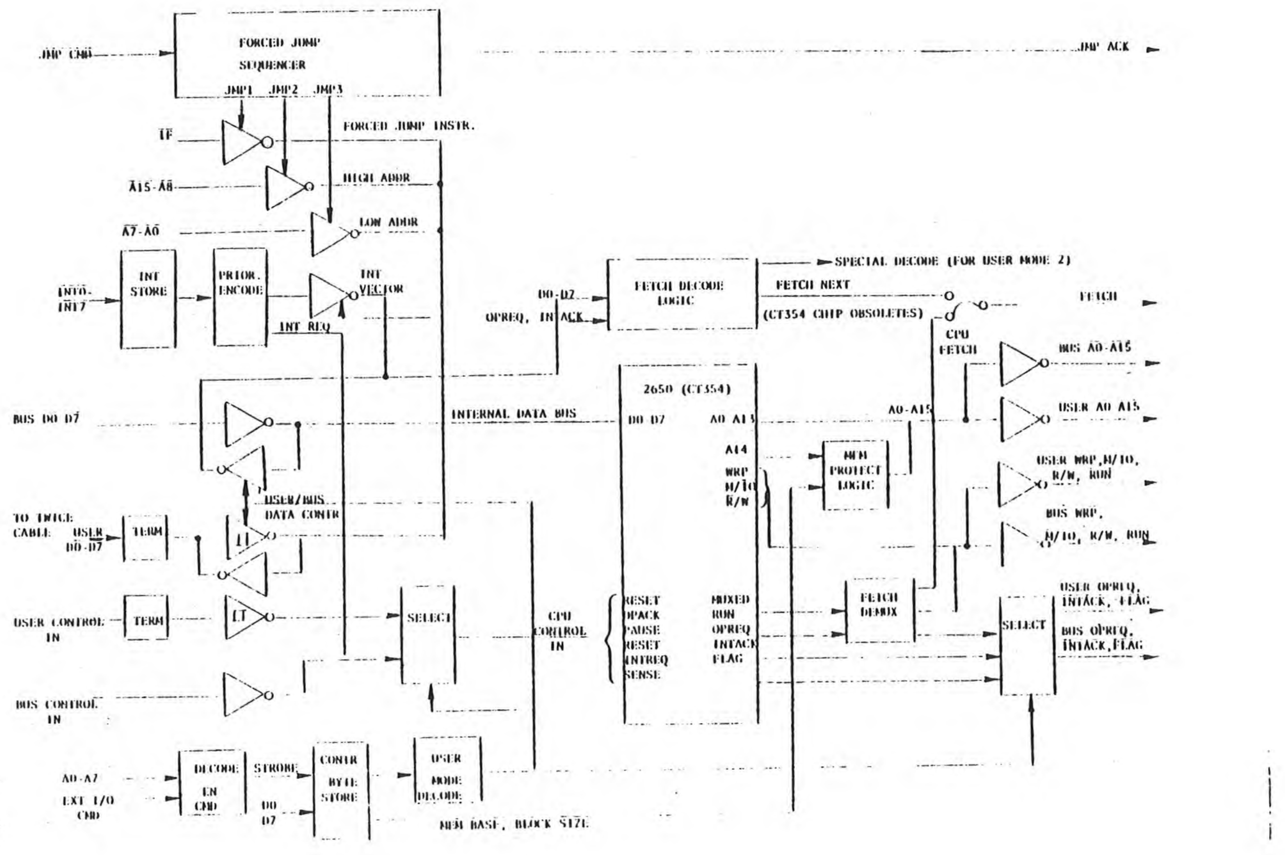


FIGURE 5-2

SLAVE CPU DETAILED BLOCK DIAGRAM

The forced jump logic, under command from the Debug module, places an unconditional branch instruction on the data bus in place of the next memory instruction fetch. This is followed by two address bytes multiplexed to the data bus and originating from the Debug module. The priority interrupt logic receives up to 8 interrupt levels, priority encodes them and interrupts the 2650 CPU. When the CPU acknowledges the interrupt, the encoded vector is placed on the external data bus providing the CPU with the interrupt subroutine address.

Memory protect logic confines the Slave to any 16K, 32K or 64K section of memory by overriding address bit 15 and/or 14 under Master CPU control. This feature allows multiple Slave operation with memory protected. It also automatically relocates the Slave address space, so that programs operate relative to zero, no matter to which section of memory the Slave is confined.

The fetch decode logic stores the op code on the data bus during each instruction fetch, decodes it sufficiently to determine the number of $\overline{\text{OPREQ}}$ cycles in the instruction, and finally provides a "FETCH" signal predicting the next instruction fetch. This indication is used for debug storage of the instruction address during trace operation and to time the forced jump sequence. Inclusion of a special 2650 (CT354) in later units with the $\overline{\text{FETCH}}$ signal multiplexed with the RUN line obsoletes most of the fetch decode logic. The special 2650 $\overline{\text{RUN}}$ line is demultiplexed to create separate $\overline{\text{RUN}}$ and $\overline{\text{FETCH}}$ lines.

5.2 DETAILED FUNCTIONAL DESCRIPTION

5.2.1 TWICE (User) and CPU Support Logic

The Master CPU controls the Slave operating mode through the use of a command byte whose format is shown in Figure 5-3. The command byte is issued with an extended I/O instruction to address E0. The byte contains memory protect, bus address and block size, an interrupt disable bit, two user mode bits, and an active/inactive bit.

The memory protect bits are further discussed in Section 5.2.5. The interrupt disable bit masks the interrupt line to the CPU and is used during certain Debug routines. The inactive bit pauses the CPU, holds it in a reset state and disables all bus lines. Figure 5-4 is a detailed block diagram of the control logic.

The user mode bits control TWICE operating mode and are coded as follows:

Mode 0	00	-	Normal Slave environment. Slave Memory and System I/O
Mode 1	01	-	Slave Memory and external prototype system I/O through the TWICE cable.

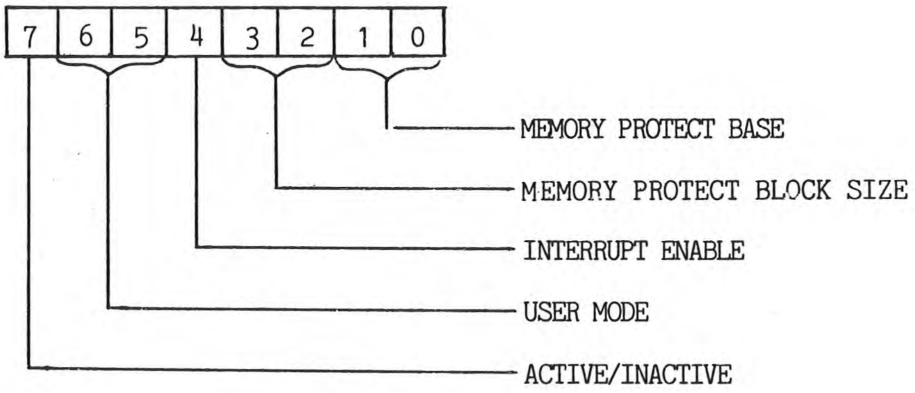


FIGURE 5-3

SLAVE COMMAND BYTE

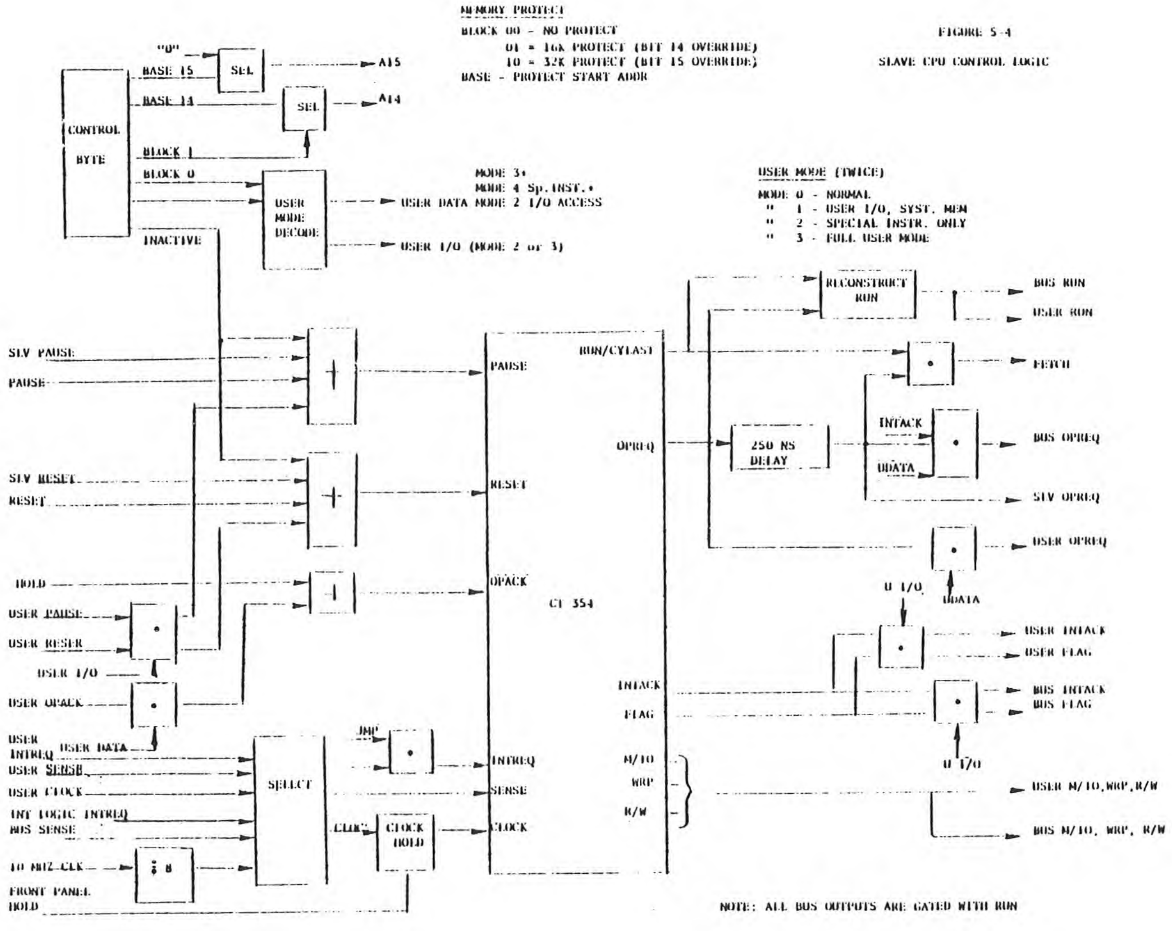


FIGURE 5-4

SLAVE CPU CONTROL LOGIC

Mode 2 11 - External prototype system memory and
prototype system I/O

Figure 5-5 shows control line connections for each user mode. In any user mode (modes 1 and 2) the user clock replaces the normal CPU clock. This prevents synchronization problems when switching between external and internal operations in mode 1. Certain signals are available to the bus in any mode. This allows system control of the Slave even while in TWICE mode. On the output side, OPREQ is multiplexed to the user or to the system depending on the mode and type of CPU operation. A separate $\overline{\text{OPREQ}}$ is sent to the debug card in any mode for maintenance front panel single step. FLAG and $\overline{\text{INTACK}}$ are routed to the user in modes 1 and 2 only. All other outputs are routed to both in any mode. These signals have no validity unless $\overline{\text{OPREQ}}$ is present.

The clock hold circuit is used for maintenance front panel single step or can be used by an I/O device slower than 10 microseconds. The 2650 requires an $\overline{\text{OPACK}}$ in 10 microseconds or loss of internal data could occur. The clock hold circuit holds the clock low for an integral number of clock cycles. The clock is restarted synchronously.

The system Slave clock is produced by dividing the 10 MHz bus clock by 8 to produce a square wave with an 800 nanosecond period.

The $\overline{\text{OPREQ}}$ 250 nanosecond delay is required as in the Master CPU to stabilize the 2650 address lines before accessing the system. The circuit is identical to the Master CPU. OPREQ to the TWICE cable is not delayed.

The special 2650 processor (CT354) has $\overline{\text{FETCH}}$ multiplexed with $\overline{\text{RUN}}$. By ANDing $\overline{\text{RUN}}$ and $\overline{\text{OPREQ}}$ together, a demultiplexed $\overline{\text{FETCH}}$ signal is produced. To eliminate a leading edge race condition, the delayed $\overline{\text{OPREQ}}$ is also ANDed with the result. The original $\overline{\text{RUN}}$ signal is reconstructed by OR'ing $\overline{\text{OPREQ}}$ and $\overline{\text{RUN}}$ together. A bridging circuit delays the trailing edge, removing potential glitches. Timing is illustrated in Figure 5-6.

5.2.2 2650 Address and Data Paths

All fourteen address bits are continuously output to the TWICE cable through tristate drivers. Address bits A0-A13 are output to the system bus through tri-state drivers gated by $\overline{\text{RUN}}$ unless a forced jump is in progress. Output of bits A14 and A15 to the bus is controlled by the memory protect logic discussed in paragraph 5.2.4.

		TWICE Mode	
INPUTS	0	1	2
CLOCK	system	user	user
PAUSE	system	user + system	user + system
RESET	system	user + system	user + system
OPACK (Hold)	system	user if I/O	user + system
INTREQ	system	user	user
SENSE	system	user	user

		TWICE Mode	
OUTPUTS	0	1	2
RUN	system + user	system + user	system + user
OPREQ	system + F.P.	user if I/O + F.P.	user + F.P.
WRP	system + user	system + user	system + user
M/I/O	system + user	system + user	system + user
FLAG	system	user	user
R/W	system + user	system + user	system + user
INTACK	system	user	user

FIGURE 5-5

CONTROL LINE MULTIPLEXING

The internal data bus connecting directly to the 2650 CPU is buffered to the system bus by bi-directional receiver/driver pairs. The bi-directional user cable data lines are driven with tristate bus drivers, and received with Schmitt trigger inverters. The inverters feed the internal bus through tristate drivers. The cable is terminated with pull up, pull down resistors at the characteristic line impedance. The forced jump instruction and data bytes are also multiplexed onto the internal data bus as discussed in Section 5.2.3.

Note that the data is gated to the system bus during forced jumps and user data input. This allows viewing all operations from the maintenance front panel during single step operating mode.

5.2.3 Forced Jump Logic

The detailed block diagram, Figure 5-2, illustrates the forced jump sequencer. A timing diagram is shown in Figure 5-7.

The Debug module commands a forced jump with a "JUMP CMD" pulse. This enables the jump circuitry. The next instruction fetch clocks the sequencer to JMP1 enabling Hex '1F' (an unconditional branch instruction) on the internal data bus. This overrides the normal memory fetch. The next $\overline{\text{OPREQ}}$ clocks the sequencer to JMP 2 which sends a jump acknowledge to the Debug module. The Debug module then presents the jump address on the system address bus. JMP 2 outputs the high address byte from the address byte to the internal data bus. The next $\overline{\text{OPREQ}}$ clocks the sequence to JMP 3 and outputs the low byte from the address bus to the internal data bus. The trailing edge of $\overline{\text{OPREQ}}$ ends the sequence and resets the sequencer. In many Debug modes, after the 1F instruction is given to the Slave, the Slave is paused and the Master executes a routine before returning the bus to the Slave. When control is returned, the second two bytes of the instruction are presented as if no interruption had occurred.

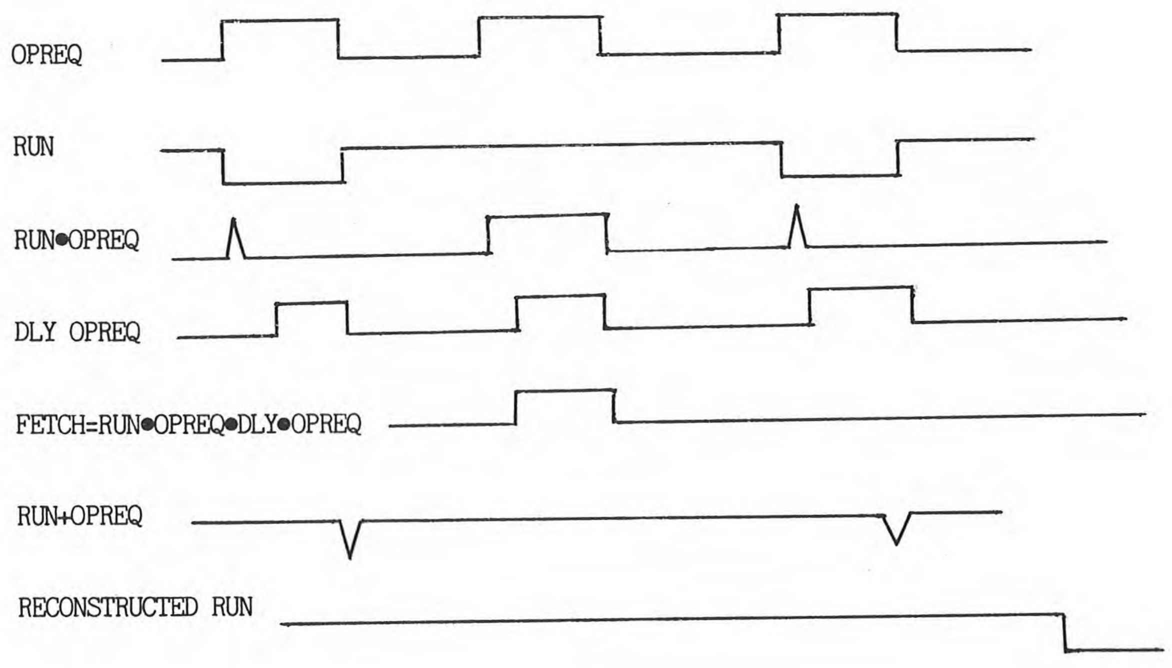


FIGURE 5-6

RUN DEMULTIPLEX TIMING

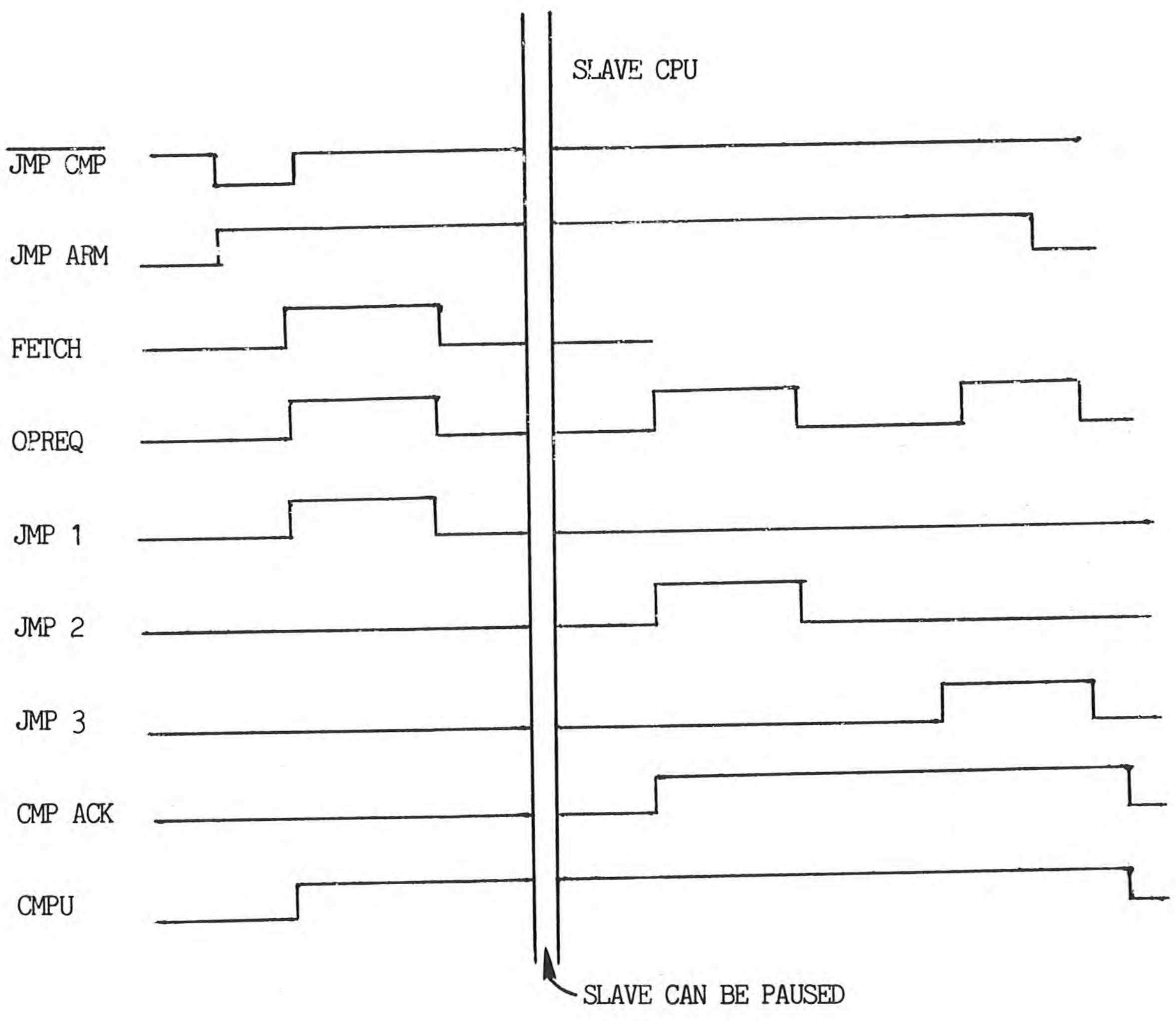


FIGURE 5-7

JUMP SEQUENCER TIMING

5.2.4 Slave CPU Priority Interrupt Logic

Nine interrupt lines enter the Slave module from the system bus. One is a direct CPU interrupt which bypasses the priority interrupt logic. To use this line, a device must generate its own vector during $\overline{\text{INTACK}}$. The other eight lines are sampled at 10 MHz rate and enter a priority encoder. If one or more interrupts are present, the CPU interrupt line is activated. When the CPU responds with INTACK, the sampling is stopped and the highest priority line is encoded and placed on the external data bus. Vectors start at location zero for the highest priority interrupt and are spaced every two bytes. This allows a branch relative indirect instruction to be placed at the vector location. An interrupting device must look at the contents of the data bus during $\overline{\text{INTACK}}$ to determine that its interrupt has been acknowledged.

5.2.5 Memory Protect Logic

The Slave CPU can be confined to a particular section of Slave memory under Master command. Four bits of the Slave command byte (Figure 5-3) specify the base address and the block size, as shown below:

Base Control Bits	Base Address
00	0
01	16K
10	32K
11	48K

The base control bits specify bits A15 and/or A14 if the CPU address is overridden. The block size bits determine whether bit 14 and/or 15 is specified by the base control bits.

Block Size Bits	Block Size
00	No Protect-16K
01	Not Used
10	32 K (Base 0 or base 32K only)
11	16K

For a 32K block bit, 15 is overridden; for a 16K block, both bits 15 and 14 are overridden.

As well as confining the Slave CPU, this relocates the Slave address. Thus, routines can be executed relative to zero no matter which base is specified.

5.3 UTILIZATION

Although the Slave CPU board was designed specifically for use in the TWIN system, it can be used in products of a general nature. External power is required as listed below. The board is designed to be plugged into a 100-pin printed circuit connector.

The following supply voltages are required:

1.	+5 V at	2.1	A	(with TWICE cable)
2.	+5 V at	1.3	A	(without TWICE cable)
3.	+12 V at	50	mA	
4.	-12 V at	50	mA	

The board should be mounted vertically with side supports to minimize vibration. Sufficient space should be left on both sides of the board to permit heat dissipation. Other components radiating large volumes of heat should not be located near the board.

Tables 5-1 through 5-3 show mother board, and TWICE I/O connections. Note that alternate ground lines on the TWICE cable provide good transmission line characteristics. A fuse near the top edge of the Slave module protects the +5 V line supplying current to the TWICE interface. This is a standard 1 amp fast blow fuse.

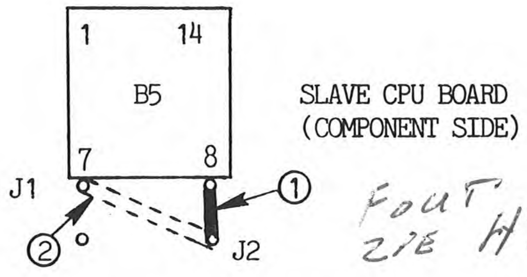
The following is a list of pertinent I/O characteristics for the board.

MOTHERBOARD LOGIC I/O LEVELS:	LOGIC 1	LOGIC 0
General Input	2.4V @ -.1mA	0.8V @ 1.6mA
General Output	2.4V @ -5.2mA	0.5V @ 48mA
A2-A7, IN	2.4V @ .1mA	0.8V @ 2.0mA
DO-D8 IN	2.4V @ .2mA	0.8V @ 3.6mA
INTO-INT 8, CPU INT, SLV PAUSE	2.4V @ .1mA	0.8V @ 4.1mA
UPAUSE OUT		0.7V @ 14mA

Delay characteristics of the TWICE interface assembly are listed in Table 5-4.

5.3.1 FLAG Line Output Gating

The FLAG output to the TWICE interface cable is normally gated by the USER I/O control signal on the Slave CPU card. In certain applications, this gating may produce undesired transitions on the FLAG output during Debug trace operations. The gating function can be removed by cutting a trace and adding a jumper as illustrated below.



FOUT
2/E HW001

Trace ① causes the FLAG output to be gated by the USER I/O control signal. To disable the gating function, cut trace ① and add jumper ②.

TABLE 5-1

PIN LIST, EDGE CONNECTOR P1

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
5-8	not used	
9	gnd	
10	gnd	
11-14	not used	
15	gnd	
16	gnd	
17	$\overline{\text{ADR 0}}$	Address Bit 0
18	$\overline{\text{ADR 1}}$	Address Bit 1
19	$\overline{\text{ADR 2}}$	Address Bit 2
20	$\overline{\text{ADR 3}}$	Address Bit 3
21	$\overline{\text{ADR 4}}$	Address Bit 4
22	$\overline{\text{ADR 5}}$	Address Bit 5
23	$\overline{\text{ADR 6}}$	Address Bit 6
24	$\overline{\text{ADR 7}}$	Address Bit 7
25	$\overline{\text{ADR 8}}$	Address Bit 8
26	$\overline{\text{ADR 9}}$	Address Bit 9
27	$\overline{\text{ADR 10}}$	Address Bit 10
28	$\overline{\text{ADR 11}}$	Address Bit 11
29	$\overline{\text{ADR 12}}$	Address Bit 12
30	$\overline{\text{ADR 13}}$	Address Bit 13
31	$\overline{\text{ADR 14}}$	Address Bit 14
32	$\overline{\text{ADR 15}}$	Address Bit 15
33	CMEM	Common Memory Enable
34	not used	
35	not used	
36	$\overline{\text{DATA 0}}$	Data Bit 0
37	$\overline{\text{DATA 1}}$	Data Bit 1
38	$\overline{\text{DATA 2}}$	Data Bit 2
39	$\overline{\text{DATA 3}}$	Data Bit 3
40	$\overline{\text{DATA 4}}$	Data Bit 4
41	$\overline{\text{DATA 5}}$	Data Bit 5
42	$\overline{\text{DATA 6}}$	Data Bit 6
43	$\overline{\text{DATA 7}}$	Data Bit 7
44-51	not used	

TABLE 5-1

PIN LIST, EDGE CONNECTOR P1 (continued)

Pin	Signal	Description
52	$\overline{\text{M}}$/IO	Memory /IO Enable Input
53	$\overline{\text{WRP}}$	Write Pulse Input
54	$\overline{\text{OPREQ}}$	Operation Request Input
55	R/$\overline{\text{W}}$	Read/Write Enable Input
56	$\overline{\text{HOLD}}$	Operation Acknowledge Input
57	$\overline{\text{JMP CMD}}$	Jump Command Input
58	$\overline{\text{RUN}}$	Run Condition Output
59	$\overline{\text{RESET}}$	Reset Input
60	$\overline{\text{JMP ACK}}$	Jump Acknowledge Output
61	$\overline{\text{INTA}}$	$\overline{\text{INTACK}}$ Output
62	$\overline{\text{INT 0}}$	Interrupt Level 0 Input
63	$\overline{\text{INT 1}}$	Interrupt Level 1 Input
64	$\overline{\text{INT 2}}$	Interrupt Level 2 Input
65	$\overline{\text{INT 3}}$	Interrupt Level 3 Input
66	$\overline{\text{INT 4}}$	Interrupt Level 4 Input
67	$\overline{\text{INT 5}}$	Interrupt Level 5 Input
68	$\overline{\text{INT 6}}$	Interrupt Level 6 Input
69	$\overline{\text{INT 7}}$	Interrupt Level 7 Input
70	$\overline{\text{SLV CPU INT}}$	Slave CPU Interrupt Input
71-74	not used	
75	$\overline{\text{SLV OPREQ}}$	Slave Operation Request Output
76	$\overline{\text{SLV RESET}}$	Slave Reset Input
77	$\overline{\text{SLV PSE}}$	Slave Pause Input
78-81	not used	
82	$\overline{\text{FETCH}}$	Fetch Operation Output
83	$\overline{\text{PAUSE}}$	CPU Pause Input
84	$\overline{\text{MAST RUN}}$	Master CPU Run Indication
85	$\overline{\text{SENSE}}$	CPU Sense Input
86	$\overline{\text{FLAG}}$	CPU Flag Output
87-94	not used	
95	2650 CLK	Slave CPU Clock Input
96	not used	
97	gnd	
98	gnd	
99	gnd	
100	gnd	

TABLE 5-2

PIN LIST, EDGE CONNECTOR P2

Pin	Signal	Description
1	$\overline{\text{UA 0}}$	User Address Bit 0
2	gnd	
3	$\overline{\text{UA 1}}$	User Address Bit 1
4	gnd	
5	$\overline{\text{UA 2}}$	User Address Bit 2
6	gnd	
7	$\overline{\text{UA 3}}$	User Address Bit 3
8	gnd	
9	$\overline{\text{UA 4}}$	User Address Bit 4
10	gnd	
11	$\overline{\text{UA 5}}$	User Address Bit 5
12	gnd	
13	$\overline{\text{UA 6}}$	User Address Bit 6
14	gnd	
15	$\overline{\text{UA 7}}$	User Address Bit 7
16	gnd	
17	$\overline{\text{UA 8}}$	User Address Bit 8
18	gnd	
19	$\overline{\text{UA 9}}$	User Address Bit 9
20	gnd	
21	$\overline{\text{UA 10}}$	User Address Bit 10
22	gnd	
23	$\overline{\text{UA 11}}$	User Address Bit 11
24	gnd	
25	$\overline{\text{UA 12}}$	User Address Bit 12
26	gnd	
27	$\overline{\text{UA 13}}$	User Address Bit 13
28	gnd	
29	$\overline{\text{UA 14}}$	User Address Bit 14
30	gnd	
31	U $\overline{\text{RUN}}$	User Run Indication
32	gnd	
33	U R/$\overline{\text{W}}$	User Read/Write Indication
34	gnd	
35	U $\overline{\text{M}}$/IO	User Memory/IO Indication
36	gnd	
37	U $\overline{\text{WRP}}$	User Write Pulse Output
38	gnd	
39	U $\overline{\text{OPREQ}}$	User Operation Request Output
40	gnd	

TABLE 5-3

PIN LIST, EDGE CONNECTOR P3

Pin	Signal	Descripton
1	$\overline{\text{UD 0}}$	User Data Bit 0
2	gnd	
3	$\overline{\text{UD 1}}$	User Data Bit 1
4	gnd	
5	$\overline{\text{UD 2}}$	User Data Bit 2
6	gnd	
7	$\overline{\text{UD 3}}$	User Data Bit 3
8	gnd	
9	$\overline{\text{UD 4}}$	User Data Bit 4
10	gnd	
11	$\overline{\text{UD 5}}$	User Data Bit 5
12	gnd	
13	$\overline{\text{UD 6}}$	User Data Bit 6
14	gnd	
15	$\overline{\text{UD 7}}$	User Data Bit 7
16	gnd	
17	U $\overline{\text{HOLD}}$	User Hold Input
18	gnd	
19	U $\overline{\text{PAUSE}}$	User Pause Input
20	gnd	
21	U $\overline{\text{RESET}}$	User Reset Input
22	gnd	
23	U $\overline{\text{INTREQ}}$	User Interrupt Request Input
24	gnd	
25	U $\overline{\text{SENSE}}$	User Sense Input
26	gnd	
27	U CLK	User Clock Input
29-36	not used	
37-40	+5 V	User Logic Power Output

TABLE 5-4
TWICE INTERFACE CHARACTERISTICS

INTERFACE PROPAGATION DELAY

Includes all delay from/to processor chip and 40-pin socket.

CONTROL TO 2650 (in nanoseconds)

	TYP	MAY
$\overline{\text{OPACK}}$	55	84
$\overline{\text{PAUSE}}$	76	121
RESET	66	98
$\overline{\text{INTREQ}}$	66	98
SENSE	46	61
CLOCK	68	118

CONTROL FROM 2650

OPREQ	57	84
RUN	83/173*	127/213*
M/$\overline{\text{IO}}$	57	84
FLAG	57	84
$\overline{\text{R}}$/W	57	84
INTACK	58	83
WRP	37	47
DATA IN/OUT	43	56
ADDRESS	43	56

*RUN TRAILING EDGE DELAYED ONE OR TWO 10MHZ CLOCK CYCLES

INTERFACE LEVELS	LOGIC 0	LOGIC 1
CONTROL TO 2650	0.4 mA @ 0.8V MAX	40.0 UA @ 2.0V MIN
CONTROL FROM 2650	16.0 mA @ 0.4V MAX	0.8 mA @ 2.4V MIN
DATA IN	0.4 mA @ 0.8V MAX	80.0 UA @ 2.0V MIN
DATA/ADDR OUT	48.0 mA @ 0.5V MAX	5.2 mA @ 24.V MIN

CHAPTER 6

RAM MEMORY

6.0 INTRODUCTION

The RAM Memory module provides RAM storage for the Master or Slave CPU. Each module is capable of storing up to 4K eight-bit bytes of data.

Each module has a switch-selected base address, allowing unique selection of modules in the system.

Communications with the CPU is accomplished through the system data bus, which is driven by a set of drivers and receivers for the RAM array. The drivers and receivers are enabled only when the board is accessed.

Memory is enabled when the CPU is executing an instruction requiring a read from, or write to memory. During each CPU cycle, control signals are active to control transfer of data to or from the data bus.

6.1 OVERALL BLOCK DIAGRAM

A block diagram of the board is shown in Figure 6-1. There are four major blocks: RAM, Address Logic, Operation Control, and Timing Logic. The following paragraphs discuss the function of each of these blocks in typical memory read and write operations.

6.1.1 RAM Read

At the beginning of a CPU read cycle, addresses are output to the system address bus by the CPU. At approximately the same time, control signals become active to determine the direction of data flow on the data bus and to enable modules which will transmit or receive the data.

Address lines are input to the RAM Memory board from the system bus. The upper four address bits (A12-15) are used to select one of the memory boards in the system. This is done by comparing the four address bits with the board address selected by four board mounted switches. If the address compares, RAM MODULE SELECT is generated and the read sequence begins. The other conditions which must be met to output data are that $\overline{\text{OPREQ}}$ must be active, R/$\overline{\text{W}}$ must be high, and

$\overline{\text{MEM}}$/IO must be low (i.e., a memory read operation must be specified by the CPU). If all these conditions are met, RAM OUTPUT ENABLE is active, and drivers output the data from the RAM array to the data bus. A signal derived from $\overline{\text{OPREQ}}$ and $\overline{\text{MEM}}$/IO triggers a one-shot, which outputs a 650 nanosecond pulse (RAM HOLD), which is output to the system bus as $\overline{\text{OPACK}}$ and holds the CPU until the data on the bus is stable.

6.1.2 RAM Write

Writing to the RAM array involves the same addressing sequence as a read. In the case of RAM write operation, however, the signal R/W must be low. A write operation triggers a one-shot and generates a 400 ns RAM WRITE pulse to strobe the data into the selected row of devices. At the same time, another one-shot is triggered to generate the $\overline{\text{OPACK}}$ signal to the CPU.

6.2 DETAILED FUNCTIONAL DESCRIPTION

Referring to Figure 6-1, the following describes the four major functional blocks.

6.2.1 Memory Organization

The RAM array is composed of 32 type 2102-2 memory devices, each capable of storing 1024 bits. These are organized into a four row by eight column array, for a total capacity of 4096 eight-bit words.

6.2.2 Memory Addressing

The board contains a group of four switches which allow the base address of the RAM array to be placed at any multiple of 4K between 0 and 60K.

The sixteen address bus bits (A0-A15) are divided into three groups to address the RAM array. The four most significant bits (A12-A15) are applied to the RAM base address comparator. The other inputs to the comparator are the four base address programming switches, the CMEM bus signal, and a CMEM programming jumper. This jumper allows the board to be used in either Master or Slave memory portions of the portions of the system. The output of the RAM base address comparator output enables the RAM read and write circuits. Lines A10 and A11 are decoded to select one of the four "rows" of the RAM array, while A0-A9 select the proper bit within the selected row.

6.2.3 Timing

Memory is enabled when the CPU is executing an instruction requiring a read or write involving memory. During the CPU cycle, control signals become active to control transfer of data to or from memory. $\overline{\text{OPREQ}}$ alerts memory and I/O units that a data transfer is in the offing. $\overline{\text{MEM}}$/IO signifies that either a memory of I/O unit is to be activated, and R/$\overline{\text{W}}$ determines whether the requested operation is a read or write.

On the board, $\overline{\text{MEM}}$/IO and $\overline{\text{OPREQ}}$ are ANDed to generate the basic memory timing signal. If RAM is addressed, and a read operation is requested, the RAM data output drivers are enabled to place RAM data on the data bus. If the operation is a write, a 400 ns write pulse is generated and applied to the RAM write circuits to write the data bus contents into the RAM array. This write pulse occurs after a 200 ns delay to allow settling of the address and data buses. In either case, a 650 ns HOLD signal is generated to hold the CPU until the operation is complete.

6.3 UTILIZATION

The RAM Memory board may be used as a general-purpose storage unit. Table 6-1 lists the required input/output lines; all levels are TTL-compatible. The unit is designed to be mounted vertically in a 100-pin printed circuit connector. Guides should be provided on both vertical edges to prevent movement.

Exposure to dust and vibration should be avoided for reliable operation. The board should not be mounted near other components which radiate heat toward the Memory board.

The board requires the following DC voltage:

+5 VDC at 1.2 A

6.3.1 Switches and Jumpers

The RAM Memory Board contains several switches and jumpers which provide user control over the operation of the board. Jumpers disabling RAM hold circuits must be cut when the board is used with different CPU devices.

6.3.2 RAM Base Address Selection

Switch E8A is used to set the base address of the 4K RAM. This may be set at any 4K multiple between 0 and 60K. The switch weights and sample settings for a a base address of 20,480 are as follows:

SWITCH	WEIGHT	SETTING FOR BASE ADDRESS OF 20,480(4096x5)
E8A-1	4,096	ON
E8A-2	8,192	OFF
E8A-3	16,384	ON
E8A-4	32,768	OFF

6.3.3 Other Jumpers

J2 determines whether the RAM is selected when the CMEM bus signal is low or when it is high. J3 can be used to disable the RAM completely. J7 does the same for the 650 millisecond RAM hold signal.

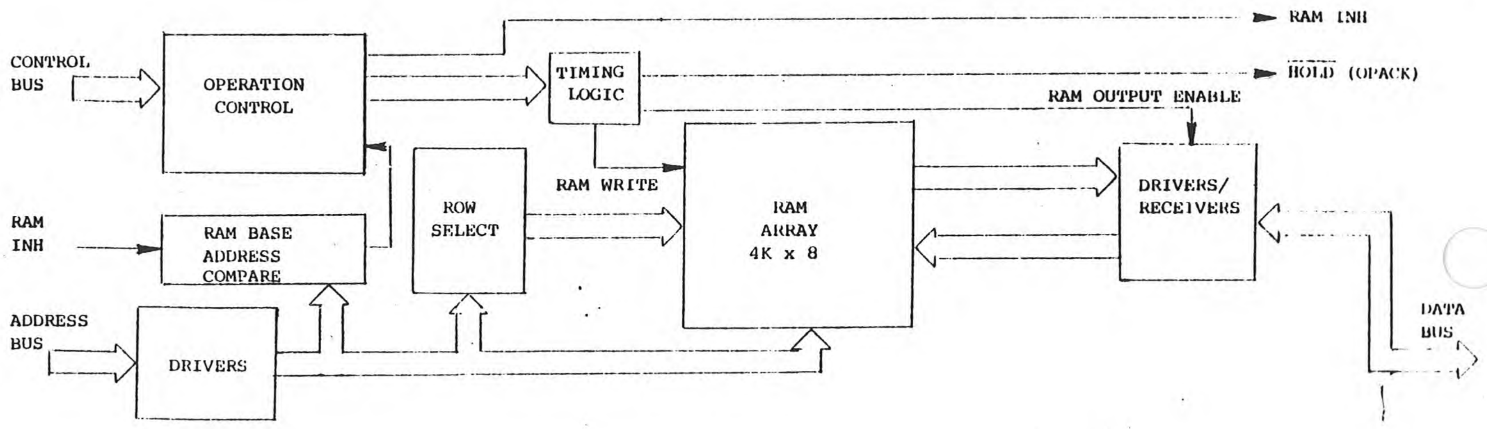


FIGURE 6-1 RAM

MEMORY BLOCK DIAGRAM

TABLE 6-1

PIN LIST, RAM MEMORY BOARD

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
5-14	not used	
15	gnd	
16	gnd	
17	$\overline{\text{ADR0}}$	Address Bit 0
18	$\overline{\text{ADR1}}$	Address Bit 1
19	$\overline{\text{ADR2}}$	Address Bit 2
20	$\overline{\text{ADR3}}$	Address Bit 3
21	$\overline{\text{ADR4}}$	Address Bit 4
22	$\overline{\text{ADR5}}$	Address Bit 5
23	$\overline{\text{ADR6}}$	Address Bit 6
24	$\overline{\text{ADR7}}$	Address Bit 7
25	$\overline{\text{ADR8}}$	Address Bit 8
26	$\overline{\text{ADR9}}$	Address Bit 9
27	$\overline{\text{ADR10}}$	Address Bit 10
28	$\overline{\text{ADR11}}$	Address Bit 11
29	$\overline{\text{ADR12}}$	Address Bit 12
30	$\overline{\text{ADR13}}$	Address Bit 13
31	$\overline{\text{ADR14}}$	Address Bit 14
32	$\overline{\text{ADR15}}$	Address Bit 15
33	CMEM	Slave memory select input
34	$\overline{\text{RAMINH}}$	RAM inhibit output
35	not used	
36	$\overline{\text{DATA 0}}$	Data Bit 0
37	$\overline{\text{DATA 1}}$	Data Bit 1
38	$\overline{\text{DATA 2}}$	Data Bit 2
39	$\overline{\text{DATA 3}}$	Data Bit 3
40	$\overline{\text{DATA 4}}$	Data Bit 4
41	$\overline{\text{DATA 5}}$	Data Bit 5
42	$\overline{\text{DATA 6}}$	Data Bit 6
43	$\overline{\text{DATA 7}}$	Data Bit 7
44-50	not used	

TABLE 6-1

PIN LIST, RAM MEMORY (continued)

Pin	Signal	Description
51	not used	
52	$\overline{M}$/IO	Memory/IO Enable
53	not used	
54	$\overline{OPREQ}$	Operation Request
55	R/$\overline{W}$	Read/Write Command
56	$\overline{HOLD}$	Hold Command to CPU
57	not used	
58	not used	
59	$\overline{RESET}$	System Reset Input
60-97	not used	
98	gnd	
99	gnd	
100	gnd	

CHAPTER 7

GENERAL-PURPOSE I/O BOARD

7.0 INTRODUCTION

The General-Purpose I/O Board provides a means of interfacing a processor (either Master or Slave) to external data devices. These include CRT terminals, teletypewriters, high-speed paper tape readers, or any 8-bit parallel device. The board includes the necessary hardware to convert serial inputs to the parallel data required by the processor, and includes teletype reader control. Status registers allow the system to monitor data I/O status and take appropriate action when necessary. Six input and output ports are included. The board also includes up to 16 levels of interrupts to service the requirements of peripherals connected to the board. Refer to Figure 7-1.

Four edge connectors are provided at the top of the board for connection to external peripheral devices. One of the connectors (P2) is dedicated to the RS-232-C/TTY port. Connector P3 is dedicated to I/O ports 0 and 1. Connectors P4 and P5 are unassigned and can be configured for custom interface connections to external peripherals. Wire-wrap pins may be installed in holes provided in the board and connections may be made by the user to I/O ports 2 and 3 or to the interrupt input circuitry.

7.1 GENERAL DESCRIPTION

Figure 7-2 is a data flow block diagram of the board.

Interrupt circuits are provided for up to 16 devices when the General-Purpose I/O board is used with the Master CPU, and for up to eight devices when the board is used with the Slave CPU.

A board-mounted DIP switch selects board address, Master/Slave select, and Baud rate. The two choices for Baud rate are Low (110 Baud) and High. The high Baud rate may be jumper-selected for 150, 300, 600, or 1200 Baud. Each Baud rate is generated by dividing down the I/O clock from the Master CPU board.

Interface to the RS-232-C and TTY devices is through serial I/O ports and a UART device. The UART performs serial-to-parallel and parallel-to-serial conversion functions and provides buffer storage for the input and output characters transmitted to and from the I/O board. Reader control for a TTY paper tape reader is also provided.

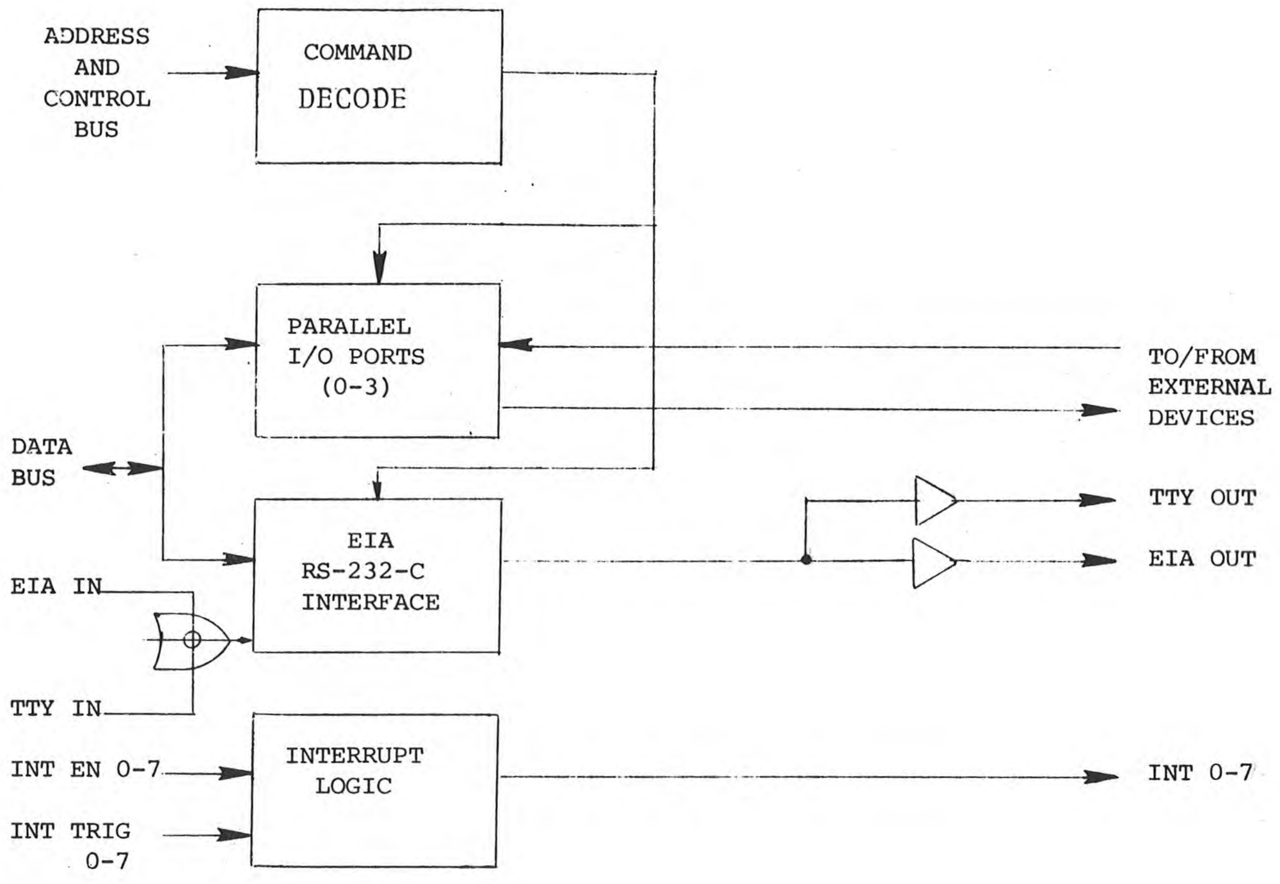


FIGURE 7-1

SIMPLIFIED BLOCK DIAGRAM

RS-232-C mark and space inputs are ORed with the TTY current loop input and connected to the serial input of the UART. In the same way, mark and space outputs for both RS-232-C and TTY are fanned out to two separate outputs (TTY XMIT and EIA XMIT DATA).

UART status is transmitted to the common I/O port bus, either from the UART itself or from external DATA SET READY, CARRIER DETECT, or RING indications.

7.2 DETAILED DESCRIPTIONS

This section provides detailed information on how the General-Purpose I/O board performs its communications functions.

The board can be divided into major blocks for purposes of discussion:

1) Input of serial data and conversion to parallel data for output to the system data bus.

2) Input of parallel data and conversion to serial data for output to the TTY or RS-232-C interfaces.

3) Input of parallel data from input ports and transmission of data to the system data bus.

4) Output of parallel data from the system data bus to the output ports.

Auxiliary functions include Baud rate generation, interrupt logic, and control/status ports for the RS-232-C interface.

7.2.1 Serial I/O Ports

The serial I/O port is a full EIA (RS-232-C) communications interface. This port is utilized for EIA and TTY operation. Refer to the shaded portion of Figure 7-3.

Parallel-to-serial transmit operation and serial-to-parallel receive operation are performed by a UART (universal asynchronous receiver/transmitter). The eight transmit data bits DB1-DB8 are tied to the board's internal output data bus. The eight receive data bits (RD1-RD8)are tied to the internal input data bus. The internal data bus is connected to the system data bus via tri-state drivers and receivers.

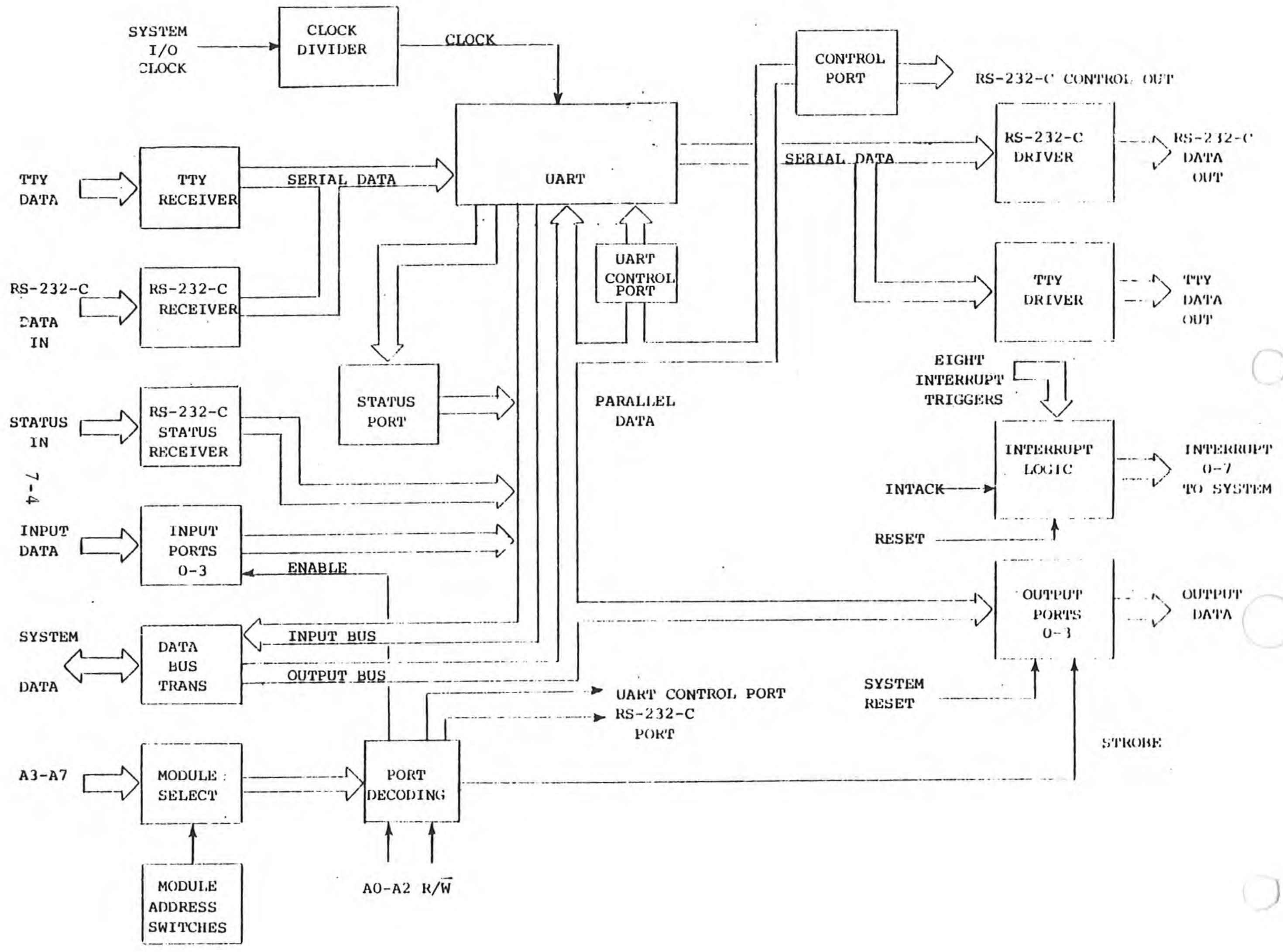


FIGURE 7-2

OVERALL BLOCK DIAGRAM

UART initialization is accomplished during system reset. Transmit Buffer Empty, End of Character, and Serial Out are set true and Data Available is set false during this operation. Transmit and receive parameters such as Bits/Character, Odd/Even Parity, No Parity, and Number of Stop Bits are entered into the UART's internal registers via a write operation to the device. The device is hard-wired for seven bits per character. The No Parity option is defeated; odd or even parity is under program control. Refer to paragraph 7.4 for a description of the byte format. The number of stop bits is determined by the Baud rate and is set by switch E1. The UART monitors the number of stop bits and parity and compares these to the previously programmed control bits.

Transmit operation is initiated by first checking the status register for a Transmit Buffer Empty indication. Data is then strobed into the internal UART data register by a Write Port 7 instruction. Transmit Buffer Empty will then go to a logic "0" at this time and the contents of the data buffer will be loaded into a UART shift register. Completion of the transfer is signified by Serial Out and End of Character both going to a logic "0". Serial transmission starts and Transmit Buffer Empty goes true immediately thereafter, generating Interrupt 7 which signifies that the next data byte may be loaded into the data register. Completion of transmission is indicated by End of Character going true.

Receive operation begins with verification of a valid start bit at the serial data input to the UART. Serial data, Parity, and Stop bits are now received and monitored. Stop bit framing is checked and latched internally at this time. Internal logic checks Data Available. If Data Available is true, the Overrun bit is set. Received data will then be transferred from the serial input register to the received data holding register. Completion of this transfer is indicated by Data Available going true which generates Interrupt 6. Receive status can be checked with a read of port 6. The next character can now be loaded into the serial input register, with one full character time allowed to remove the previous character from the holding register. Read port 7 generates a signal which transfers data from the holding register to the system data bus for processing. This action also resets Data Available which indicates completion of the read data operation. External EIA receive status is converted from EIA levels to TTL levels by use of EIA line receivers. These are connected to the internal data bus via tri-state drivers which are enabled by a read port 5 command.

Various Baud rates are available for serial I/O operation. Switch E1 is used to set high or low Baud rate. It also sets the number of stop bits used: one bit for high speed operations and two bits for low speed operation. In high speed mode, the rate may be jumper-selected for 150, 300, 600, and 1200 Baud. In low speed mode, the Baud rate is 110. This is used for TTY operation.

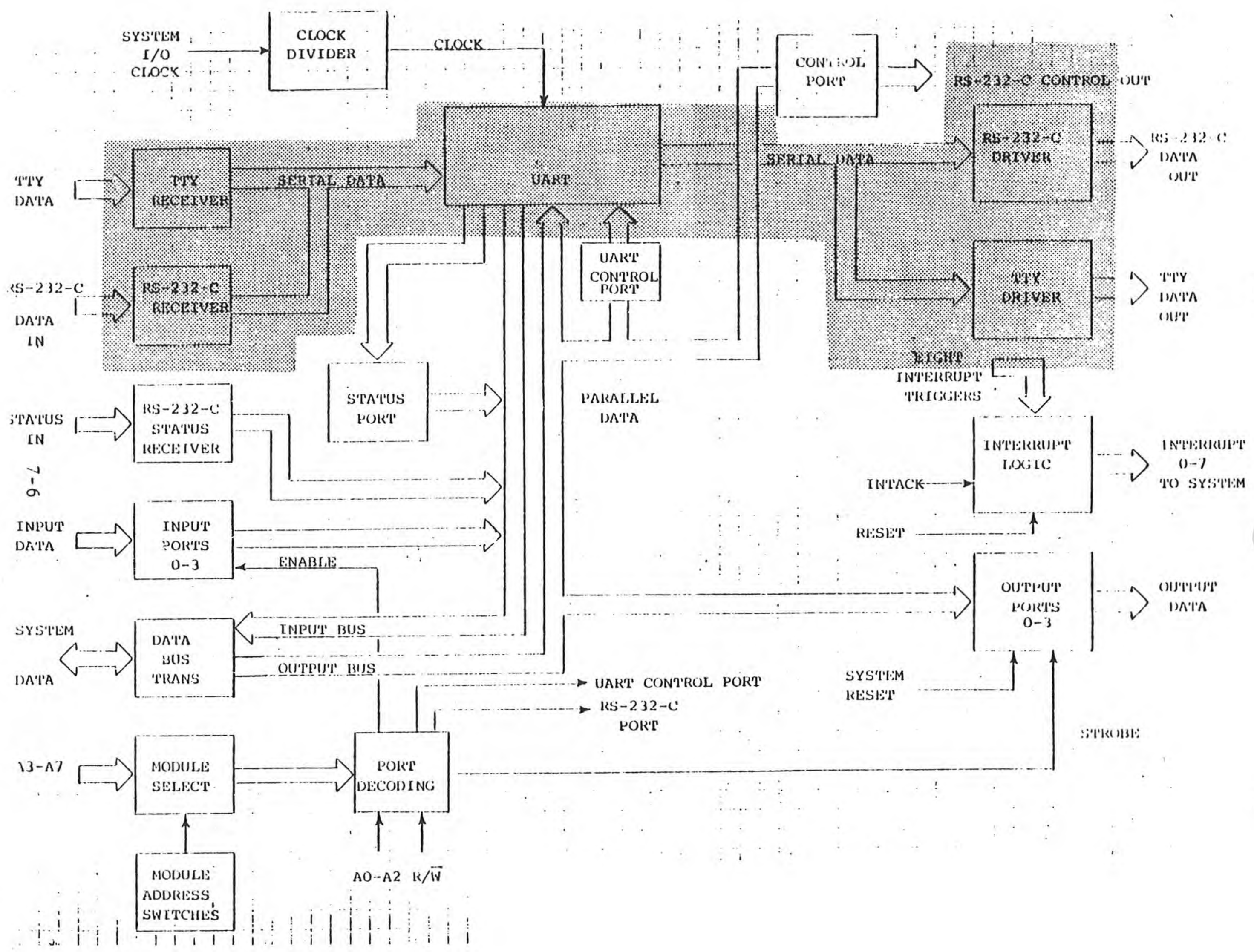


FIGURE 7-3

TTY AND RS-232-C LOGIC

7.2.2 Address Control

Because there may be more than one I/O module in the system, a means of selecting individual modules is required. This logic block is shown in the shaded portion of Figure 7-4. Address control is responsible for module address decoding, port decoding, Master or Slave selection, and read or write functions.

Module address selection is accomplished by comparison of the board-mounted address switch settings and the contents of the system address bus. Module address is predetermined by system configuration and set with the address select switch located at E1 on the printed circuit board. Module select logic also determines Master or Slave operation and is done by comparison with the MSTR RUN signal from the system bus. Two three-line to eight-line decoders are used to determine the address of the port. One decoder is used for read port operation and the other for write port operation. Module Select ANDed with Read or Write is used to enable the appropriate decoding of address bits A0-A2 to determine which port is to be used. Address bit 13 (extended I/O) in conjunction with an I/O command and OPREQ enables the Module Select line for transmission to the decoders. In addition, the Write Decoder uses the CPU write pulse (WRP) as a strobe signal.

7.2.3 Input and Output Ports

There are four sets of independently addressable, general-purpose input and output ports on the board. The input ports are simply drivers which are enabled by control lines from the Read Decode discussed previously. These are shown in the shaded portion of Figure 7-5. The driver outputs from each port are connected directly to the board's internal data bus.

Output ports are shown in the shaded portion of Figure 7-5. The ports consist of an 8-bit register made up of a 6 bit register and two flip-flops. The data from the data bus is latched into the register as each port is addressed by lines from the decoder and strobed by the decoder strobe. The output of the registers go to line drivers which are connected to edge connector P3 or brought out to pads on the board which may be connected to edge connectors P4 and P5.

7.2.4 Interrupt Logic

Interrupt logic is shown in the shaded portion of Figure 7-6. There are eight identical circuits, consisting of a gate and a flip-flop. Each interrupt circuit is enabled by an external line, and triggered by a separate external line. This sets the latch and outputs the interrupt request to the system bus. When the interrupt has been acknowledged, a decoder decodes the contents of the

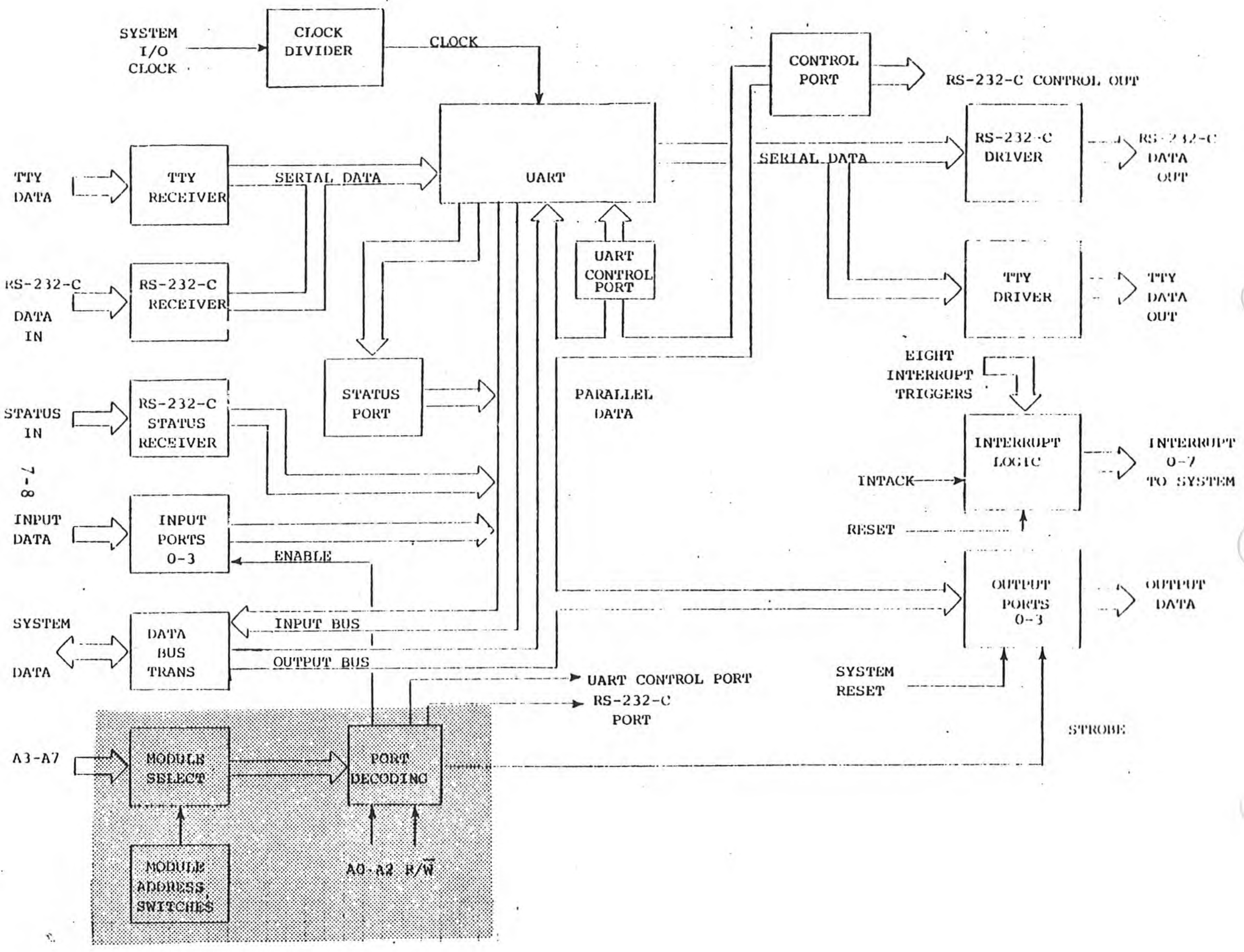


FIGURE 7-4

ADDRESS DECODING

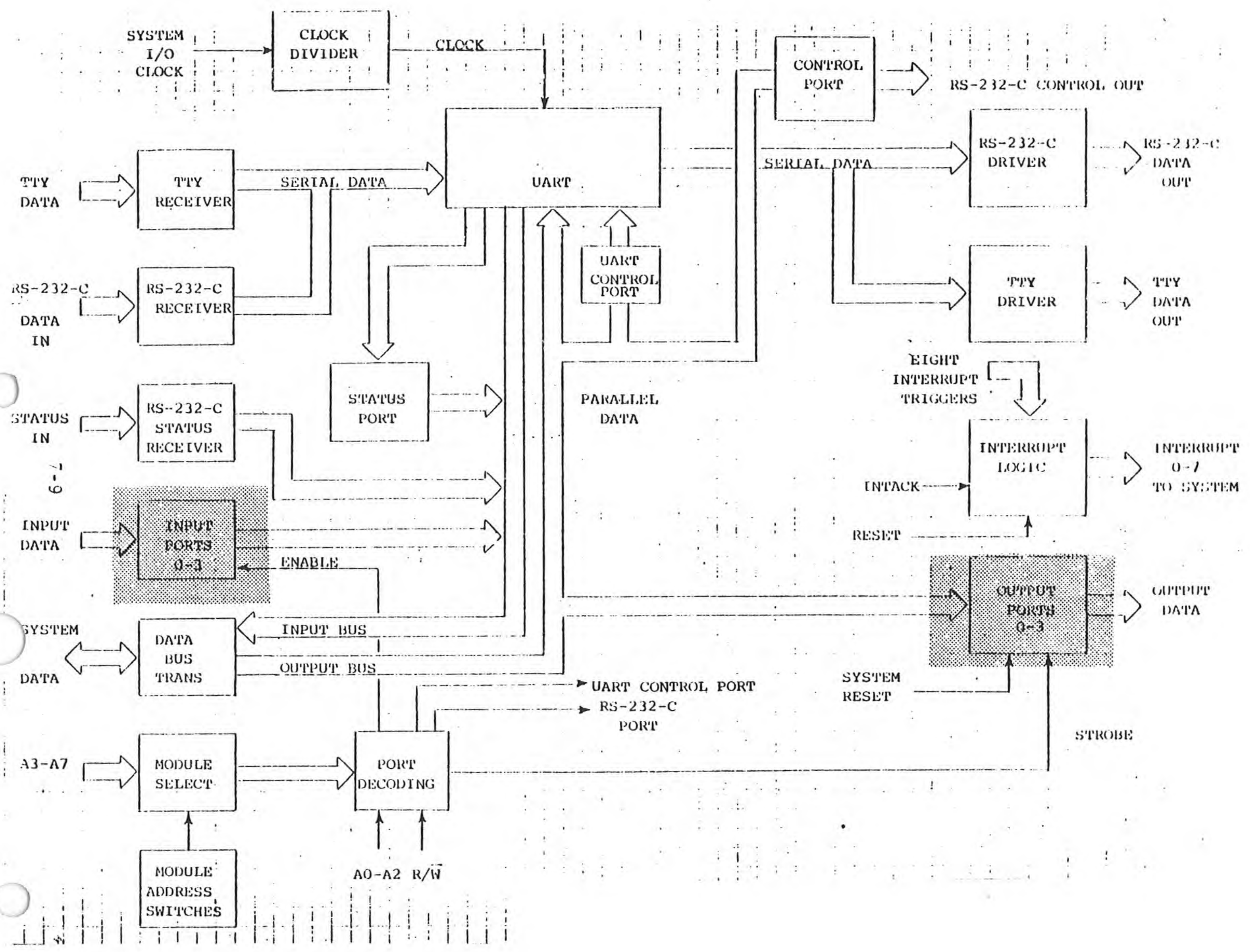


FIGURE 7-5

I/O PORTS

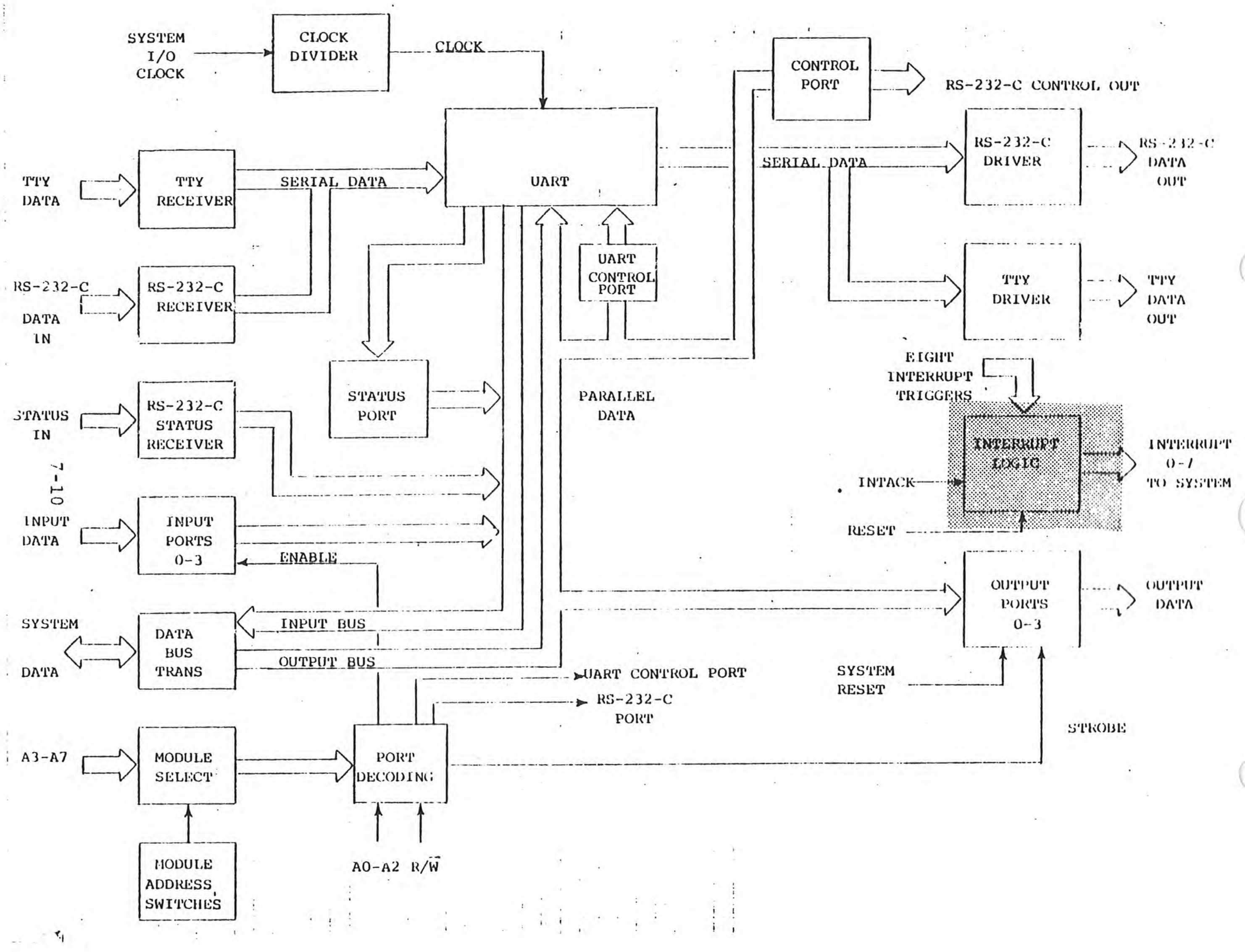


FIGURE 7-6

INTERRUPT LOGIC

data bus to determine which interrupt circuit requested the interrupt. When $\overline{\text{INTACK}}$ goes low, a reset output resets the interrupt latch circuit which requested the interrupt. The master system reset also clears any interrupts pending.

7.3 UTILIZATION

7.3.1 Switch Settings

There is a single DIP switch mounted on the General-Purpose I/O board. This switch selects module address and Baud rate. The upper five switch sections select module address; the lower two switches select Baud rate. One section is not used on boards with part number 9001211C and above. The table below shows settings for address, Master/Slave, and Baud rate.

Section	P/N 90012011B and below	P/N 90012011C and above
1	A5	A3
2	A3	A4
3	A7	A5
4	A4	A6
5	A6	A7
6	mstr/slv	not used
7	high/110	mstr/slv
8	high/110	high/110

Note: In the table above "A5" refers to address line 5, "A3" refers to address line 3, etc. The board can be switched for any combination of the five lines, and this address is used for bits 3-7 in the extended read or write command in the user program. To switch in an address line, move the switch section to the "on" position. In the Master/Slave and Baud rate sections, the "on" position is listed second. Addresses F2-FF are reserved for SVC's.

7.3.2 Jumpers

There are three possibilities for jumper installation on the board. Edge connectors P4 and P5 are not assigned to any I/O port. These can be wired by installing wire-wrap pins in the holes provided and wiring to sockets mounted in board grid locations A6 (Port 0), A7 (Port 1), A8 (interrupts), A9 (Port 2), and A10 (Port 3). The second possibility is wiring the board for interrupt levels 8-15 by cutting the board traces for levels 0-7 and wiring to the proper holes in the board. The third possibility is selection of high Baud rate. Table 7-1 shows jumper options and Baud rate jumper placement.

TABLE 7-1

JUMPER OPTIONS

<u>Interrupt Inputs</u>

J1
J2
J3
J4

<u>Interrupt Trigger Inputs</u>

J5	Trig	- 7
J6		- 5
J7		- 1
J8		- 4
J9		- 6
J10		- 2
J11		- 3
J12		- 0

<u>Interrupt Resets</u>

$$\begin{tikzcd} J13 \arrow[d] & \text{- Reset} & 8 \arrow[d] \\ J20 & \text{- Reset} & 15 \end{tikzcd}$$

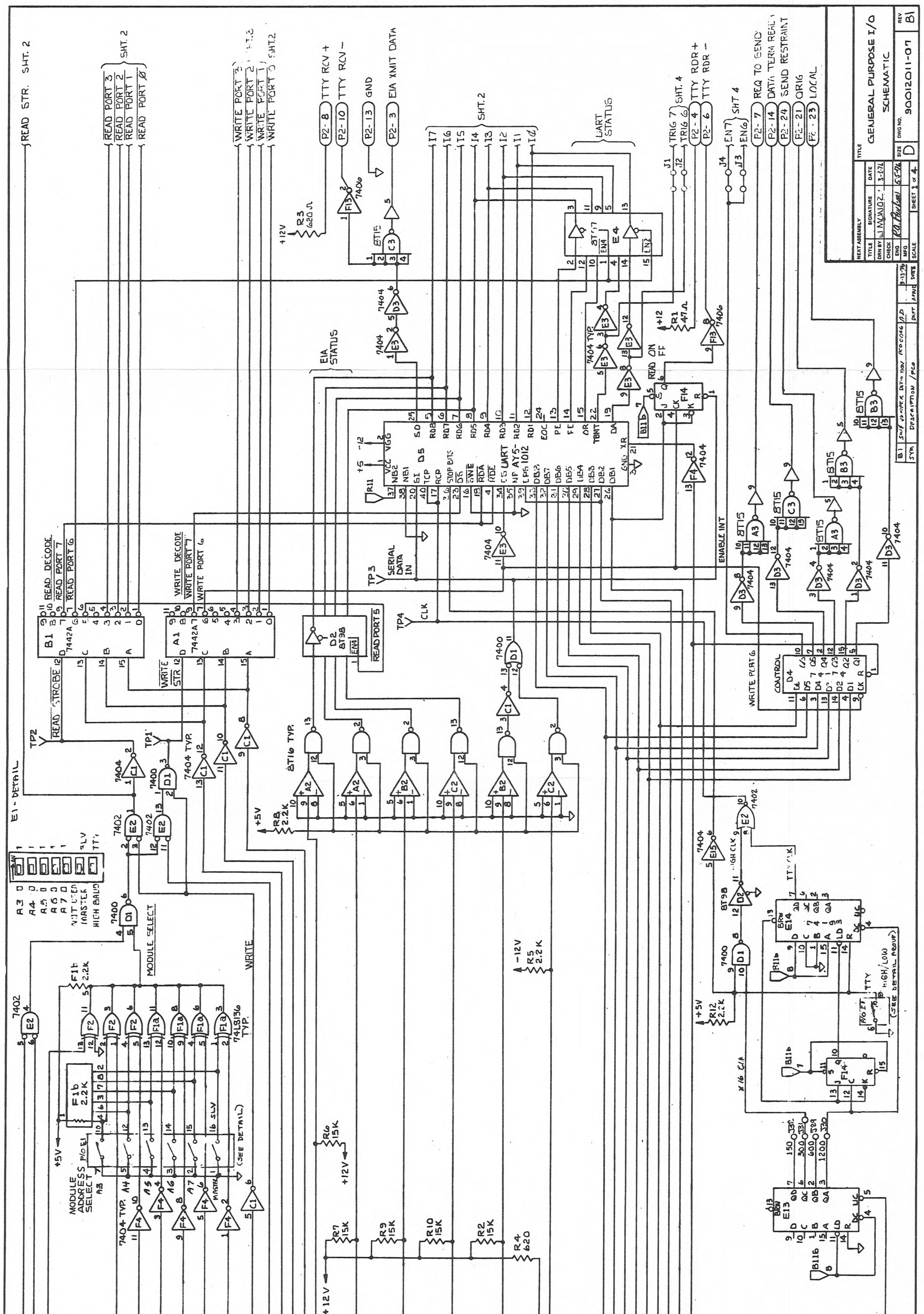
$$\begin{tikzcd} J21 \arrow[d] & \text{- Reset} & 0 \arrow[d] \\ J28 & \text{- Reset} & 7 \end{tikzcd}$$

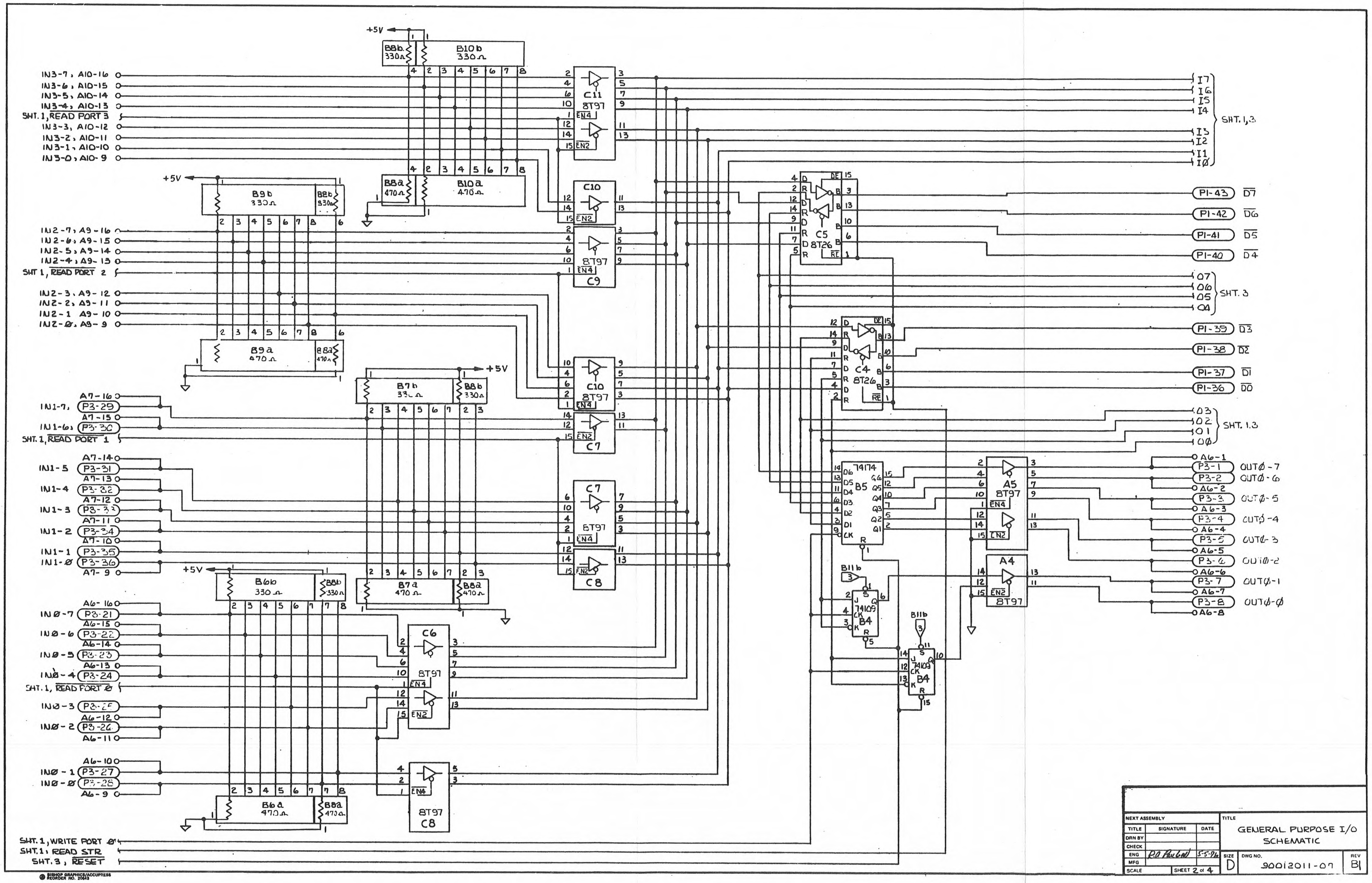
<u>Interrupt Outputs</u>

$$\begin{tikzcd} J33 \arrow[d] & \text{- INT} & 0 \arrow[d] \\ J48 & \text{- INT} & 16 \end{tikzcd}$$

EIA Baud Rate Jumpering

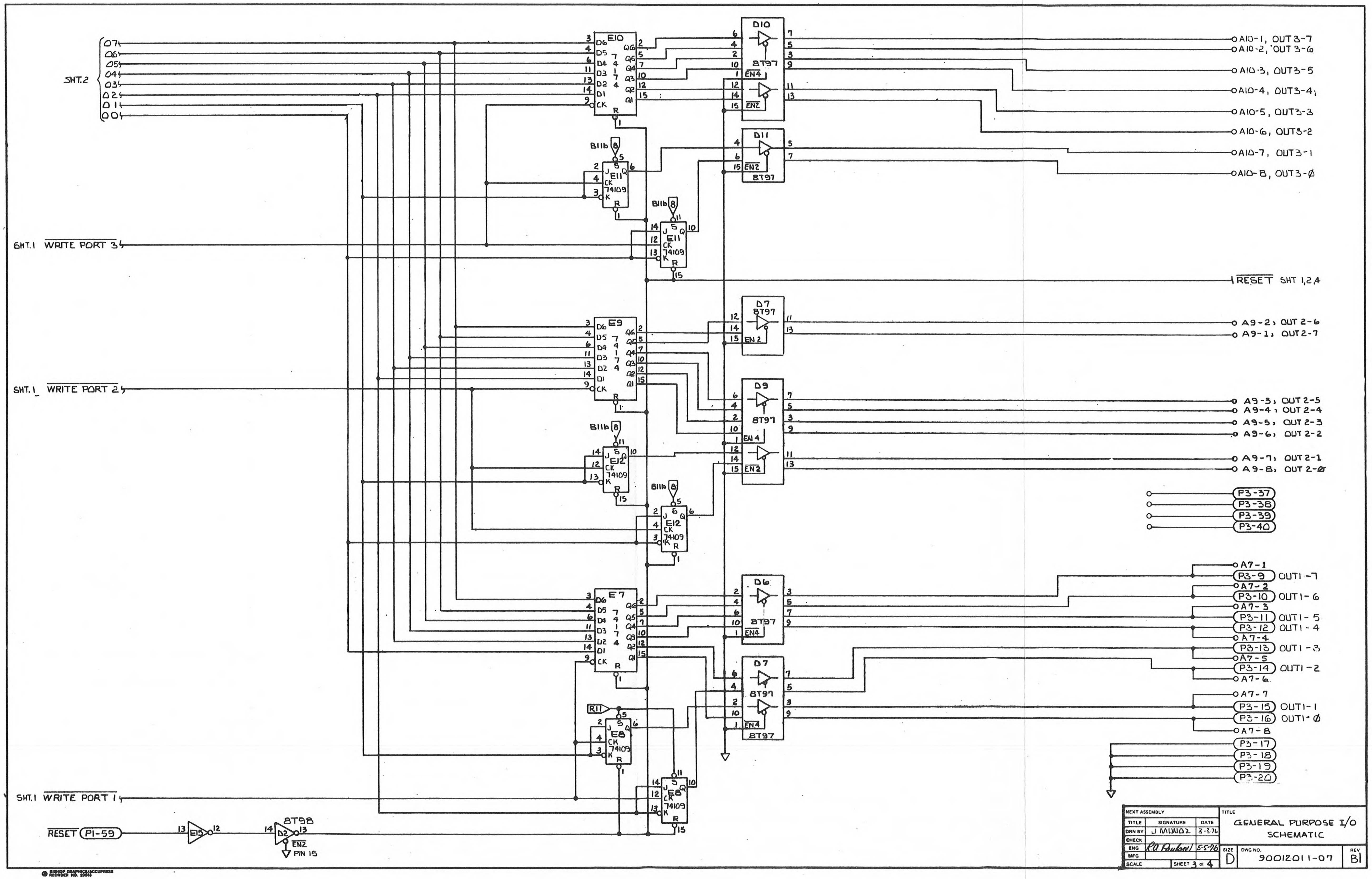
$$\begin{tikzcd} \text{600 (J29)} \arrow[ddr, dashed] & \\ \text{1200 (J30)} \arrow[dr, dashed] & \\ \text{300 (J31)} \arrow[r, dashed] & \text{Output} \\ \text{150 (J32)} \arrow[ur, dashed] & \end{tikzcd}$$


GENERAL PURPOSE I/O SCHEMATIC
DWG NO. 90012011-07
REV B1
SIZE D
SHEET 1 of 4
NEXT ASSEMBLY
READ STR. SHT. 2
READ PORT 3
READ PORT 2
READ PORT 1
READ PORT Ø
SHT. 2
WRITE PORT 3
WRITE PORT 2
WRITE PORT 1
WRITE PORT Ø
TTY RCV +
TTY RCV −
GND
EIA XMIT DATA
LART STATUS
EIA STATUS
TTY RDR +
TTY RDR −
SHT. 4
REQ TO SEND
DATA TERM READY
SEND RESTRAINT
ORIG
LOCAL
READ DECODE
READ PORT 7
READ PORT 6
WRITE DECODE
WRITE PORT 7
WRITE PORT 6
READ STROBE
WRITE STR
MODULE SELECT
WRITE
E1 - DETAIL
MODULE ADDRESS SELECT
SERIAL DATA IN
CLK
READ PORT 5
LART AY5-1012
CONTROL
WRITE PORT 6
ENABLE INT
READ ON FF
HIGH CLK
TTY CLK
HIGH/LOW
X 16 CLK
A13 OPREQ M/IO
MSTR RUN
R/W
WRP
CLEAR TO SEND
DATA SET READY
CARRIER DETECT
RING INDICATOR
EIA RCVD DATA
TTY XMIT +
TTY XMIT −
SHT. 3 RESET
I/O CLOCK
150 300 600 1200



NEXT ASSEMBLY			TITLE
TITLE	SIGNATURE	DATE	GENERAL PURPOSE I/O SCHEMATIC
DRN BY			
CHECK			
ENG	[illegible]	5-5-76	SIZE D — DWG NO. 90012011-07 — REV B1
MFG			
SCALE		SHEET 2 of 4	

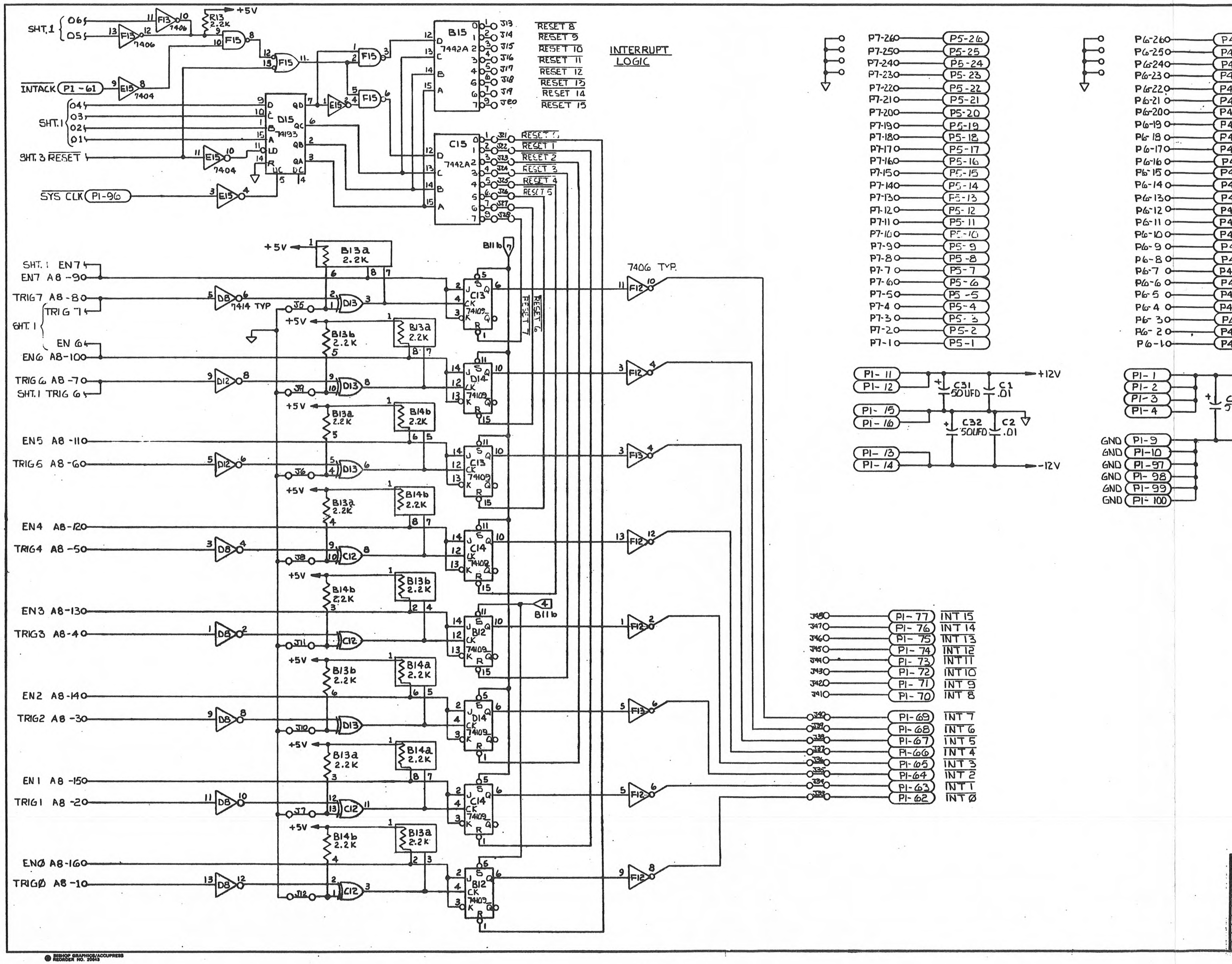
BISHOP GRAPHICS/ACCUPRESS REORDER NO. 20643



NEXT ASSEMBLY		
TITLE	SIGNATURE	DATE
DRN BY	J MUNOZ	3-3-76
CHECK		
ENG	[illegible]	5-5-76
MFG		
SCALE	SHEET 3 of 4	

TITLE: GENERAL PURPOSE I/O SCHEMATIC

SIZE D DWG NO. 90012011-07 REV B1



B	PCO.0017		CB.	5-5-76
A	PCO.0016		CB	
SYM	DESCRIPTION/PCO		SIGNATURE DRFT	APPVD DATE

NEXT ASSEMBLY

TITLE	SIGNATURE	DATE
DRN BY	J. MUNOZ	3-3-76
CHECK		
ENG	[illegible]	5-5-76
MFG		
SCALE	SHEET 4 of 4	

TITLE GENERAL PURPOSE I/O SCHEMATIC

SIZE D DWG NO. 90012011-07 REV B1

BISHOP GRAPHICS/ACCUPRESS REORDER NO. 20043

7.4 SOFTWARE SUPPORT

Because the General-Purpose I/O board is not supported by the operating system, the user must write programs that address the I/O ports, the TTY interface, and the RS-232-C interface as system devices. The following describes general guidelines for inputting or outputting data and status to the interfaces.

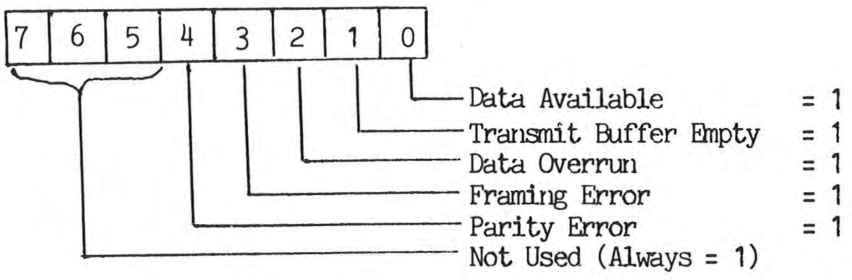
7.4.1 TTY INTERFACE

TTY DEVICE ASSIGNMENT
The TTY requires two device numbers in addition to the switch-selected board address. The devices are assigned and used in the following manner:

Control Device	Port 6 (plus board address)
Data Device	Port 7 (plus board address)

TTY COMMANDS
There are four commands available to the TTY interface, described as follows:
1) REQUEST STATUS COMMAND

The TTY control device has an I/O port where status information is received. TTY status can be read at any time. The status received is a one-byte value that has the following definition:



DATA AVAILABLE. A character of input data is available in the input data port. The bit is reset when the data byte is read from the input data port.

TRANSMIT BUFFER EMPTY. The transmit data port is empty and ready to take a data character for output. The bit is reset when an output data byte is sent to the output data port.

DATA OVERRUN. The data available in the input data port was not read prior to a new input data character being received. The bit is reset when the data available bit is cleared.

FRAMING ERROR. The received data character does not have a valid stop bit. The error is reset when a valid character and stop bit are received.

PARITY ERROR. The received data character had an odd bit count. The bit count should be even. The condition is cleared when a valid even parity character is received.

To read the TTY status a Read Extended instruction,

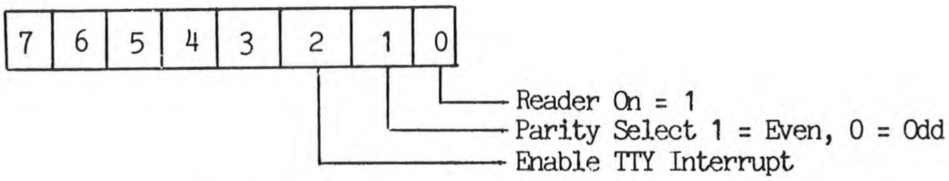
REDE,r V

is issued, where: r = the 2650 general purpose register to receive the status information.

V = the control device address.

2) WRITE TTY CONTROL

The TTY control device output port is used to control the programmable paper tape reader interface, parity, and TTY interrupt. The output control byte has the following format:



The Reader On bit reads paper tape upon command. The command turns the reader on. The reader is turned off after the first start bit is received. The reader on command must be issued for every character. The Reader On output is a current loop output which drives a relay in the TTY. See Appendix C.

To issue the control byte, a Write Extended instruction

WRTE,r V

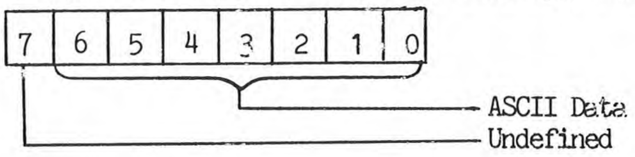
is issued, where: r = the 2650 register containing the control byte.

V = the control device address.

3) READ/WRITE TTY DATA

Data Device (Port 7)

The data device has an input and output data port associated that are used to receive and transmit data. The data byte has the following format:



A write TTY data command can be issued to the TTY when the transmit buffer empty status bit is true. To write data to the TTY, a Write Extended instruction,

WRTE,r V

is issued, where: r = the 2650 general purpose register that contains the output data.

V = the data device address.

A read TTY data command should be issued when the data available status bit is true. To read TTY data, a

REDE,r V

instruction is issued, where: r = the 2650 general purpose register to receive the data byte.

V = the data device address.

INTERRUPTS

The TTY forces program interrupt requests if the Interrupt enable control bit is set when the data available or transmit buffer empty conditions occur. The interrupt levels are as follows:

```
        DATA AVAILABLE = INTERRUPT LEVEL 6    VECTOR ADDRESS H'000C'
TRANSMIT BUFFER EMPTY = INTERRUPT LEVEL 7    VECTOR ADDRESS H'000E'
```

7.4.2 RS-232-C INTERFACE

The RS-232-C interface is a full communications interface. The following is a description of the interface and its utilization techniques.

DEVICE ASSIGNMENT

There are three devices associated with the RS-232-C interface.

Device Port 7 (plus board address):	Data device
Device Port 6 (plus board address):	Control device 1
Device Port 5 (plus board address):	Control device 2

RS-232 COMMANDS

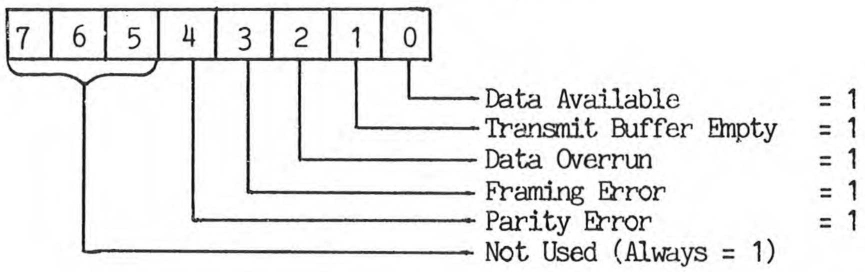
The RS-232 interface has commands to read and write data, read status, and write control.

1) REQUEST STATUS COMMANDS

STATUS BYTE 1

Control device 1 is used to read input status byte 1 information pertaining to the UART status.

Input Status Byte 1



DATA AVAILABLE. A character of input data is available in the input data port. This bit is reset when the data byte is read from the input data port.

TRANSMIT BUFFER EMPTY. The transmit data port is empty and ready to take a data character for output. The bit resets when an output data byte is sent to the output data port.

DATA OVERRUN. The data available in the input data port was not read prior to a new input data character being received. The bit is reset when the data available bit is cleared.

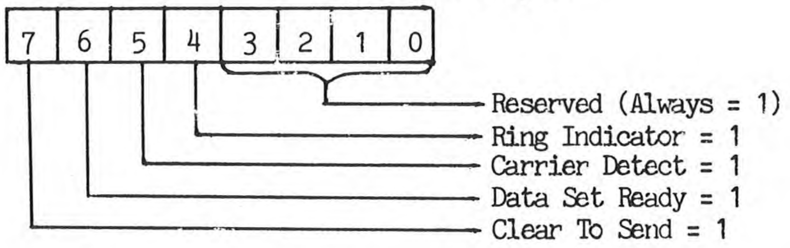
FRAMING ERROR. The received data character does not have a valid stop bit. The error bit is reset when a valid character and stop bit are received.

PARITY ERROR. The received data character had an incorrect bit count as specified by the odd/even parity selection control.

STATUS BYTE 2

Control device 2 is used to receive input status byte 2 information pertaining to communication line discipline.

Input Status Byte 2



RING INDICATOR. Indicates that the data set has received ringing current from a remote station. The indicator is true (=1) when the ringing current is present and false (=0) all other times.

CARRIER DETECT. Indicates that a data carrier has been detected by the data set. The bit will remain true (=1) as long as the carrier is detected.

DATA SET READY. Indicates the data set is ready to operate.

CLEAR TO SEND. Indicates that the data set is ready to transmit data.

The read extended command is used to read status byte 1 or 2 from the RS-232-C interface. To read status a

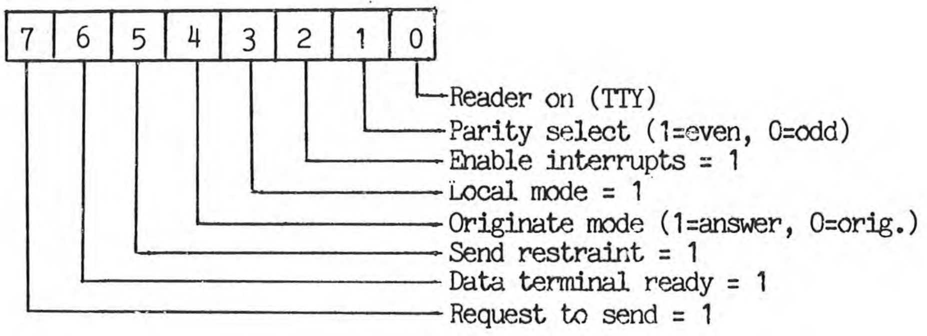
```
REDE,r V
```

instruction is issued, where: r = the general purpose 2650 register to receive status byte.

V = the control device address (6 for status byte 1 and 8 for status byte 2).

2) WRITE RS-232 CONTROL

Control Device 1 is used to control the RS-232 interface. The control byte has the following format:



READER ON. Sets the TTY paper-tape reader on and starts tape movement. (Available to TTY's with the reader on feature only).

PARITY SELECT. Selects data parity select mode (1-even, 0=odd parity select). The appropriate data parity is generated for transmit and checked on receive data.

ENABLE INTERRUPTS. Enables the generation of program interrupt requests when the data available or transmit buffer empty status indicators change from 0 (false) to 1 (true).

LOCAL MODE. Places the data set in the local mode so that the interface can function as a local circuit.

ORIGINATE MODE. Places data set in the answer mode (bit = 1) or originate mode (bit = 0). Answer mode implies the interface will respond to dial up indicators.

SEND RESTRAINT. Causes the data set to send a restraint signal when the ON condition is maintained. The OFF condition causes data transmission to be resumed.

DATA TERMINAL READY. Causes the data set to be connected to a communications channel.

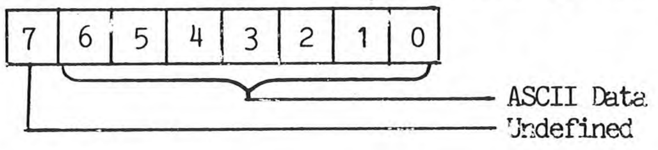
REQUEST TO SEND. Places the data set in a transmit condition. The ON condition must be maintained whenever traffic is ready to be transmitted or is being received.

The write extended command is used to control the RS-232-C interface. To write the control command, a

```
          WRTE,r V
instruction is issued, where: r = the general purpose 2650 register
                                  containing control byte.
                              V = the control device address (6).
```

3) READ/WRITE RS-232 DATA

The data device has an input and output I/O port associated with it used to receive and transmit data. The data byte has the following format:



The read data command is issued when data is available in the RS-232-C input data port. To read data a

```
        REDE,r V
```

instruction is issued, where: r = the general 2650 purpose register to receive data.

V = the data device address.

The RS-232-C write data command is issued when the transmit buffer is empty. To write data a

```
        WRTE,r V
```

instruction is issued, where: r = the general purpose 2650 register containing data.

V = the data device address (7).

INTERRUPTS

Program interrupts are generated by the RS-232-C interface if the interrupt enable control bit is set whenever the data available or transmit buffer empty status indicators change from a 0 to a 1 condition. The interrupt levels and vector addresses are defined below:

```
      DATA AVAILABLE = INTERRUPT LEVEL 6   VECTOR ADDRESS H'000C'
TRANSMIT BUFFER EMPTY = INTERRUPT LEVEL 7   VECTOR ADDRESS H'000E'
```

7.5 INSTALLATION

The General-Purpose I/O board may be utilized in products of a general nature. The board should be mounted, with side supports, in 100-pin PC Connector. Operating power is +5 V at 1.8 A. Mounting the board near other heat-radiating devices should be avoided. Pin lists for the board are given in tables 7-2 through 7-4.

TABLE 7-2

Pin List, Edge Connector P1

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
6	not used	
7	not used	
8	not used	
9	gnd	
10	gnd	
11	+12 V	Power Input
12	+12 V	Power Input
13	-12 V	Power Input
14	-12 V	Power Input
15	gnd	
16	gnd	
17	$\overline{\text{ADR 0}}$	Address Bit 0
18	$\overline{\text{ADR 1}}$	Address Bit 1
19	$\overline{\text{ADR 2}}$	Address Bit 2
20	$\overline{\text{ADR 3}}$	Address Bit 3
21	$\overline{\text{ADR 4}}$	Address Bit 4
22	$\overline{\text{ADR 5}}$	Address Bit 5
23	$\overline{\text{ADR 6}}$	Address Bit 6
24	$\overline{\text{ADR 7}}$	Address Bit 7
25-29	not used	
30	E/$\overline{\text{NE}}$	Extended Read/Write
31-35	not used	
36	$\overline{\text{DATA 0}}$	Data Bus Bit 0
37	$\overline{\text{DATA 1}}$	Data Bus Bit 1
38	$\overline{\text{DATA 2}}$	Data Bus Bit 2
39	$\overline{\text{DATA 3}}$	Data But Bit 3
40	$\overline{\text{DATA 4}}$	Data Bus Bit 4
41	$\overline{\text{DATA 5}}$	Data Bus Bit 5
42	$\overline{\text{DATA 6}}$	Data Bus Bit 6
43	$\overline{\text{DATA 7}}$	Data Bus Bit 7
44-51	not used	

TABLE 7-2

Pin List, Edge Connector P1 (continued)

Pin	Signal	Description
52	$\overline{\text{M}}$/IO	Memory/IO Enable Input
53	$\overline{\text{WRP}}$	Write Pulse Input
54	$\overline{\text{OPREQ}}$	Operation Request Input
55	R/$\overline{\text{W}}$	Read/Write Command Input
56	not used	
57	not used	
58	not used	
59	$\overline{\text{RESET}}$	System Reset Input
60	not used	
61	$\overline{\text{INTACK}}$	Interrupt Acknowledge Input
62	$\overline{\text{INT 0}}$	Interrupt Level 0 Output
63	$\overline{\text{INT 1}}$	Interrupt Level 1 Output
64	$\overline{\text{INT 2}}$	Interrupt Level 2 Output
65	$\overline{\text{INT 3}}$	Interrupt Level 3 Output
66	$\overline{\text{INT 4}}$	Interrupt Level 4 Output
67	$\overline{\text{INT 5}}$	Interrupt Level 5 Output
68	$\overline{\text{INT 6}}$	Interrupt Level 6 Output
69	$\overline{\text{INT 7}}$	Interrupt Level 7 Output
70	$\overline{\text{INT 8}}$	Interrupt Level 8 Output
71	$\overline{\text{INT 9}}$	Interrupt Level 9 Output
72	$\overline{\text{INT 10}}$	Interrupt Level 10 Output
73	$\overline{\text{INT 11}}$	Interrupt Level 11 Output
74	$\overline{\text{INT 12}}$	Interrupt Level 12 Output
75	$\overline{\text{INT 13}}$	Interrupt Level 13 Output
76	$\overline{\text{INT 14}}$	Interrupt Level 14 Output
77	$\overline{\text{INT 15}}$	Interrupt Level 15 Output
78-83	not used	
84-93	$\overline{\text{MAST RUN}}$	Master CPU Run Input
94	I/O CLK	I/O Clock Input
95	not used	
96	not used	
97	gnd	
98	gnd	
99	gnd	
100	gnd	

TABLE 7-3

Pin List, Edge Connector P2

Connector Pin	RS-232-C Pin	RS-232-C Designation	Description
1	1	AA	Protective Ground
2	14	--	TTY current loop return, xmit
3	2	BA	Transmitted data out
4	15	--	TTY reader control out
5	3	BB	Received data
6	16	--	TTY reader control return
7	4	CA	Request to send
8	17	--	TTY current loop out, rec
9	5	CB	Clear to send
10	18	--	TTY current loop return, rec
11	6	CC	Data set ready
12	--	--	Not used
13	7	AB	Signal ground
14	20	CD	Data terminal ready
15	8	CF	Carrier detected
16	21	--	Not used
17	9	+P	(Modem test only)
18	--	--	Not used
19	22	--	Not used
20	23	--	Not used
21	11	CK	Originate
22	24	--	Not used
23	12	CJ	Local
24	25	CL	Send restraint
25	13	--	TTY current loop out, xmit

TABLE 7-4

Pin List, Edge Connector P3

Pin	Signal	Description
1	OUT 0-7	Port 0, Output Bit 7
2	OUT 0-6	Port 0, Output Bit 6
3	OUT 0-5	Port 0, Output Bit 5
4	OUT 0-4	Port 0, Output Bit 4
5	OUT 0-3	Port 0, Output Bit 3
6	OUT 0-2	Port 0, Output Bit 2
7	OUT 0-1	Port 0, Output Bit 1
8	OUT 0-0	Port 0, Output Bit 0
9	OUT 1-7	Port 1, Output Bit 7
10	OUT 1-6	Port 1, Output Bit 6
11	OUT 1-5	Port 1, Output Bit 5
12	OUT 1-4	Port 1, Output Bit 4
13	OUT 1-3	Port 1, Output Bit 3
14	OUT 1-2	Port 1, Output Bit 2
15	OUT 1-1	Port 1, Output Bit 1
16	OUT 1-0	Port 1, Output Bit 0
17	gnd	
18	gnd	
19	gnd	
20	gnd	
21	IN 0-7	Port 0, Input Bit 7
22	IN 0-6	Port 0, Input Bit 6
23	IN 0-5	Port 0, Input Bit 5
24	IN 0-4	Port 0, Input Bit 4
25	IN 0-3	Port 0, Input Bit 3
26	IN 0-2	Port 0, Input Bit 2
27	IN 0-1	Port 0, Input Bit 1
28	IN 0-0	Port 0, Input Bit 0
29	IN 1-7	Port 1, Input Bit 7
30	IN 1-6	Port 1, Input Bit 6
31	IN 1-5	Port 1, Input Bit 5
32	IN 1-4	Port 1, Input Bit 4
33	IN 1-3	Port 1, Input Bit 3
34	IN 1-2	Port 1, Input Bit 2
35	IN 1-1	Port 1, Input Bit 1
36	IN 1-0	Port 1, Input Bit 0

CHAPTER 8

PROM PROGRAMMERS

8.0 INTRODUCTION

Two PROM programmer boards are used in the system, one for the bipolar type 82S115 and one for the MOS type 1702A. These boards are connected directly to the two PROM programmer sockets mounted on the front panel of the development computer. Through appropriate software commands, an assembled program can be "burned" into the PROM for the user's prototype hardware. Figures 8-1 and 8-2 are block diagrams of the two PROM programmers.

Initially, the 1702A PROM to be programmed will have been "erased" by exposure to an ultraviolet source and all 2048 bits will be in the "0" state (output low). Information is programmed by selectively holding "1"s in the proper bit locations while pulsing the power supply voltages for a specified period of time. Each address is selected and the object code to be entered in that loca-is programmed. When programming is complete, the programmer is deactivated under system control and the stored program can be compared with the source data to verify that the stored data is correct.

The 82S115 family of PROMs are so-called "fusible link" devices which have all bits initially set to logic "0". The PROM programmer "fuses" the links by applying high-voltage pulses and sets the appropriate bits at logic "1". This is a non-reversible process, so care must be taken to ensure that the program to be burned into the PROM is correct. As with the 1702A programmer, the 82S115 family is programmed by applying addresses and data to the appropriate device pins and cycling through each address until the PROM is completely programmed. The basic difference in programming the two PROM types (besides non-reversible programming) is the fact that the 82S115 family has data entered serially by bit instead of serially by word.

Although the usual procedure is to program the PROM completely in the first pass, the software supports a means of beginning programming at an address other than 00, so that an incompletely programmed device can be programmed completely by specifying a starting address other than 00 and writing until the PROM has been filled with data.

8.1 1702A PROGRAMMER

Referring to Figure 8-1, the 1702A Programmer consists of voltage switching, data registers, control registers, address registers, sequencer, and drivers.

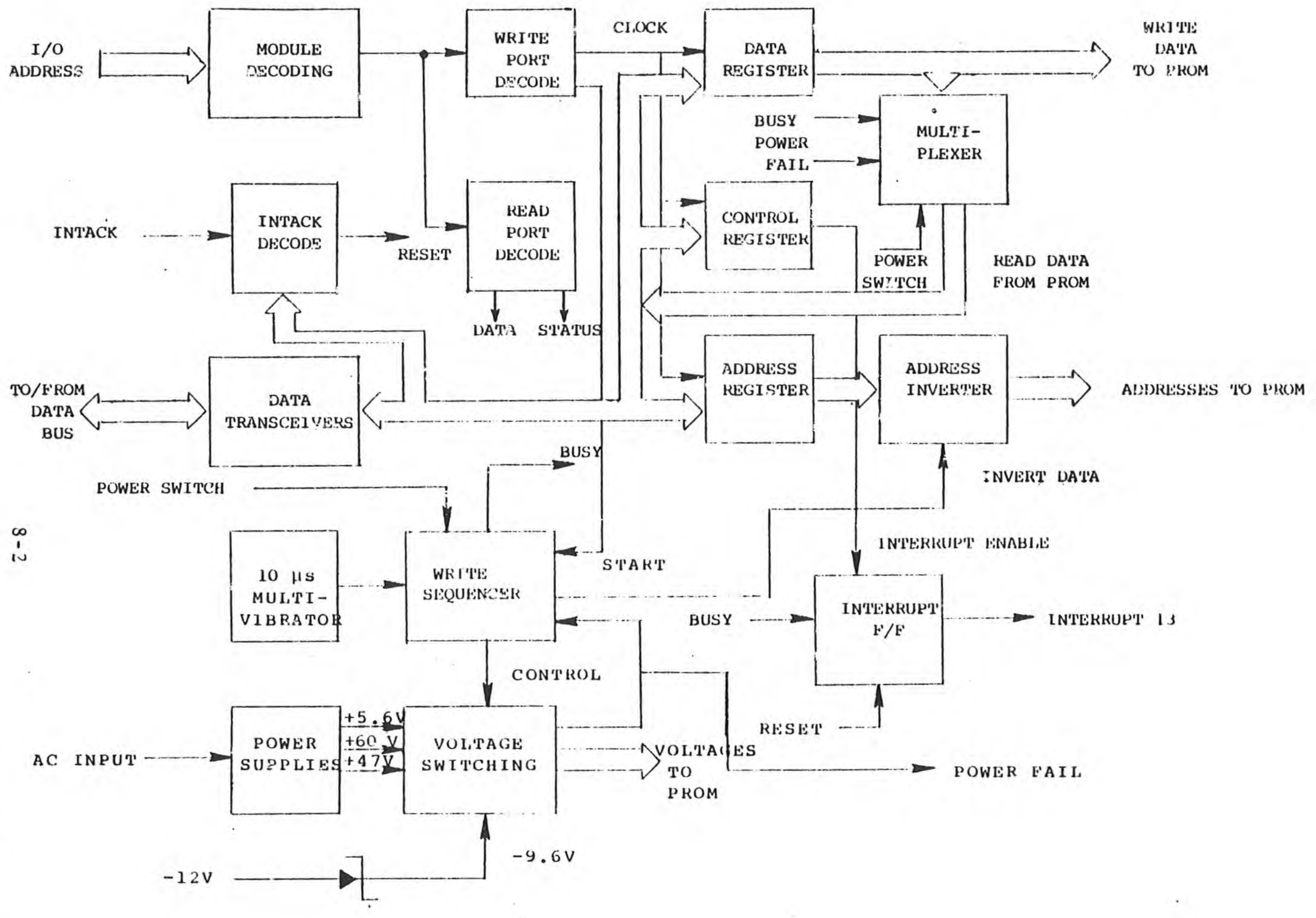


FIGURE 8-1

1702A PROGRAMMER BLOCK DIAGRAM

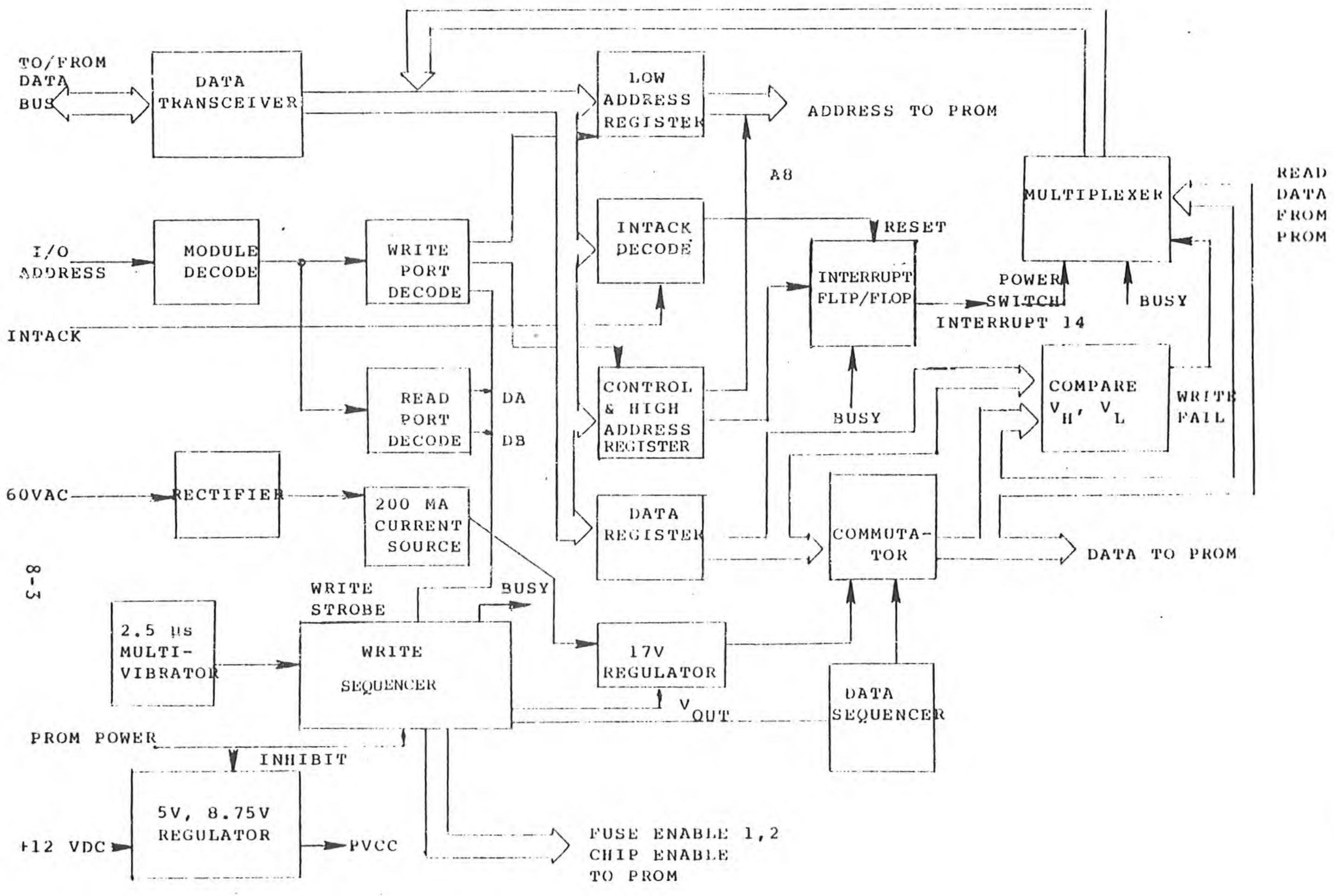


FIGURE 8-2

82S115 PROGRAMMER BLOCK DIAGRAM

Operation of the programmer is under control of the TWIN system and its associated commands. Once a PROM has been inserted in the socket and the PROM POWER switch turned on, the sequence of events is controlled by operator interaction with the TWIN system. If a PROM is completely erased, and a complete program is to be entered, the operator commands "write PROM" from Slave memory. The TWIN system now outputs a low address (eight bits) and data (eight bits) to the programmer registers. Initially, the address bits are inverted, and as soon as the programming voltages have reached their negative levels, returned to the normal condition. The programming voltages are held for a period of approximately 3 ms. After the initial write, the word is read from the PROM and compared to the input word. If a valid comparison is made, the programmer then programs the device for five additional cycles. If a valid comparison is not made, the programmer tries again. If 80 retries are made without success, the programmer interrupts the CPU and flags the system.

When a programming sequence is successful, the software increments the address and outputs data for the next location. This sequence continues until the software reaches the operator-specified ending address, or the last PROM address (the default value).

After the device has been programmed, its contents can be compared to the data in the Slave memory to verify correct programming.

8.1.1 CIRCUIT DETAILS

The system address and data buses are brought to the programmer board through the mother board. The address lines are used to select either the 1702A or the 82S115 boards. If a valid address for the board is decoded, a control line (MODULE SELECTED) enables the address registers for the addresses which will be input from the data bus in subsequent portions of the cycle.

Data lines are input to transceiver devices which are enabled by a decoded data strobe and input to registers which hold the data throughout each programming cycle. The output of both data and address registers drive transistor drivers, which are used because of the high levels used in driving the address lines to the device to be programmed.

Figure 8-1 shows the programming power supplies, the sequencer, and the outputs for programming the PROM. Unregulated DC for the three supplies (+60, +47, and +5.6 V) is provided by a full-wave bridge rectifier and capacitive filter. A series pass transistor regulates the +60 V supply, with current limiting provided. Voltage reference for the +60 V supply is provided by the +47 V supply. The +47 V supply is regulated by a series pass transistor with current limiting provided. Overvoltage protection comes from an SCR which monitors the +47 V supply. If the supply rises above +47 V, one of the two PROM programmer supply fuses located on the rear panel will blow. As soon as the monitor triggers, the Power Fail flip-flop is set. This shuts down the sequencer and sends a "power fail" indication to a multiplexer to indicate to the system software that a failure has occurred. A supply voltage of -9.6 V is provided by dropping the system -12 V supply voltage through a Zener diode. It should be noted that, while the 1702A programming voltages are normally specified as negative, the PROM is connected so that Vbb is at +60 V, and all the other programming voltages are negative with respect to +60 V.

There are normally three states assumed by the power supplies: 1) normal operation when the device is being read, 2) an "off" condition when PROM POWER on the front panel is turned off, and 3) the condition when the PROM is programmed. These conditions are controlled by the sequencer on the board.

The Vbb, Vcc, and Vgg power supplies are pulsed by the sequencer. Other pulsed outputs (Program and Chip Select) are derived from other sequencer outputs in the chain. Basic timing is set by the Data Strobe input and two one-shots.

8.2 82S115 PROGRAMMER

The 82S115 Programmer is very similar to the 1702, with one important exception: the device is programmed serially by bit instead of serially by word. Another exception is the level of programming voltages required by the PROM to fuse the links and write logic "ones" in the selected locations.

Referring to Figure 8-2, the address and data inputs are brought into the board via the system mother board bus. The operation of the programmer is determined by decoding the address lines during the first portion of the cycle. Either a write, compare, or read operation can be performed.

Data is input to the programmer through transceivers connected to the internal board data bus. The sequence of events for programming is as follows:

1) Read and verify status (power switch on, power fail, programmer busy, etc.).
2) Strobe the least significant address byte from the data bus into the address registers.
3) Strobe the most significant address byte from the address bus into the address registers.
4) Write the data.
5) Wait for the busy status line to go false, indicating that the write cycle is complete, and interrupting the Slave CPU.

8.2.1 CIRCUIT DETAILS

Data is strobed into the eight-bit data register which consists of one hex flip-flop and a dual flip-flop.

The address bits are strobed into a 9-bit register similar to the data register. The output of the register is connected to a set of gates and inverters which drive the address lines of the PROM directly.

Data lines from the PROM are brought into a set of comparators and gates which input the data to a multiplexer. This device holds the data for transmission to the data bus at the appropriate time in the cycle. Outputs are taken from the comparators and brought to gates which perform a comparison function with the data stored in the data register previously discussed. This generates a COMPARE signal which controls the sequencer.

The power supplies used in the 82S115 programmer are much simpler than those used in the 1702A. There is a Vcc level used in programming, and a normal Vcc used in reading and verifying, and there is an enable voltage for the fusing process. Data programming voltage is derived from a power supply consisting of a full-wave rectifier and capacitive filter followed by a 200 mA current source and a voltage regulator. Power (Vcc) Programming voltage, labeled PVCC, is provided by a regulator which derives its input voltage from the system +12 V bus. Failure detection is provided by a transistor which is held in conduction by current through a Zener diode connected to the Vcc output. If the supply voltage drops below approximately +4.2 V, the transistor turns off and sets a flip-flop to give a Power Fail indication.

In general, the programming sequence is as follows:

1. The Vcc supply is raised to the programming level of +5 V and the address lines are activated.

2. After 10 microseconds, Fuse Enable 1 is raised to +5 V.

3. After an additional 10 microseconds, +17 V is applied to the first bit to be programmed as a logic "one". This voltage level is held for approximately 1 ms.

4. After a 10 microsecond delay, Fuse Enable 2 is raised to +5 V and held for 1 ms.

5. After a 10 microsecond delay, the +17 level is turned off.

6. Programming is then verified by applying two verification voltages and verifying that the bit stays in the "one" state. If verification fails, the sequence is

repeated. If verification does not fail, the sequence is repeated eight times. The sequencer then shifts to the next bit to be programmed as a "one" and the programming sequence is repeated as in steps 1-5 above.

8.3 APPLICATION

The two PROM Programmer boards are special-purpose modules which require special software support not normally available to the user for inclusion of the boards in a general product. For this reason, no instructions for application of the boards outside the system are given.

Pin lists for the two boards are given in Tables 8-1 through 8-3.

TABLE 8-1

PIN LIST, EDGE CONNECTOR P1

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
5-8	not used	
9	gnd	
10	gnd	
11	not used	
12	not used	
13	-12 V	Power Input
14	-12 V	Power Input
15	gnd	
16	gnd	
17	$\overline{\text{ADR 0}}$	Address Bit 0 Input
18	$\overline{\text{ADR 1}}$	Address Bit 1 Input
19	$\overline{\text{ADR 2}}$	Address Bit 2 Input
20	$\overline{\text{ADR 3}}$	Address Bit 3 Input
21	$\overline{\text{ADR 4}}$	Address Bit 4 Input
22	$\overline{\text{ADR 5}}$	Address Bit 5 Input
23	$\overline{\text{ADR 6}}$	Address Bit 6 Input
24	$\overline{\text{ADR 7}}$	Address Bit 7 Input
25-29	not used	
30	$\overline{\text{ADR 13}}$	Address Bit 13 Input
31-35	not used	
36	$\overline{\text{DATA 0}}$	Data Bit 0 Input
37	$\overline{\text{DATA 1}}$	Data Bit 1 Input
38	$\overline{\text{DATA 2}}$	Data Bit 2 Input
39	$\overline{\text{DATA 3}}$	Data Bit 3 Input
40	$\overline{\text{DATA 4}}$	Data Bit 4 Input
41	$\overline{\text{DATA 5}}$	Data Bit 5 Input
42	$\overline{\text{DATA 6}}$	Data Bit 6 Input
43	$\overline{\text{DATA 7}}$	Data Bit 7 Input
44-51	not used	
52	$\overline{\text{M}}$/IO	Memory/IO Command Input
53	$\overline{\text{WRP}}$	Write Pulse Input
54	$\overline{\text{OPREQ}}$	Operation Request Input
55	R/$\overline{\text{W}}$	Read//Write Command Input
56	$\overline{\text{HOLD}}$	Hold Command to CPU
57-60	not used	
61	$\overline{\text{INTACK}}$	Interrupt Acknowledge Input
62-74	not used	

TABLE 8-1

PIN LIST, EDGE CONNECTOR P1 (continued)

Pin	Signal	Description
75	$\overline{\text{INT 13}}$	Interrupt Level 13 Output
76-82	not used	
83	50 VAC	Power Supply AC Input
84	50 VAC	Power Supply AC Input
85	50 VAC	Power Supply AC Input
86	50 VAC	Power Supply AC Input
87-95	not used	
96	gnd	
97	gnd	
98	gnd	
99	gnd	
100	gnd	

TABLE 8-2

PIN LIST, 1702A PROGRAMMER EDGE CONNECTOR P2

Pin	Signal	Description
1	PRA 7	Device Address Output Bit 7
2	PRA 6	Device Address Output Bit 6
3	PRA 5	Device Address Output Bit 5
4	PRA 4	Device Address Output Bit 4
5	PRA 3	Device Address Output Bit 3
6	PVCC	Programming Vcc Output
7	PVCC	Programming Vcc Output
8	PVDD	Programming Vdd Output
9	PVGG	Programming Vgg Output
10	PVBB	Programming Vbb Output
11	CS	Chip Select Output
12	V PROG	Device Programming Pulse Output
13	PRD 7	Device Data Output Bit 7
14	PRD 6	Device Data Output Bit 6
15	PVCC	Programming Vcc Output
16	PRD 5	Device Data Output Bit 5
17	PRD 4	Device Data Output Bit 4
18	PRD 3	Device Data Output Bit 3
19	PRA 2	Device Address Output Bit 2
20	PRA 1	Device Address Output Bit 1
21	PRA 0	Device Address Output Bit 0
22-24	not used	
25	PROM PWR SWITCH	Front-Panel Switch Output
26	gnd	

TABLE 8-3

PIN LIST, 82S115 PROGRAMMER EDGE CONNECTOR P2

Pin	Signal	Description
1	PRD0	Device Data Output Bit 0
2	CE	Chip Enable Output
3	PRA8	Device Address Output Bit 8
4	PRA7	Device Address Output Bit 7
5	PRD1	Device Data Output Bit 1
6	PRD2	Device Data Output Bit 2
7	PRD3	Device Data Output Bit 3
8	PRA6	Device Address Output Bit 6
9	PRA5	Device Address Output Bit 5
10	PRA4	Device Address Output Bit 4
11	PRA3	Device Address Output Bit 3
12	PRA0	Device Address Output Bit 0
13	PRA1	Device Address Output Bit 1
14	PRA2	Device Address Output Bit 2
15	PRD5	Device Data Output Bit 5
16	PRD6	Device Data Output Bit 6
17	PRD7	Device Data Output Bit 7
18	not used	
19	CE2	Chip Enable 2 Output
20	CE1	Chip Enable 1 Output
21	PRA0	Device Address Output Bit 0
22	PRA1	Device Address Output Bit 1
23	PRA2	Device Address Output Bit 2
24	PVCC	Programming Vcc Output
25	PRD4	Devic Data Output Bit 4
26	FE 1	Fuse Enable 1 Output
27	FE 2	Fuse Enable 2 Output
28	gnd	
29	PRD3	Device Data Output Bit 3
30	PRD2	Device Data Output Bit 2
31	PRD1	Device Data Output Bit 1
32	PRD0	Device Data Output Bit 0
33	PRA3	Device Address Output Bit 3
34	PRA4	Device Address Output Bit 4
35	PRA5	Device Address Output Bit 5
36	PRA6	Device Address Output Bit 6
37	PRA7	Device Address Output Bit 7
38	PRA8	Device Address Output Bit 8
39	not used	
40	gnd	

CHAPTER 9

DEBUG AND FRONT PANEL I/O MODULE

9.0 INTRODUCTION

The Debug and front panel I/O module performs three distinct functions within the development computer: a) It controls the interaction and bus time-sharing of the Master and Slave CPU's, b) It supports such software Debug features as breakpoint, trace, and start of Slave programs at any memory location. c) It provides interface to either the standard or maintenance front panel and supports full maintenance panel control and display functions. The card is centrally located in the computer bus structure, providing a dividing line for Master and Slave halves. Unique control lines are connected to the Master CPU and Slave CPU in each half of the bus.

9.1 GENERAL DESCRIPTION

A simplified block diagram of the board is shown in Figure 9-1. Although the logic sections are interrelated, the board is described under four major topics:

1. Master/Slave control
2. I/O commands and interrupts
3. Debug Features
4. Front panel interface

Master/Slave control determines bus control between Master and Slave CPU's at all times. Debug features include storage of the Slave CPU address value (program counter) during instruction fetch of the last instruction and present instruction. Also included are two breakpoint registers with address comparison logic, and forced jump logic including jump address storage. The I/O consists of port decoding for storing of breakpoint values, loading the Debug command and control bytes, and reading the stored program counter address values. Slave I/O service requests are also decoded. Debug interrupts (breakpoint, single cycle, Slave halted) as well as the eight service requests are priority vector encoded for presentation to the Master CPU. The front panel interface routes bus address and data to the front panel and interfaces data and address values from the front panel during memory or I/O access. It also supports various maintenance front panel control operations such as single step.

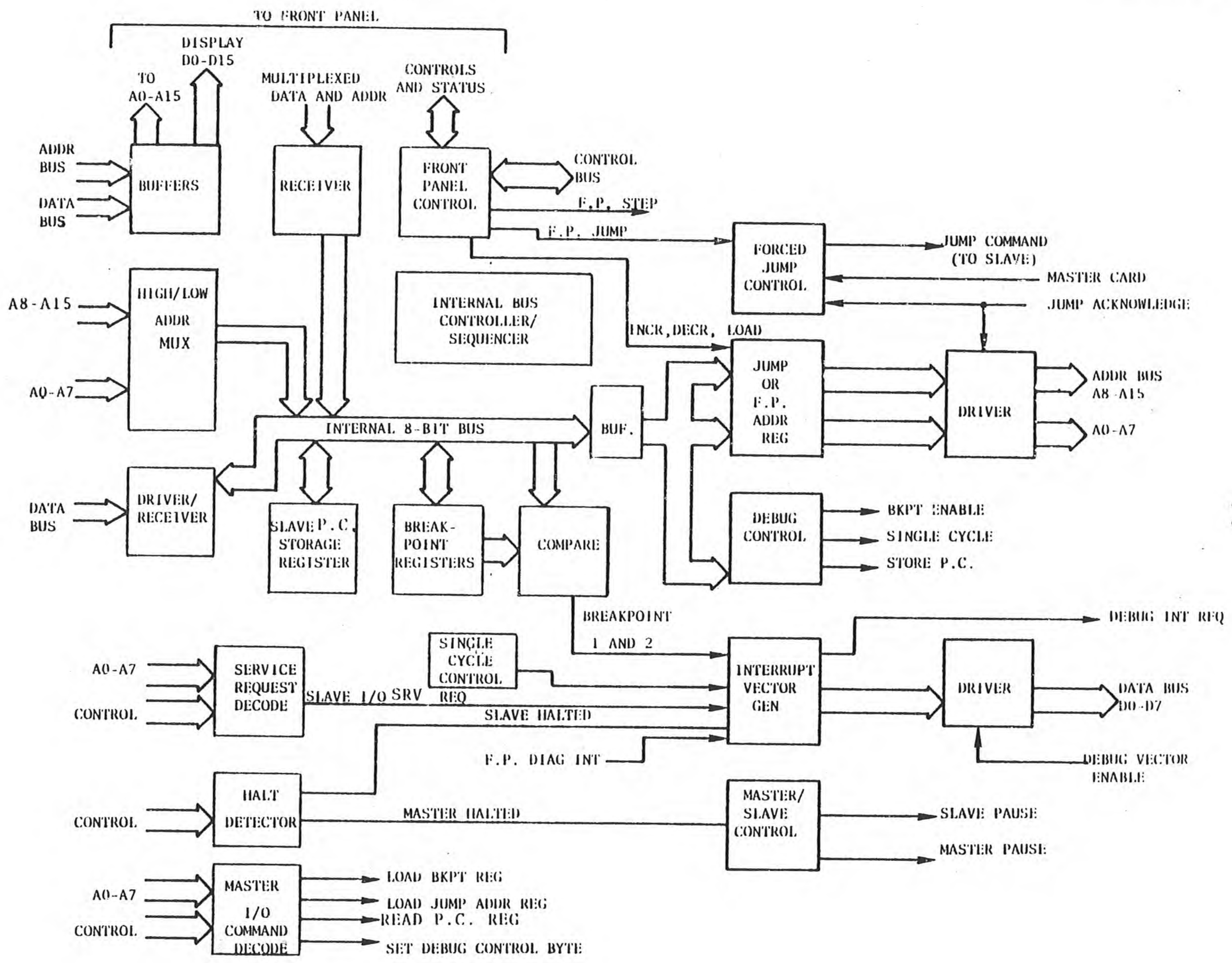


FIGURE 9-1

DEBUG AND FRONT PANEL I/O BLOCK DIAGRAM

9.1.1 Master/Slave Control

Master/Slave control is straightforward. The Master relinquishes bus control by executing a program halt. The Slave is then allowed to RUN and the Master is paused until a Master Interrupt occurs. This may be due to a breakpoint or other Debug interrupt, a service request, a console interrupt or an interval timer interrupt. The Slave is paused, and when off the bus, the Master is again allowed to run.

Both Master and Slave may be deactivated by a maintenance front panel signal or by a DMA controller causing a pause to both Master and Slave. When the pause is removed, bus control is resumed in the state prior to the pause. The maintenance panel can also set the state of the Master/Slave control to either Master or Slave.

9.1.2 Debug Features

Certain software Debug routines require hardware support that is not necessarily unique to a particular Slave processor. These features are contained on the Debug module. The module contains two breakpoint registers with a memory address comparator, two Slave program counter storage registers, and a forced jump address register. It also includes logic to provide single cycle interrupts during trace operation, a forced Slave reset, and forced Slave interrupt logic.

A central feature of the Debug Logic is use of an internal 8-bit bus to handle all address and data flow. File registers organized into four words by eight bits are used to write and read breakpoint values and Slave program counter values. A bus controller containing sequencing logic controls bus flow and has several modes of operation. The front panel also makes use of this bus structure to load the dual-purpose address register and place data on the system bus. The sequencer operates only when commanded by Debug mode or during front panel access. A normal sequence is triggered by a Slave OPREQ. The low and high address bytes are sequentially placed on the internal bus and compared to breakpoint address 1. At the same time, the address bytes are loaded into one of the program counters. Then the two address bytes are again sequentially accessed for comparison to breakpoint address 2. If the breakpoint conditions specified by the command byte are met, the Slave is paused and the Master interrupted.

Forced jump logic is partially contained on the Slave CPU and partially on the Debug module. The Slave contains a jump sequencer and data multiplexing logic that when triggered, substitute an unconditional branch for the next instruction fetch and sequentially feed the following two jump address bytes to the CPU data bus. The jump address is stored in the dual purpose address register on the Debug module. Upon Master I/O commands, the address is loaded and a jump command trigger given to the Slave CPU. During the sequence, the Debug module places the jump address on the address bus for use by the Slave multiplexer.

Other Debug features include single cycle mode, forced Slave reset and forced Slave interrupt. During single cycle, the Slave is paused on execution of a single instruction and the Master is interrupted. Single cycle or breakpoint interrupts to the Master also initiate a forced jump sequence for the Slave CPU. The 2650 processor, when paused, always stops after fetching the first byte of the next instruction. Since it is necessary to jump the Slave to a register dumping routine immediately without an intervening instruction, the forced unconditional branch command must be received by the Slave for its next instruction before stopping. The Master then takes control and loads the Debug module with the jump destination. The Master then halts, reenabling the Slave, which jumps to the register dump routine.

9.1.3 I/O Commands and Interrupts

I/O commands to the Debug module come only from extended Master CPU I/O commands. These include the following:

1) 4 bytes output breakpoint (1 and 2, low and high byte)

2) 4 bytes input program count (next and last, low and high byte)

3) 2 bytes output jump address

4) 1 byte Debug command

5) 1 byte Debug control

Breakpoints, program counter values, and jump address have been discussed in section 9.1.2. The command byte contains individual bits commanding a Slave forced reset, forced interrupt, forced jump and a Debug interrupt mask bit. The control byte sets the Debug mode to active or inactive mode and controls single cycle and breakpoint modes.

The interrupt structure can be divided into two sections. Slave I/O commands, F0-F7, called service requests, directly produce interrupt vectors 0020 thru 002E (Priority levels 16-23). Debug interrupts consisting of the two breakpoints, single cycle, Slave halted and front panel diagnostic interrupt are priority vector encoded for addresses 0030-0038 (priority level 23-28). Presence of any interrupt produces a single Debug interrupt to the Master CPU. Priority is assigned below the 16 level Master interrupt structure. If no higher level is pending, the CPU gives priority to the Debug card during $\overline{\text{INTACK}}$ allowing the Debug module to place the vector on the data bus and reset the interrupts.

9.1.4 Front Panel Interface

The Debug module contains full interface capability for the maintenance front panel, allowing field substitution for the standard front panel at any time. Since the CRT console provides all operator interface and display necessary during normal operation, the standard front panel contains minimal controls. The maintenance panel, on the other hand, contains full hexadecimal data and address displays and individual switches for data and address entry. Full control functions such as front panel access of memory or I/O ports, single step and hardware breakpoint are included.

The Debug module contains buffers to route the 16 address bits and up to 16 data bits on the front panel displays. Data and address switch values are multiplexed in 8-bit bytes to the internal Debug module bus as previously discussed. Most of the logic necessary for the front panel functions is contained on the front panel circuit boards, but some interaction with the Debug control functions is necessary. These are Master/Slave control, forced jump feature, diagnostic interrupt and a special function called Master test which allows display of the last address value executed.

9.2 DETAILED DESCRIPTION

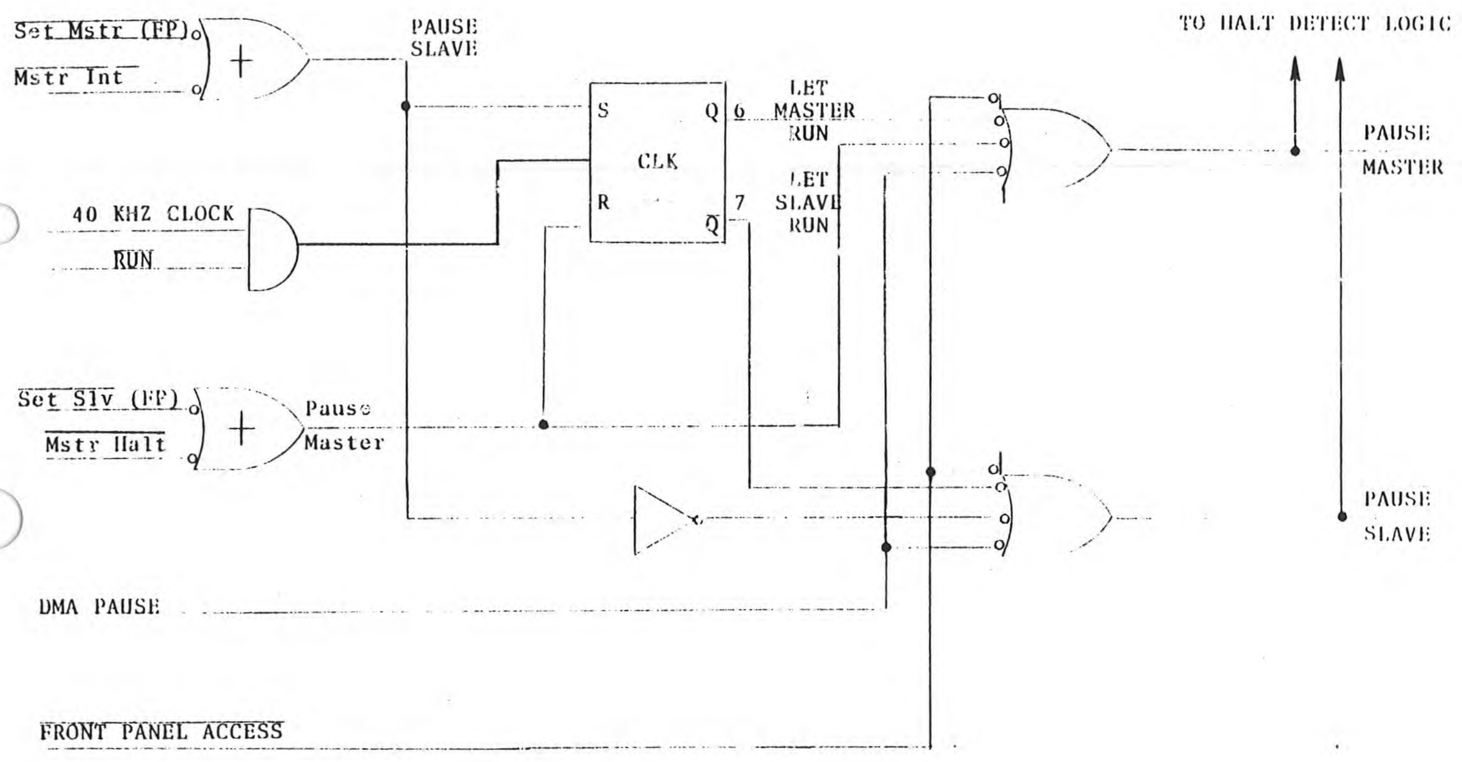
9.2.1 Master/Slave Control and Halt Detector

A simplified logic diagram of the Master/Slave control is shown in Figure 9-2. One line to each half of the mother board controls Master/Slave operation. A pause to the Master or to the Slave causes it to relinquish bus control and causes the output of all address, data and control lines to go into a high-impedance state at the end of the current instruction.

As previously discussed, both Master and Slave are paused during front panel access, or for DMA operations. Otherwise, the Debug control logic takes precedence. A single flip-flop stores Master/Slave status. When an event occurs to change the status, the active device is directly paused. When out of the run state, the status flip-flop is clocked to the new state allowing the inactive device to run. The Slave is paused if a Master interrupt occurs or if the front panel "SET MSTR" switch is depressed. The Master is paused if a Master halt is detected or "SET SLV" switch on the front panel is depressed.

Master/Slave halt detectors are shown in Figure 9-3. A Master halt is the normal mechanism the Master uses to relinquish bus control. A Slave halt indicates a programming error and immediately causes a Master interrupt, so the user can be notified. Operation of the halt detectors is identical. If either the Master or Slave is in run, the respective halt indicators are held in a reset condition. If the run indication is not present and no pause has been issued for at least 25 microseconds, the halt detected flip-flop is set. Two flip-flops are required to provide the necessary 25 microsecond delay.

FIGURE 9-2 MASTER/SLAVE CONTROL



Control is transferred to the front panel in Front Panel Access.
Control is transferred to Slave if the Master is halted or Slave selected.
Control is transferred to Master if any Master interrupt occurs.
"RUN" must be low for any of the above to occur.

9.2.2 I/O Commands and Interrupts

Debug features are controlled totally by the Master CPU module. To do this, I/O commands controlling the operation and the register contents are issued by the Master CPU. A list of input-output port assignments is shown in Figure 9-4. The breakpoint addresses are loaded by the Master CPU by user request. The forced jump address may be loaded to execute a "GO" command from the user or to jump the Slave to a trace routine. The program counter contents are read only during trace or after detecting a breakpoint.

OUTPUT

-Command byte	F9
-Control byte	F8
-Jump addr low, high	FA,FB
-Breakpoint 1 low, high	FC,FD
-Breakpoint 2 low, high	FE,FF

INPUT

-Program counter last low, high----FC,FD
-Program counter next low, high----FE,FF

FIGURE 9-4
DEBUG I/O PORT ASSIGNMENTS

The command byte consists of four individual commands:

Forced Interrupt Command

This command forces a Slave processor interrupt so that when the Slave processor is started, a forced ZBSR instruction at page 0 location 0 is issued.

Forced Jump Command

This command forces the Slave processor to branch unconditionally to the address contained in the dual purpose address register. The address register must be set prior to issuing this command.

Mask Debug Interrupts Command

This command sets a condition that will not allow any Debug hardware interrupt to occur. The function is activated to disable concurrent Slave program service requests and breakpoint interrupts. The masked interrupt is stacked and is allowed to occur when the mask is reset.

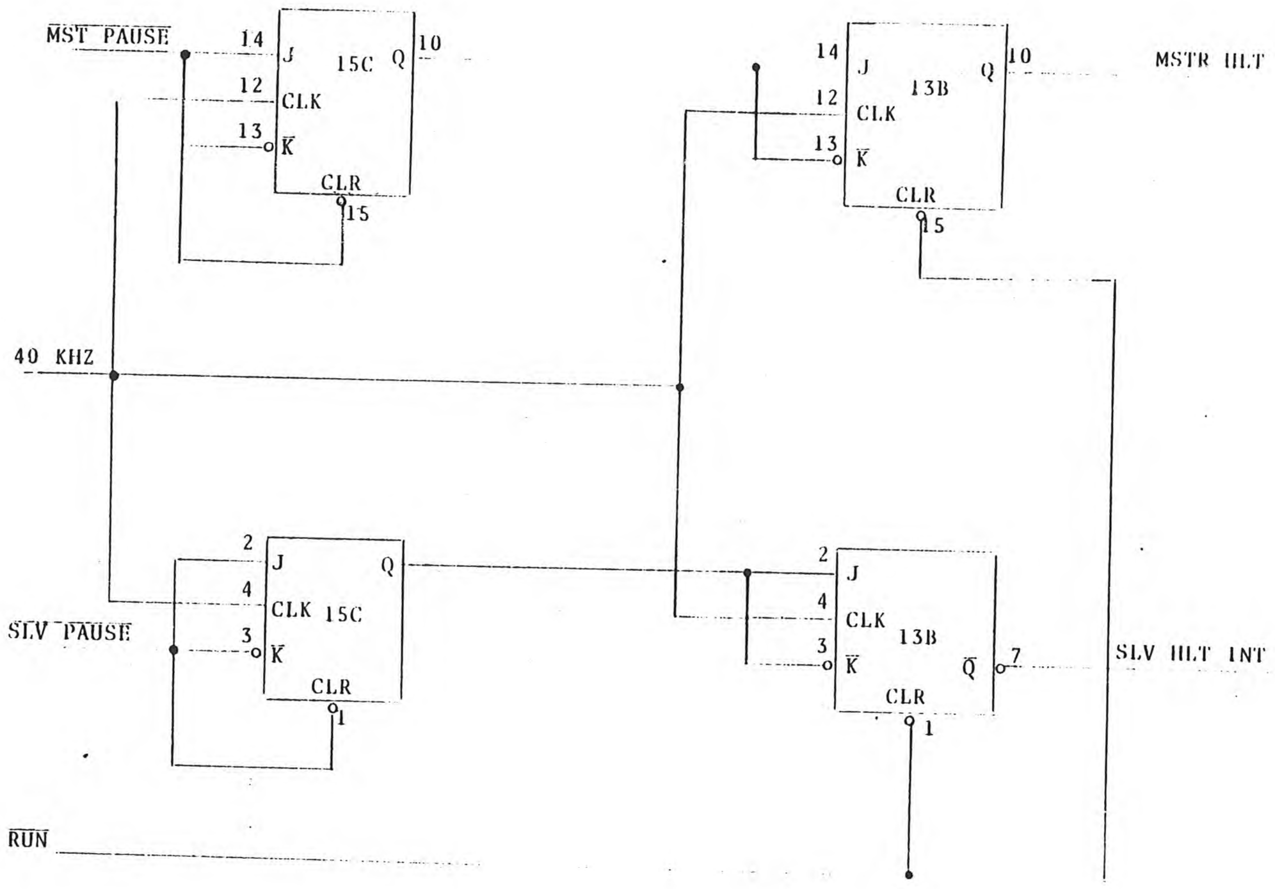
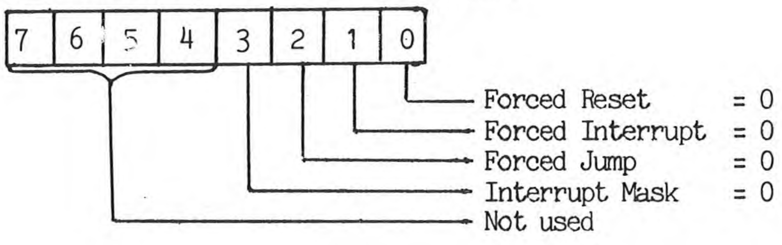


FIGURE 9-3

MASTER/SLAVE HALT DETECTOR

The command byte format is as follows:



The control byte contains six individual bits controlling the Debug mode of operation. The format is shown in Figure 9-5. The Debug mode enables or disables the sequencer loading the program counter values and comparing breakpoint address. The other bits are self-explanatory, involving trace or breakpoints upon memory read or write. The bits are non-exclusive and any combination of breakpoint bits may be used.

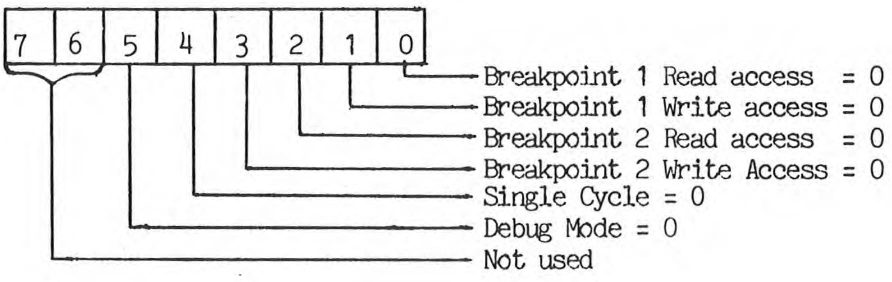

7
6
5
4
3
2
1
0
Breakpoint 1 Read access = 0
Breakpoint 1 Write access = 0
Breakpoint 2 Read access = 0
Breakpoint 2 Write Access = 0
Single Cycle = 0
Debug Mode = 0
Not used

FIGURE 9-5

CONTROL BYTE FORMAT

All commands decode the lower eight address bits to channel the data to or from a particular register and require Master active, extended I/O, OPREQ and WRP (if an output).

Interrupts

Interrupts from the Debug module are initiated by two sources: Slave service requests or by Debug features. A simplified block diagram is shown in Figure 9-6.

Slave service requests utilize the I/O module address recognition, but require the Slave to be active rather than the Master. When a Slave I/O write from

F0-F7 is detected, a "DBG INT" goes to the Master CPU. Priority is established by a combination of the vector encoders on the Master CPU and on the Debug module. A list of Master interrupts is shown in Figure 9-7. When the CPU acknowledges the interrupt with "DBG VEN" (Debug vector enable) the lower three address bits of the I/O service request (A0-A2) are routed to the data bus as D1, D2, and D3 in the inverted form. Bit 5 is always set to "one" and bit 4 is set to "zero" for service requests and "one" for Debug interrupt. Note that a Slave I/O command corresponds to Slave service request 1 and vector 0020. The trailing edge of "DBG VEN" clocks the register holding the service request vector back to zero.

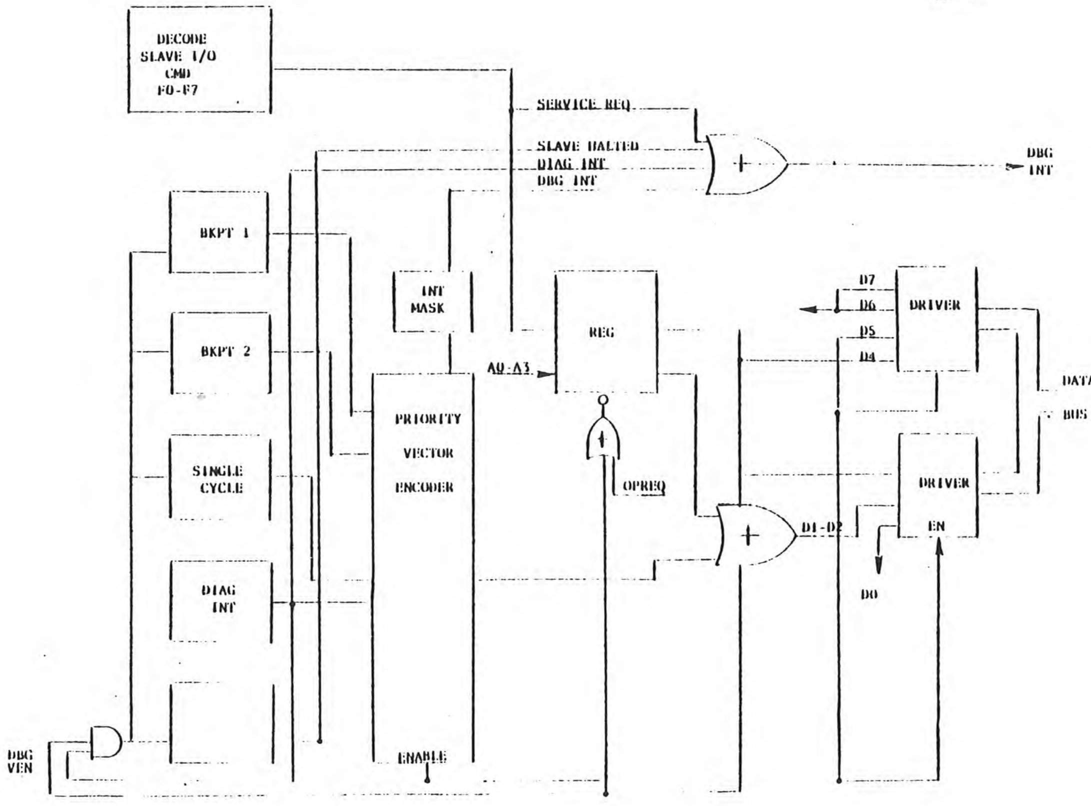


FIGURE 9-6

Priority	Address		
0	0000	Reset	
1	0002		
2	0004	Paper Tape Rdr	
3	0006		
4	0008	TTY In	
5	000A	TTY Out	
6	000C	RS-232-C In	Master I/O
7	000E	RS-232-C Out	
8	0010		
9	0012	Clock	
10	0014	Printer	
11	0016	Floppy Disk	
12	0018		
13	001A	PROM Burner 1	
14	001C	PROM Burner 2	
15	001E		
16	0020	Slave SVC 1	
17	0022	Slave SVC 2	
18	0024	Slave SVC 3	
19	0026	Slave SVC 4	Service Requests
20	0028	Slave SVC 5	
21	002A	Slave SVC 6	
22	002C	Debug SVC 1	
23	002E	Debug SVC 2	
24	0030	Breakpoint 1	
25	0032	Breakpoint 2	
26	0034	Single Cycle	
27	0036	Halt	Debug Hardware
28	0038	Diag. Control	
29	003A		
30	003C		
31	003E		

FIGURE 9-7

MASTER CPU INTERRUPT PRIORITY ASSIGNMENT

The Debug feature interrupts start with individual flip-flops set by an event. Breakpoint 1, Breakpoint 2, single cycle Slave halted, and the front panel DIAGNOSTIC INT switch each set a particular flip-flop.

A priority encoder receives the output of the flip-flops and generates a "DBG INT" to the Master. Action is as previously described for the service requests. The highest priority interrupt present when "DBG VEN" is received is encoded and placed on the data bus. Note that there is an interrupt mask, masking only the breakpoint and single cycle interrupts. This is controlled by bit 3 of the command byte. All Debug interrupt flip-flops are reset simultaneously, thus only the highest priority interrupt is seen by the Master CPU.

9.2.3 Debug Features

As previously noted, the Debug hardware consists of two Slave instruction address registers (program counters), two breakpoint registers and associated address comparators, forced jump address storage and control, and Slave single cycle control. These functions are enabled or disabled by the Debug Control byte. Additionally covered under this topic are Slave forced reset and forced interrupt.

Internal Bus and Sequencer

The internal bi-directional bus is connected to the mother board data bus by bus driver-receiver pairs. It is connected to both the input and output of the file registers used for program counter storage. It is connected to the input of the breakpoint file registers and one input of the comparators. Through buffers it is further connected to the dual purpose (jump address/front panel) register and to the Debug control byte register.

The data bus is used during the following operations:

1) Reading the P.C. file registers

2) Loading the breakpoint file registers

3) Transferring maintenance front panel switch data to the bus

4) Loading the Debug control byte

5) Loading the jump address register.

These operations are controlled by the I/O decode logic except item 3. During front panel access, an enter command causes data to be multiplexed from the front panel data switches, output to the Debug internal bus and out to the mother board bus.

During sequencer operation, the data bus connection is not used. Rather, the address bus is multiplexed onto the bus eight bits at a time. When the Debug mode is enabled, the sequencer, utilizing a 10 MHz clock, is triggered by the leading edge of "OPREQ". A hold is issued to insure that the address data lasts for the duration of the sequencer operation. Four flip-flops generate the basic timing sequence. There are five basic intervals. During the first interval, the low half of the address bus is output to the internal bus. This is stored in the low byte of one of the program counter file registers. At the same time, the low address is accessed from breakpoint 1 file register and compared to the address on the internal bus with exclusive OR gates. The result of this comparison is clocked into the first stage of a compare circuit. In the second interval, the upper address byte is written into the high byte of the same program counter file register and the high address is accessed from the breakpoint 1 register and compared to the high address byte on the bus. If the last compare is valid, the result is clocked into the second stage of the compare circuit. During the third interval, the breakpoint 2 low address byte is accessed while the result of the last two breakpoint 1 comparisons is gated to interrupt circuitry. This sequence continues and on cycle 5 the result of the breakpoint 2 comparisons is gated to the interrupt circuitry. For logic simplicity, the program counter file register is written into five times, three times with the same low byte information and twice with the high byte. Program counter register "next" is normally written into. Program counter "last" is only written into during an instruction fetch when a Master interrupt is not pending. Thus, during an instruction where a breakpoint occurs, program counter last is written into during the instruction fetch before the breakpoint occurs. Program counter next is written into on the subsequent cycles of the instruction through the next instruction fetch. Thus, when the Master CPU is interrupted, program counter last contains the address of the instruction that was just completed and program counter next contains the address of the instruction that the Slave must return to after register dump and restore routines.

The sequencer is also activated in two other situations. During front panel access a "load" command starts the sequencer. The normal address bus multiplexer is inhibited and instead, the front panel inputs containing the low and high bytes of the front panel address switches is enabled. Instead of loading the program counter file register, the low and high bytes of the address register are sequentially loaded (again five times). The other situation occurs when the front panel "MSTR CPU TEST" switch is activated. Operation is identical to normal sequencer operation except that the address register is loaded from the address bus just as the program counter is loaded. Slave operation is not required. This feature allows front panel viewing of the last address executed by either CPU before a halt and is used for detailed troubleshooting.

Breakpoint and Program Counter Registers

As discussed in the last section, the breakpoints and Slave program counter values are stored in a four word by 8-bit file register. This structure economizes logic for busing, storage and comparison. As discussed, the low and high bytes of each breakpoint are sequentially compared. If the particular breakpoint is enabled for the Slave operation in progress, it is allowed to set the associated interrupt flip-flop. This occurs during cycle 3 of the sequencer for breakpoint 1 and cycle 5 for breakpoint 2. The compare is inhibited during forced jumps or if an interrupt is already pending. Breakpoint registers are loaded only by the Master CPU and cannot be read by an I/O command.

Program Counter value storage allows the Master CPU to trace Slave 2650 program execution. The 2650 has no provision for accessing the internal program counter directly, thus auxiliary logic must be used.

The 2650 also stops when paused after fetching the first byte of the next instruction, making it difficult to determine the address of the last instruction where the breakpoint occurred. Thus the hardware implements storage of both the last instruction address values and the next or just fetched instruction address. The program counter file register contents are read only by the Master CPU.

Note that neither the breakpoint or program counter registers operate unless the Debug mode is enabled. Note also that the sequencer may slow the operation of Slave processors other than the 2650 due to the required 600 nanosecond hold time.

Single Cycle

The single cycle logic consists of a single flip-flop enabled by the control byte. A Slave fetch clocks the flip-flop to an active state, causing a Master interrupt. This automatically pauses the Slave CPU at the end of the single instruction.

Forced Slave Jump

The forced Slave jump is initiated by any one of several sources. The single cycle, either breakpoint, front panel jump logic, or a Master I/O command have the same effect. Two flip-flops generate a 100 nanosecond "JMP CMD" to the Slave CPU which arms its jump mechanism. A jump pending flip-flop is also triggered to disallow a single cycle interrupt. After receipt of the jump command, the Slave outputs a forced branch instruction on the Slave data bus during the next instruction fetch. The absolute jump address is supplied by the Debug jump address register. On the leading edge of the following OPREQ in which the CPU receives the high byte of the jump address, "JMP ACK" is generated. The Debug card uses this signal to gate the address register into the address bus. The Slave multiplexes the high and low bytes of the address bus into the data bus for byte 2 and 3 of the jump instruction and resets "JMP ACK". Receipt of JMP ACK also resets the JMP PENDING flip-flop on the Debug module.

Forced Slave Reset and Forced Interrupt

The Master CPU may command a Slave reset at any time. This is one bit of the command byte. A flip-flop is set on receipt of this command or by system reset. This immediately activates the Slave reset line. The reset is removed when the Master/Slave control flip-flop enables the Slave CPU. This avoids bus conflict since the Slave tries to fetch one byte after reset even through paused and must be held in a reset state until the Master is off the bus.

Slave forced interrupt is simply a pulse generated by bit 1 of the I/O command byte. The interrupt is sent out on the Slave interrupt level zero line. It is stored and processed on the Slave CPU card. This function is not used by the 2650 Slave CPU.

9.3 UTILIZATION

Although the Debug and Front Panel I/O board was designed specifically for use in the TWIN system, it can be used in products of a general nature. External power is required as listed below. The board is designed to be plugged into a 100-pin printed circuit connector.

The following supply voltage is required:

+5 V at 1.3 A

The board should be mounted vertically with side supports to minimize vibration. Sufficient space should be left on both sides of the board to permit heat dissipation. Other components radiating large volumes of heat should not be located near the board.

A list of edge connector pin assignments is found in Tables 9-1 through 9-3.

TABLE 9-1

PIN LIST, EDGE CONNECTOR P1

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	+5 V	Logic Power Input
4	+5 V	Logic Power Input
5-8	not used	
9	gnd	
10	gnd	
11-14	not used	
15	gnd	
16	gnd	
17	$\overline{\text{ADR 0}}$	Address Bit 0 Input/Output
18	$\overline{\text{ADR 1}}$	Address Bit 1 Input/Output
19	$\overline{\text{ADR 2}}$	Address Bit 2 Input/Output
20	$\overline{\text{ADR 3}}$	Address Bit 3 Input/Output
21	$\overline{\text{ADR 4}}$	Address Bit 4 Input/Output
22	$\overline{\text{ADR 5}}$	Address Bit 5 Input/Output
23	$\overline{\text{ADR 6}}$	Address Bit 6 Input/Output
24	$\overline{\text{ADR 7}}$	Address Bit 7 Input/Output
25	$\overline{\text{ADR 8}}$	Address Bit 8 Input/Output
26	$\overline{\text{ADR 9}}$	Address Bit 9 Input/Output
27	$\overline{\text{ADR 10}}$	Address Bit 10 Input/Output
28	$\overline{\text{ADR 11}}$	Address Bit 11 Input/Output
29	$\overline{\text{ADR 12}}$	Address Bit 12 Input/Output
30	$\overline{\text{ADR 13}}$	Address Bit 13 Input/Output
31	$\overline{\text{ADR 14}}$	Address Bit 14 Input/Output
32	$\overline{\text{ADR 15}}$	Address Bit 15 Input/Output
33	CMEM	Slave Memory Access Indication
34	not used	
35	not used	
36	$\overline{\text{DATA 0}}$	Data Bit 0 Input/Output
37	$\overline{\text{DATA 1}}$	Data Bit 1 Input/Output
38	$\overline{\text{DATA 2}}$	Data Bit 2 Input/Output
39	$\overline{\text{DATA 3}}$	Data Bit 3 Input/Output
40	$\overline{\text{DATA 4}}$	Data Bit 4 Input/Output
41	$\overline{\text{DATA 5}}$	Data Bit 5 Input/Output
42	$\overline{\text{DATA 6}}$	Data Bit 6 Input/Output
43	$\overline{\text{DATA 7}}$	Data Bit 7 Input/Output
44-51	not used	

TABLE 9-1

PIN LIST EDGE CONNECTOR P1 (continued)

Pin	Signal	Description
52	$\overline{\text{M}}$/IO	Memory/IO Control Input
53	$\overline{\text{WRP}}$	Write Pulse Input
54	$\overline{\text{OPREQ}}$	Operation Request Input
55	R/$\overline{\text{W}}$	Read/Write Control Input
56	$\overline{\text{HOLD}}$	CPU Hold Output
57	$\overline{\text{JMP CMD}}$	Jump Command Output
58	$\overline{\text{RUN}}$	Run Indication Input
59	$\overline{\text{RESET}}$	System Reset Input
60	$\overline{\text{JMP ACK}}$	Jump Acknowledge Output
61	$\overline{\text{SLV INTACK}}$	Slave Interrupt Acknowledge Input
62	$\overline{\text{INT 0}}$	Slave Interrupt Level 0 Input
63-69	not used	
70	$\overline{\text{SLV CPU INT}}$	Slave Interrupt Input
71-73	not used	
74	$\overline{\text{U PSE}}$	User Pause Input
75	$\overline{\text{SLV OPREQ}}$	Slave CPU Operation Request Input
76	$\overline{\text{SLV RESET}}$	Slave CPU Reset Output
77	$\overline{\text{SLV PSE}}$	Slave Pause Output
78-81	not used	
82	$\overline{\text{FETCH}}$	Slave CPU Fetch Indication
83	$\overline{\text{PAUSE}}$	Master CPU Pause Output
84	$\overline{\text{MAST RUN}}$	Master CPU Run Indication
85	$\overline{\text{SENSE}}$	Slave CPU Sense Input
86	$\overline{\text{FLAG}}$	Slave CPU Flag Output
87	$\overline{\text{F.P. HOLD}}$	Front-Panel Hold Input
88-94	not used	
95	2650 CLK	CLK CPU Clock Input
96-100	gnd	

TABLE 9-2

PIN LIST, EDGE CONNECTOR P2

Pin	Signal	Description
1	$\overline{\text{ADR 15}}$	Address Bit 15 Output
2	$\overline{\text{ADR 14}}$	Address Bit 14 Output
3	$\overline{\text{ADR 13}}$	Address Bit 13 Output
4	$\overline{\text{ADR 12}}$	Address Bit 12 Output
5	$\overline{\text{ADR 11}}$	Address Bit 11 Output
6	$\overline{\text{ADR 10}}$	Address Bit 10 Output
7	$\overline{\text{ADR 9}}$	Address Bit 9 Output
8	$\overline{\text{ADR 8}}$	Address Bit 8 Output
9	$\overline{\text{ADR 7}}$	Address Bit 7 Output
10	$\overline{\text{ADR 6}}$	Address Bit 6 Output
11	$\overline{\text{ADR 5}}$	Address Bit 5 Output
12	$\overline{\text{ADR 4}}$	Address Bit 4 Output
13	$\overline{\text{ADR 3}}$	Address Bit 3 Output
14	$\overline{\text{ADR 2}}$	Address Bit 2 Output
15	$\overline{\text{ADR 1}}$	Address Bit 1 Output
16	$\overline{\text{ADR 0}}$	Address Bit 0 Output
17	not used	
18	$\overline{\text{OPREQ}}$	Operation Request Output
19	not used	
20	not used	
21	$\overline{\text{FPD 7}}$	Front-Panel Data Bit 7
22	$\overline{\text{FPD 6}}$	Front-Panel Data Bit 6
23	$\overline{\text{FPD 5}}$	Front-Panel Data Bit 5
24	$\overline{\text{FPD 4}}$	Front-Panel Data Bit 4
25	$\overline{\text{FPD 3}}$	Front-Panel Data Bit 3
26	$\overline{\text{FPD 2}}$	Front-Panel Data Bit 2
27	$\overline{\text{FPD 1}}$	Front-Panel Data Bit 1
28	$\overline{\text{FPD 0}}$	Front-Panel Data Bit 0
29	$\overline{\text{CMEM SW}}$	Common Memory Switch Input
30-34	not used	Slave
35	$\overline{\text{DTA 0}}$	Front-Panel Display Bit 0
36	$\overline{\text{DTA 1}}$	Front-Panel Display Bit 1
37	$\overline{\text{DTA 2}}$	Front-Panel Display Bit 2
38	$\overline{\text{DTA 3}}$	Front-Panel Display Bit 3
39	$\overline{\text{DTA 4}}$	Front-Panel Display Bit 4
40	$\overline{\text{DTA 5}}$	Front-Panel Display Bit 5
41	$\overline{\text{DTA 6}}$	Front-Panel Display Bit 6
42	$\overline{\text{DTA 7}}$	Front-Panel Display Bit 7
43-50	not used	

TABLE 9-3

PIN LIST, EDGE CONNECTOR P3

Pin	Signal	Description
1	+5 V	Logic Power Input
2	+5 V	Logic Power Input
3	gnd	
4	gnd	
5	gnd	
6	$\overline{\text{INTACK}}$	Interrupt Acknowledge
7	$\overline{\text{FETCH}}$	Fetch Indication
8	$\overline{\text{RUN}}$	CPU Run Indication
9	MSTR/$\overline{\text{SLAVE}}$	Master/Slave Enable Indication
10	$\overline{\text{HOLD}}$	CPU Hold Indication
11	MEM/IO	Memory/IO Control Input
12	not used	
13	$\overline{\text{MSTRST}}$	Master CPU Test Switch Input
14	$\overline{\text{RSET}}$	Reset Switch Input
15	JMP	Jump Indication
16	$\overline{\text{ENT}}$	Enter Switch Input
17	INC	Increment Switch Input
18	DEC	Decrement Switch Input
19	$\overline{\text{LOAD}}$	Load Switch Input
20	SET SLV	Slave Select Switch Input
21	$\overline{\text{SET MSTR}}$	Master Select Switch Input
22	RST HOLD	Reset Hold Input
23	MSTR INT	Master Interrupt Input
24	$\overline{\text{ACCESS}}$	System Access Indication
25	MEM/$\overline{\text{IO}}$	Memory/IO Indication
26	OP RSET	Operation Reset Input
27	not used	
28	not used	
29	not used	
30	CMEM	Common Memory Access Input

CHAPTER 10

MOTHER BOARD AND POWER SUPPLIES

10.0 INTRODUCTION

The Mother board distributes operating power, data, addresses, and control signals between other boards in the system. It is also the receptacle for system boards through the use of printed circuit connectors mounted on the top side.

The system contains two modular power supplies which provide operating power for system modules and for the TWICE assembly.

10.1 THE MOTHER BOARD

The Mother board carries the system buses and has provisions for mounting up to 20 boards. The board is divided into two sections: Master and Slave. The Front-Panel I/O and Debug board forms the division between the two sections and is common to both sections. The Master section contains the PROM Programmer boards, which are plugged into the two connectors at the left side of the Mother board. These positions are keyed mechanically so that no other boards can be plugged into their place. The other boards which must be installed on the Master side are the Master CPU and Master Memory. Optionally, a General-Purpose I/O board can be installed on the Master side. The Slave side must contain the Slave CPU and Slave Memory boards. Optionally, a General-Purpose I/O board may be installed on the Slave side.

The system bus carried by the Mother board is split into sections also. Data, Control, and Address buses are common to all boards. Interrupts are split so that each CPU has its own set of interrupt lines.

10.2 POWER SUPPLIES

Operating power for the system is provided by two modular power supplies mounted in the development computer chassis. Outputs are connected to, and distributed by, the Mother board. One supply has three outputs: +12 VDC, -12 VDC, and a common ground line which is tied to a separate return ground on the Mother board bus. The second supply has a DC output of +5 V, and AC outputs of 50 VAC, 30 VAC, and 15 VAC for the PROM Programmer boards.

APPENDIX A

FLOPPY DISK CONTROLLER

GENERAL

The controller performs all disk drive signal sequencing, including track seeking. The controller acts as an interpreter for the driver program. Communication between the driver and controller proceeds as described later.

SYSTEM BLOCK DIAGRAM

Figure A-1 shows the overall system configuration and the functions provided in each block. It should be noted that the controller may be used with any computer I/O having two input bytes and two output bytes. (May be one 16-bit I/O port). Figure A-2 shows I/O buses.

CONTROLLER ARCHITECTURE

The Controller consists of 2 basic blocks (see Figure A-3):

a) A Sector Buffer, which is used for bidirectional (Read/Write) Data Storage.

b) A small ROM driver microcontroller used for overall control, CRC generation and checking, sector seeking, and sector data formatting.

The data exchange from the CPU to the disk proceeds in two steps:

a) The Sector Buffer is loaded from the disk (read) or CPU (write).

b) The CPU (read) or the disk (write) receives the data from the sector buffer.

In both of the above cases, the driver program must present to the controller the track and sector addresses, which are used as a part of the sector preamble as well as for the track and sector seeking operation.

It should be noted that along with the aforementioned data transfers, there will be other transfers of command and status bytes between the CPU and the controller before, during, and after the actual sector data transfer.

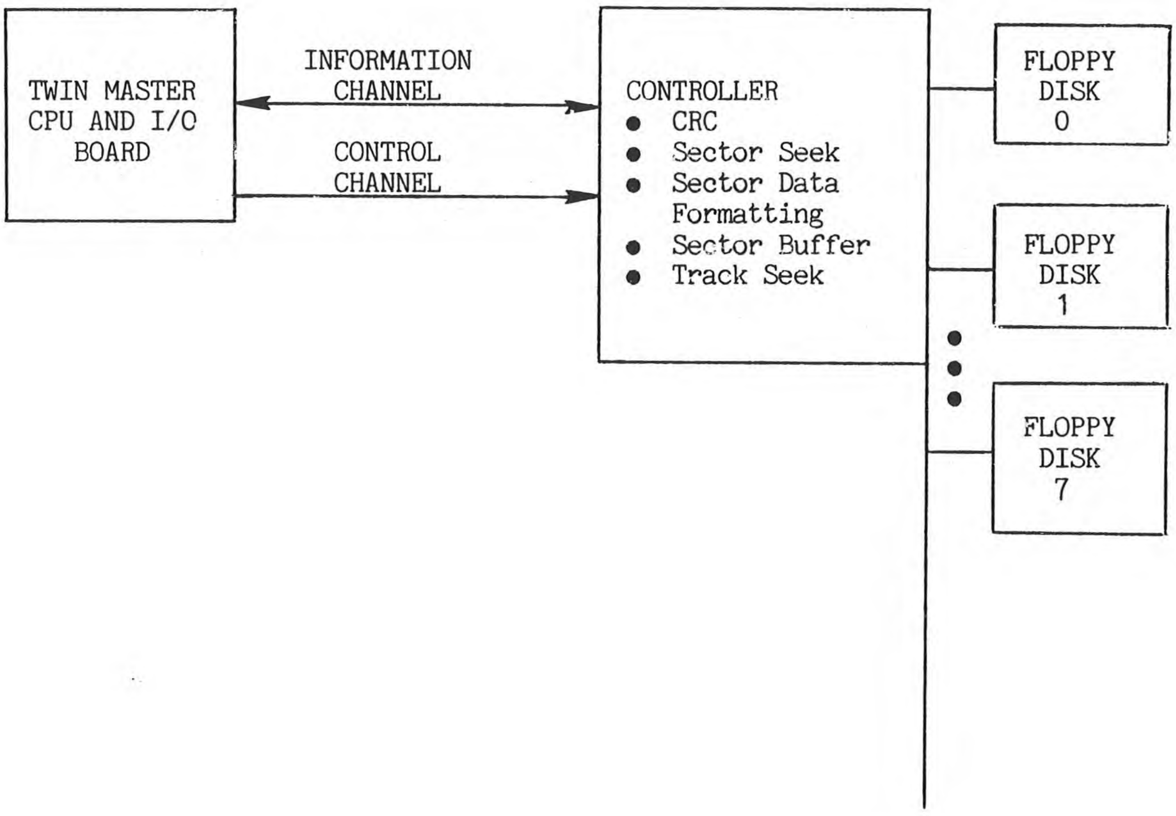
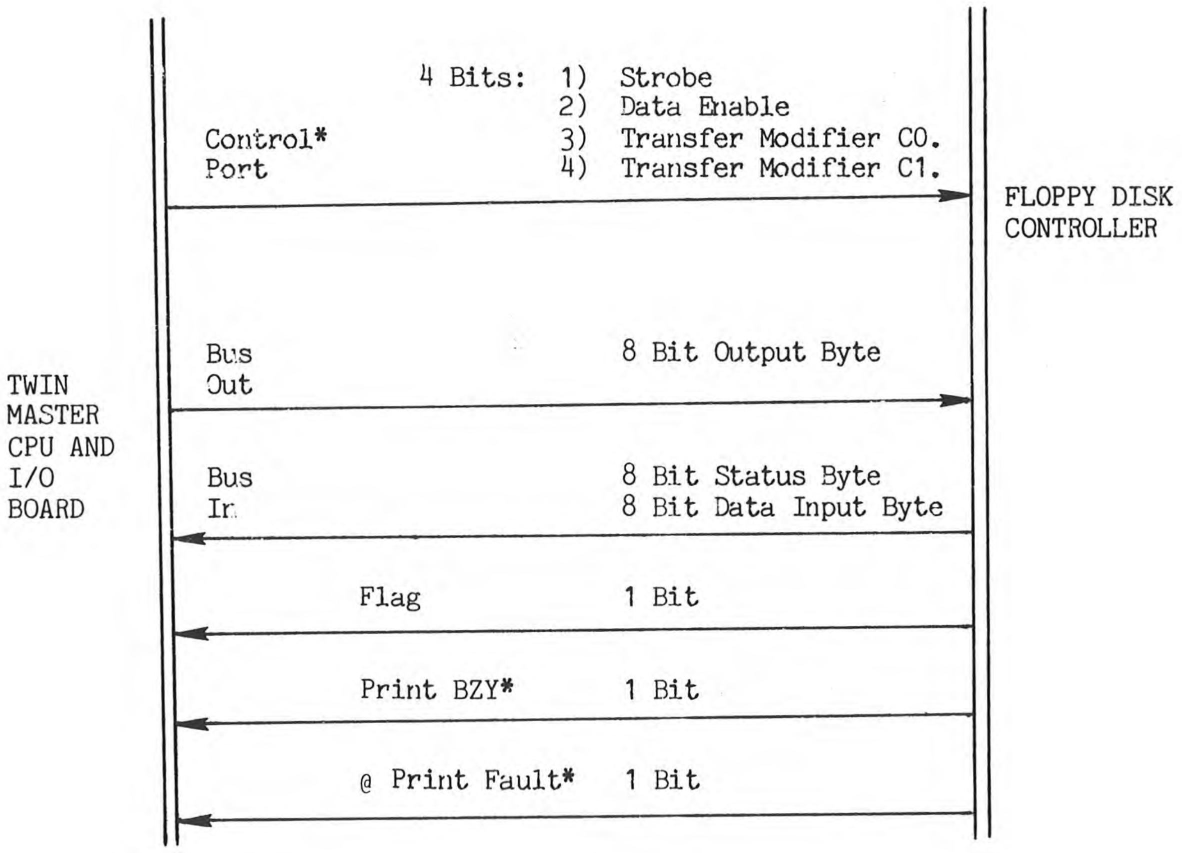


FIGURE A-1

BLOCK DIAGRAM OF FLOPPY DISK SYSTEM

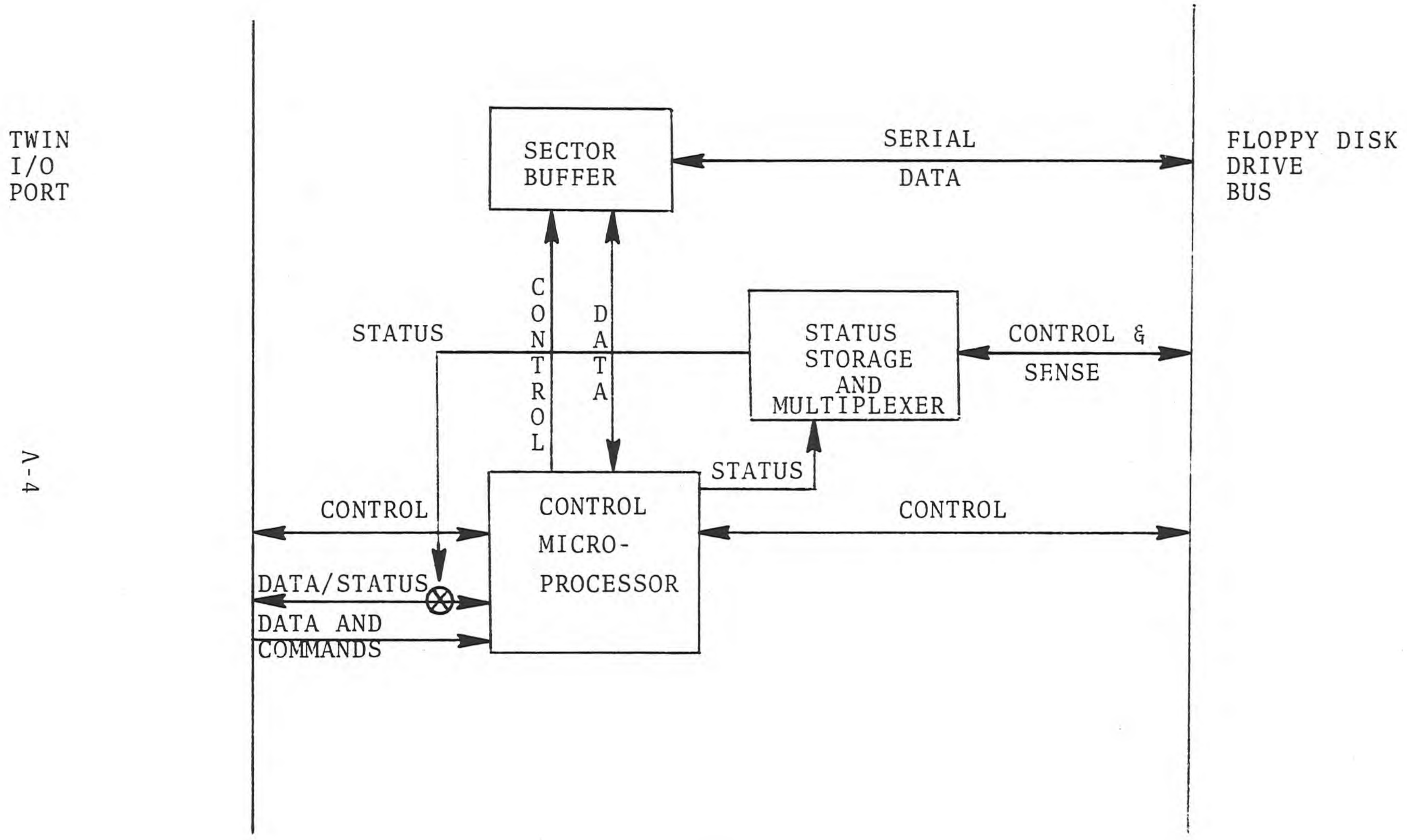


* Negative True
Level at this interface

@ Fault = Not Connected
Paper Out
Not Selected

FIGURE A-2

I/O INFORMATION EXCHANGE BUSES



CONTROLLER ARCHITECTURE

FIGURE A-3

CONTROLLER CIRCUIT DESCRIPTION

This section describes the circuits, the associated logic implementation and functional blocks of the controller (see Figure A-4). A generalized timing diagram is provided to indicate the timing sequences of the data and control paths during the execution of the instruction (see Figure A-5).

FUNCTIONAL BLOCK DESCRIPTION

1) Register File. There are 16 8-bit registers designated, R0 through R16. The outputs of the register are OR-tied to the WR-BUS and it is the A port of the ALU. The inputs to the register file are from the T-BUS.

2) Accumulator. This is an 8-bit LATCH. The input is tied to the T-BUS and the output is connected to the B port of the ALU.

3) ALU. All logic and arithmetic functions are provided by this block. The output of the ALU is the input to the T-Register.

4) T-Register. The T-Register is an 8-bit latch. It serves as the slave register for all operations. It is the source register for all output group instructions.

5) Jump Condition. This portion of the logic verifies the JUMP instruction condition and enables the JUMP when the condition is valid.

6) Carry Input Generator. A carry will be generated based on the instruction type.

7) ROM Address Register - RAR. This is an 8-bit programmable counter. It stays in the up count mode for all operations except for the JUMP instruction which loads the pointer address from the WR-BUS. The output addresses the ROM to source the content as microcode to the RIR register.

8) Control Memory. This consists of ROM register (512 x 16-bit) and RIR register (16-bit) which holds the current instruction word from the ROM. At the beginning of each cycle the new ROM instruction word is clocked into the RIR register. The output is the logic input source for the Jump, Control, and Decoding Logics.

9) Command Decoder. The outputs from the RIR are decoded and status/control lines are generated.

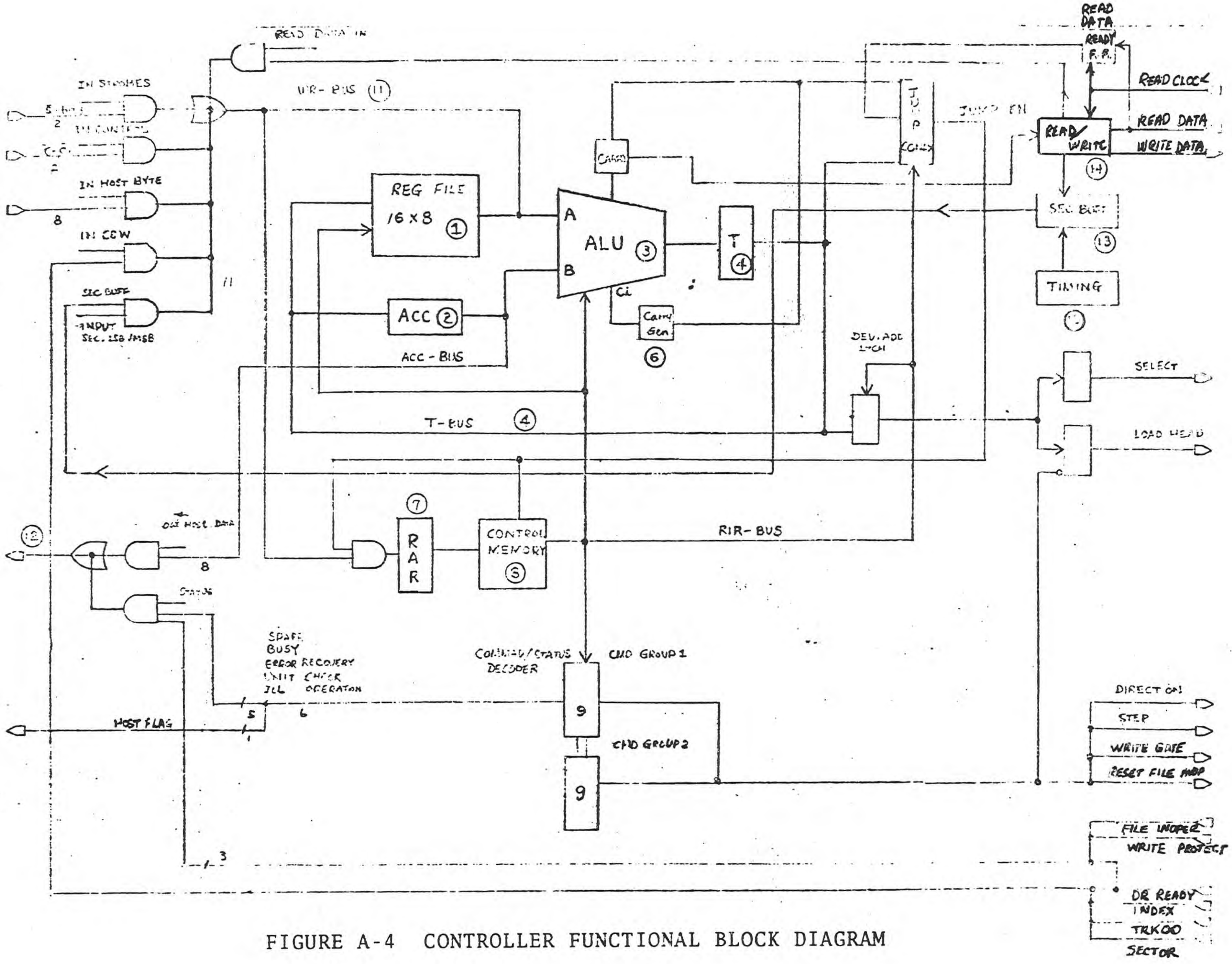
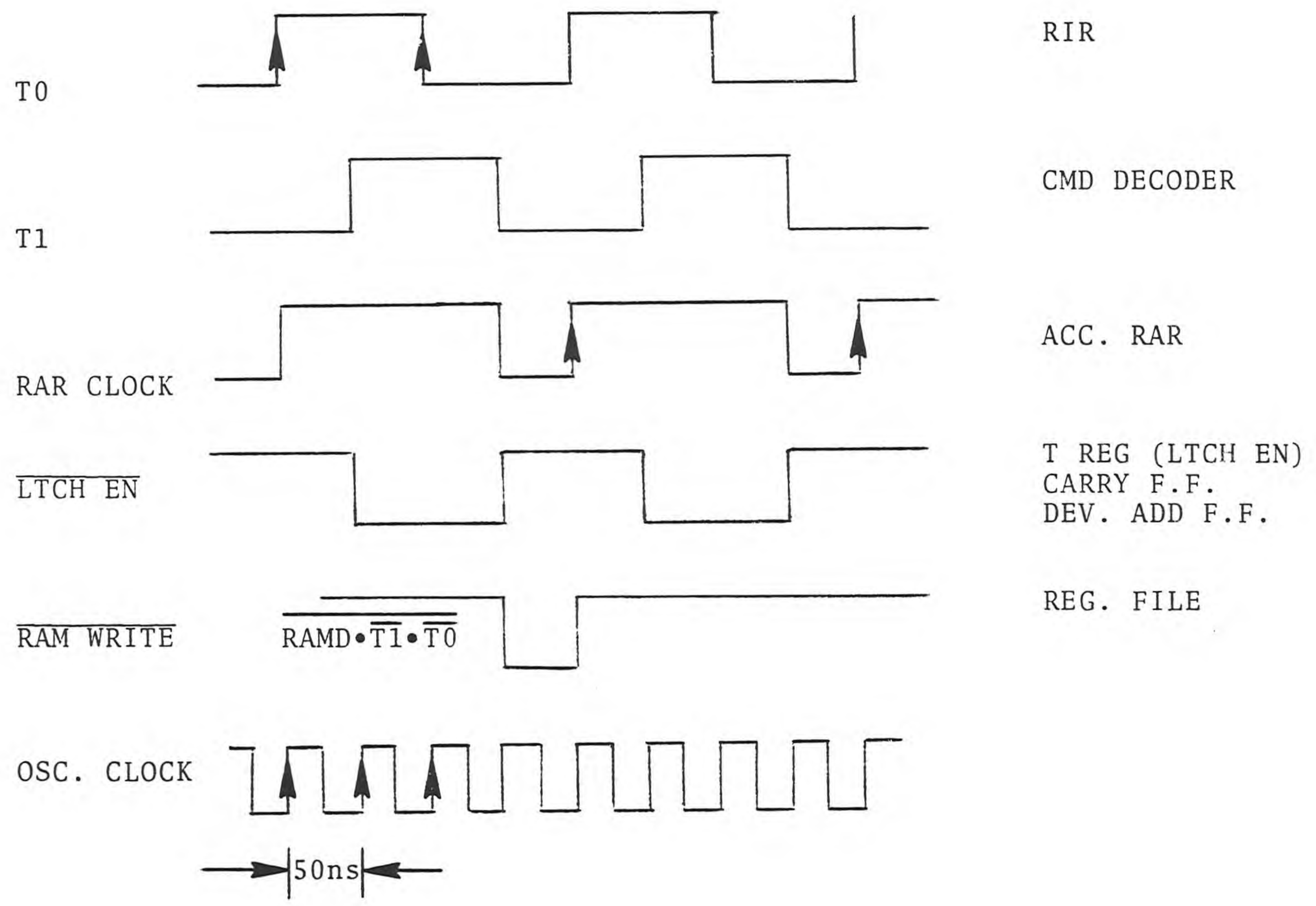


FIGURE A-4 CONTROLLER FUNCTIONAL BLOCK DIAGRAM



T0
RIR
T1
CMD DECODER
RAR CLOCK
ACC. RAR
$\overline{\text{LTCH EN}}$
T REG (LTCH EN)
CARRY F.F.
DEV. ADD F.F.
$\overline{\text{RAM WRITE}}$
$\overline{\text{RAMD} \cdot \overline{\text{T1}} \cdot \overline{\text{T0}}}$
REG. FILE
OSC. CLOCK
50ns

FIGURE A-5

CONTROLLER TIMING DIAGRAM

10) <u>Timing</u>. A 20-MHz crystal controlled oscillator is used to provide the 200 ns cycle timing with the listed control clock: *

a) RAR CLOCK

b) $\overline{\text{RAM WRITE}}$

c) T0

d) T1

e) $\overline{\text{LTCH}}$

11) <u>WR-BUS</u>. The (Working Register) WR-BUS provides input to the A port of the ALU. It is an open-collector bus and information is gated to the WR-BUS from the following sources:

Sources	No. of Bits
Sector Buffer	1
Host Data in Byte	8
Host Control Bit	4
Drive Timing/Status	4
Read Data Flip-Flop	1

12) <u>Output Bus</u>. The output bus provides either the status byte or the data byte to the host input bus.

13) <u>Sector Buffer</u>. The Sector Buffer will store 1024 bits of input/output data information from the host drive. The data byte received from the host is serialized by the microcode and tranferred to the Sector Buffer via the Carry Flip-Flop. The read data bit received from the drive is stored into the Sector Buffer via the Carry Flip-Flop. After 1024 bits of data is received, the microcode routine retrieves the data from the Sector Buffer and deserializes the bits to form an 8-bit byte to transfer to the Host.

14) <u>Read/Write Circuit</u>. The double-frequency horizontal non-return to zero (NRZI) method of recording is used to send Write Data to the Drive. The timing and generation of clock and data bits are controlled by the microcode. The Read Circuit detects the starting of the information and transfers the data bit to the WR-BUS.

* See the timing diagram.

FLOPPY DISK COMMANDS

The floppy disk is controlled from a set of command bytes passed to the floppy disk controller. The mechanism to pass the command bytes is defined later.

Write Unformatted

This command allows the controller to accept a three byte device and seek address and 128 bytes of data.

After the 128th data byte is received, the drive seeks to the appropriate track and sector and writes the address ID and data on that sector, thereby formatting a sector. If subsequent data bytes are presented to the controller without a new command sequence, the next sequential sector is formatted. Data must be presented in 128 byte increments. Automatic track switching may occur.

Write Formatted

This command allows the controller to accept a three byte device and seek address and 128 bytes of data. After the 128th byte of data is received, the drive seeks to the appropriate address, verifies the address ID, and writes the data into that sector. Subsequent 128 byte data byte presentation without a new command sequence writes in the next sequential sector. Automatic track switching may occur.

Read Command

This command allows the controller to accept a three byte device and seek address, seeks to the appropriate track and sector, and reads 128 bytes of data. Sequential sectors can be read by continuing the data reads in 128 byte blocks without issuing a new command sequence. Automatic track switching may occur. Non-sequential partial records can be read from the sector buffer.

Read Diagnostic Command

This command reads the controller working registers. Reading starts at register 1 and continues through register 0. Reading can stop at any time. If more than 16 registers are read, a wraparound will occur.

Print Command

There is no direct print command. Print data transfer is accomplished by using the PTS control byte command. Data is passed to the printer until a carriage return character is recognized by the printer. At this time the printer prints the received data and performs a carriage return.

DEVICE ASSIGNMENT

The floppy disk/printer controller uses two device addresses, each with an associated input/output port pair. The two devices are used for control and data transfer functions.
Device EA Control Device
Device EB Data Device

Control Device (EA)

The control device provides the following functions for the floppy disk/printer controller.

-Control byte transfer
-Flag byte input

The control byte write sends one of five control bytes to the controller.

-SIO	Start an I/O
-OTD	Output Data
-IND	Input Data
-RST	Reset flag bit upon operation complete
-PTS	Strobe data to the printer

The transfer of the control byte is done as follows:

```
WRTE,r V
```

r = the general purpose register containing the control byte
V = the control device address

Flag byte input transfers the controller flag byte to the CPU. The flag byte transfer is accomplished as follows:

```
REDE,r V
```

r = the general purpose register to receive the sense byte
V = the control device address

Data Device (EB)

The data device provides the following function for the floppy disk/printer controller:

-Write data transfer
-Read data transfer
-Read status transfer

The write data sequence is used to place a data byte in the data device output port. The data is sequence dependent and will consist of a command byte to the controller, drive, track and sector addresses, and/or data. Placing data in the device output port is as follows:

```
WRTE,r V
```

r = the general purpose register that contains the data
V = the data device address

The read data sequence accepts data from the controller into the data device input port. The read sequence is as follows:

```
REDE,r V
```

r = the general purpose register to receive the data
V = the data device address

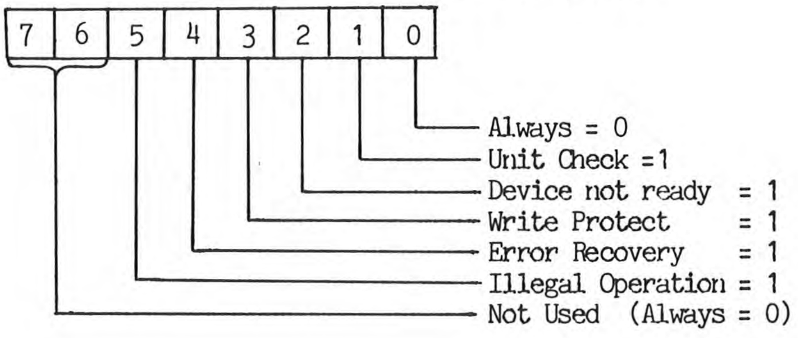
The read status sequence is identical to the read data sequence. However, status presentation only occurs after a SIO, RST, or PTS control byte has been written to the control device output port.

DISK STATUS BYTE

The disk status byte contains information pertinent to the operability and usability of the disk controller and devices. The status byte is read from the input port of the data driver and is available only after one of the following control bytes have been placed in the output port of the control device.

-SIO	Start I/O
-RST	Reset flag
-PTS	Printer strobe

The disk status byte has the following format:



Unit Check -- Indicates disk failure

Device Not Ready -- Indicates the selected drive has one or all of the following faults:
- 24V missing
- Door open
- Power off
- Diskette not up to speed

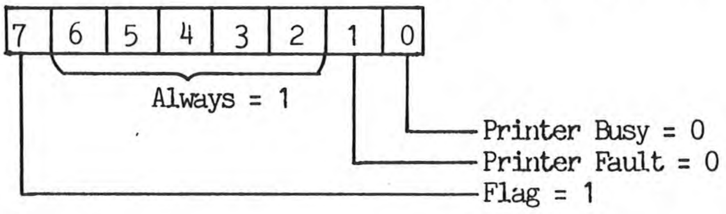
Write Protect -- Indicates the selected drive is in the write protect mode. Any write command to the drive functions normally but no data will be written on the disk.

Error Recovery -- This bit indicates the controller has gone into an error recovery mode. The bit will reset when the controller finishes.

Illegal Operation -- Indicates the selected drive track or sector address are out of the specified limits. The limits are track 0-76, sector 0-31.

CONTROLLER FLAG BYTE

The controller flag byte is read from the input port of the control device. This flag information read is described below:



Flag -- This bit serves two functions when it is set to a 1 condition by the controller.

Read Operation -- Indicates the controller has a data byte for the CPU.

Write Operation -- Indicates the controller is ready to receive a data byte from the CPU.

The flag is reset when the controller completes its operation by the RST control command.

Printer Fault -- This bit indicates the printer controller has an error condition. (Power off, out of paper, or disconnected). The bit is reset by correcting the error condition. As long as any of the above conditions prevail, the bit will remain set.

Printer Busy -- This bit is true when the printer is active printing or moving paper.

CPU/CONTROLLER DATA EXCHANGE PROTOCOL

The interface lines required to support the control and data exchange between the CPU and the controller are shown in Figure A-2. Timing is shown in Figures A-6 and A-7.

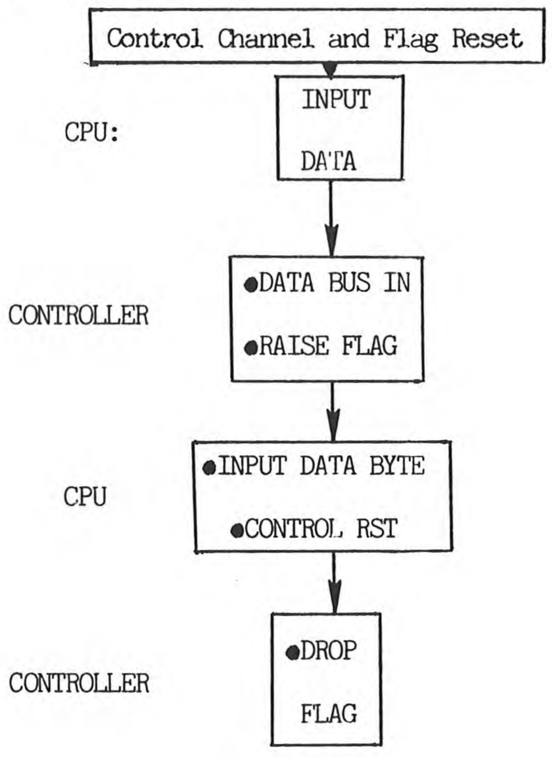
The input and output ports are used for the exchange of input data and controller status (input port), and output data and controller commands (output port).

The definition and formats of the information with the exception of data, which is unrestricted, are given below. The control port is given first since it defines the nature of the information exchange ensuing at the interface.

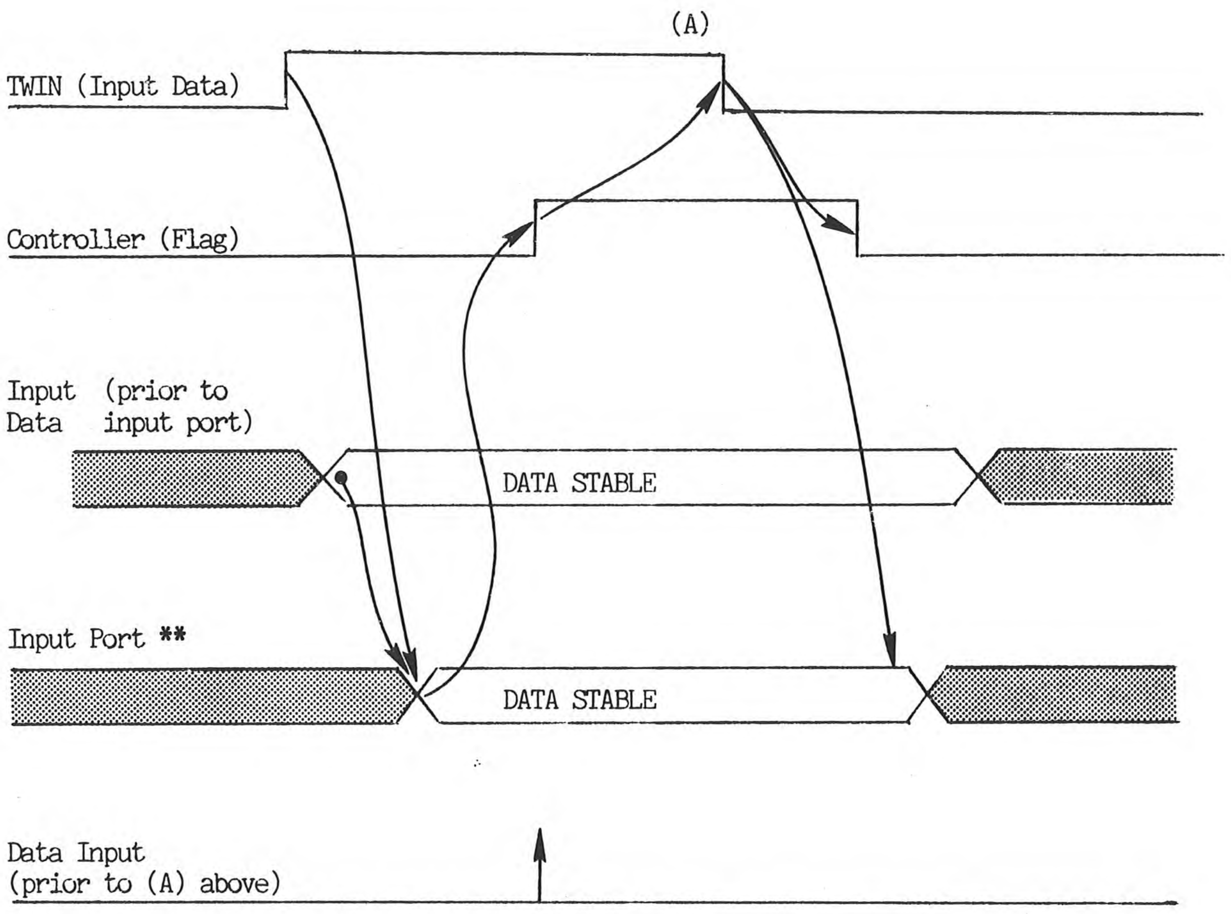
CONTROL PORT OPERATION

Input Data Sequence

Flowchart at the Interface



INPUT DATA TIMING

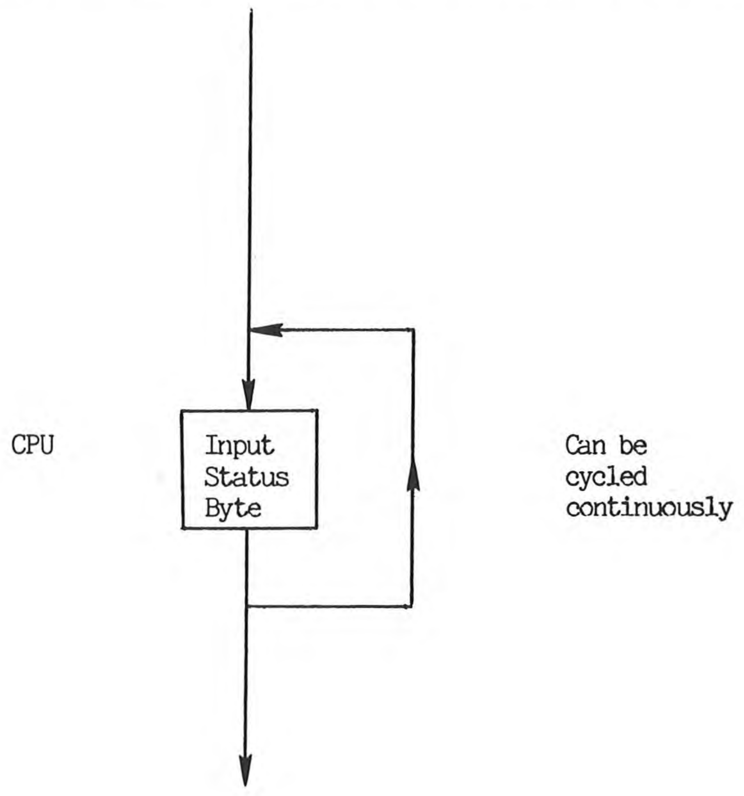


** Status Byte present before and after data stable on input port

FIGURE A-6

INPUT STATUS SEQUENCE

The control port must be in a state to allow status to be gated to bus.



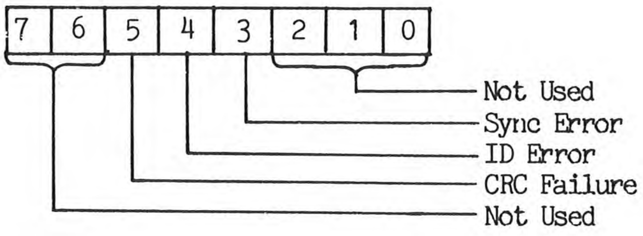
Note: No controller response is required.

FLOPPY DISK DIAGNOSTIC REGISTERS

The floppy disk diagnostic registers are read using the diagnostic read command. The register contents are as follows:

Register	Usage
R0	Sense Byte
R1	Working Storage
R2	Working Storage
R3	Program State Flags
R4	Device Address
R5	Head Delay Counter
R6	Command
R7	Sector N (current)
R8	Controller Status Word
R9	Target Sector
R10	Current Track Head Position
R11	Target Track
R12	Working Storage
R13	Host Byte Counter
R14	CRC 1 Byte
R15	CRC 2 Byte

R0 Sense Byte Format

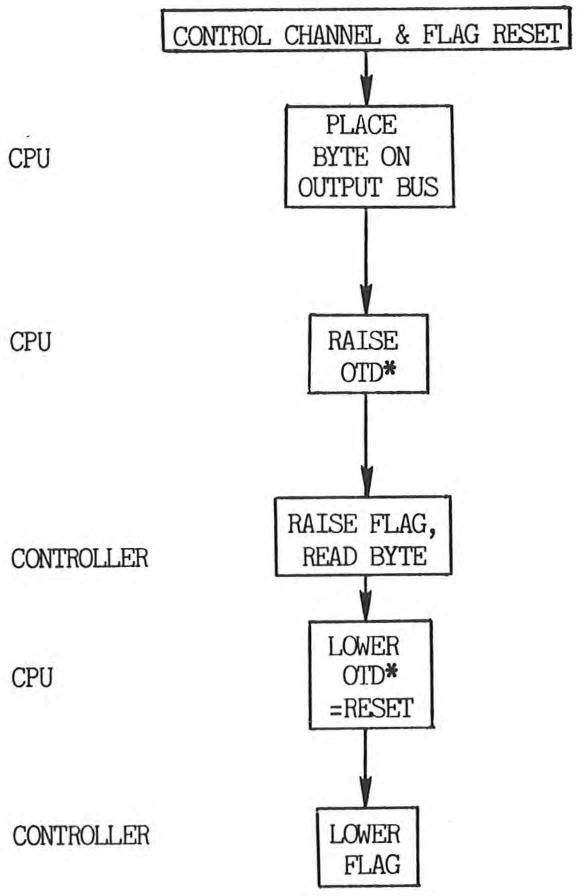


INTERRUPTS

The interface logic for the floppy disk controller has the ability to supply or suppress program interrupts during I/O sequences. Interrupts occur after the fall of the Flag control line. To enable interrupts, the floppy disk enable interrupt bit in the Floppy Disk control byte must be set. When interrupts are enabled, the next fall of the Flag control line will produce an interrupt. Interrupts are vectored indirectly to the user service routine. The interrupt vector and priority are assigned by the hardware.

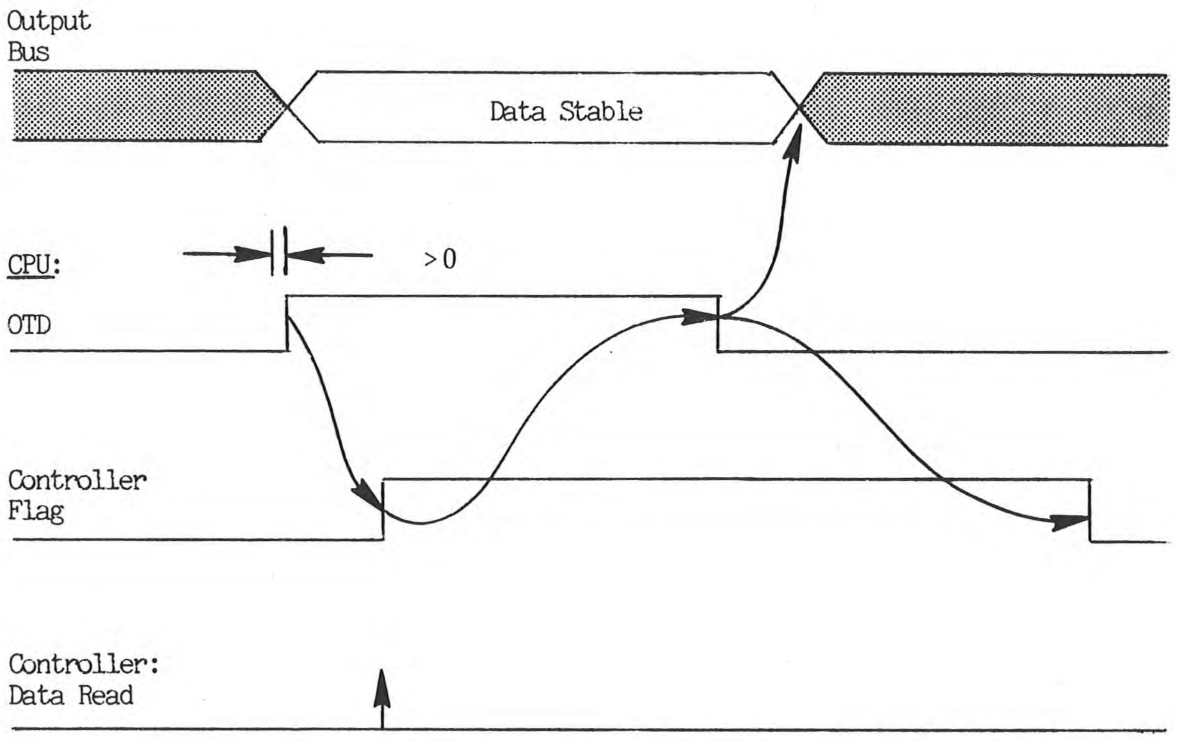
Printer interrupt capabilities are the same as the floppy disk except a different priority and vector are assigned. Also a unique interrupt enable bit is defined in the Floppy Disk Control byte.

Output Sequence (commands are a part of the output data stream.)



* OTD = Output Data Byte

DATA OUTPUT TIMING



* Status Byte on Input Data Port Before and After the OTD Pulse

FIGURE A-7

Print Command (on Control Port)

This is a control port command that allows the host to command the controller to effect a Print Strobe.

The Data on the Output Port is ignored by the controller.

STATUS/SENSE BYTES

General

These bytes are used to convey to the system the following classes of information:

Status Byte

a) Information as to the state of the peripheral device:

- Write protect
- Drive not Ready

b) Information as to the state of the I/O operation:

- Error Recovery
- Unit check (indicates a failure)
- Illegal Operation (track or sector out of limits)

Sense Byte

Information as to the disposition of the failure of an I/O transaction.

Only used for Diagnostic purposes. Sense Byte is contained in register 0. It is the last register whose contents are output during the diagnostic read.

- CRC failure
- ID error
- SYNC error

Encoding of the status and sense bytes is defined in the sections pertinent to the peripheral device.

Diagnostic Read Data Bytes

This command allows the host to interrogate the controller's internal working registers.

R1	Working Storage	Byte 1
R2	Working Storage	Byte 2
R3	Flag Byte	Byte 3

R4	Drive Number	Byte 4
R5	Head Delay	Byte 5
R6	Command Register	Byte 6
R7	Current Sector	Byte 7
R8	Controller Status Word	Byte 8
R9	Target Sector	Byte 9
R10	Current Track Head Position	Byte 10
R11	Target Track	Byte 11
R12	Working Storage	Byte 12
R13	Host Byte Counter	Byte 13
R14	CRC 1 Byte	Byte 14
R15	CRC 2 Byte	Byte 15
R0	Sense Byte	Byte 16

HOST-CONTROLLER COMMUNICATIONS PROTOCOL

Operation Initiation Sequence

I/O operations are initiated by sending a sequence to the controller which consists of a variable length string of bytes, defining the desired operation. The general sequence is as follows:

CONTROL BYTE	DATA BYTE
SIO	Initialize
OTD	Command
OTD	Drive #
OTD	Sector #
OTD	Track #

Start I/O

This is a control port byte which tells the controller that a new I/O operation is requested. The byte which is in the output port during the next OTD data out cycle defines the command which is to be executed.

Task Descriptor Heading

As described above the first byte in the heading is the command byte, which implicitly defines the length of the Task Descriptor.

Single Byte Headings

This class of descriptor is used for the following I/O operations:

- Loop Back
- Diagnostic Read

Multibyte Headings

This is the normal sequence and consists of a series of bytes following the command byte. The first is the device number from hex 00 to FF, and is described in the next section.

Bytes subsequent to the device address are defined for a particular command/ device address combination, which is further defined in the next section.

Device Address Space

Floppy Disk Drives (0-7) -- These drives are selected during the command Initiation Sequence and remain selected for the complete Disk I/O sequence. It should be noted that for these addresses the task descriptor will contain the sector and track bytes to be operated upon .

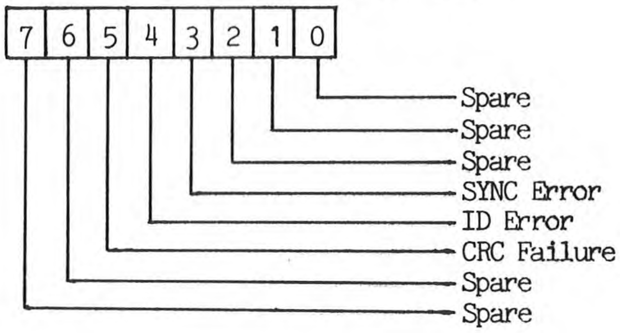
Data Transfer Sequence

The data transfer sequence is a function of the class of device selected as well as the command. For disk I/O this is detailed in the next paragraph. For the Centronics printer it is given later.

DISK I/O OPERATIONS

Sense Command (See also Diagnostic Read)

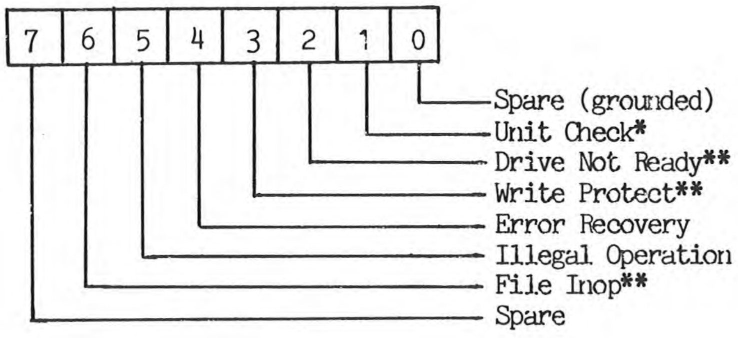
The sense byte is contained in Register R0.



SENSE BYTE

Status Byte Definition

The status byte is available to the TWIN system as described previously and is encoded as follows:



STATUS BYTE

NOTE: A negative assertion level is normal at the interface. This allows the host to receive bad status for loose controller and TWIN system interconnections.

*The unit check status bit tells the host that the I/O operation execution malfunctioned. Determination of the malfunction can be obtained by a Diagnostic Read and 16 IND's.

**These status bits originate from the drive; the others originate from the microprocessor.

READ COMMANDS

The description of these commands and their associated descriptors is given below.

Normal Read

The sequence of bytes in the descriptor is as follows:

CONTROL BYTE	DATA BYTE
SIO	-
OTD	COMMAND (READ TYPE)
OTD	DRIVE NUMBER
OTD	SECTOR NUMBER
OTD	TRACK NUMBER
IND	BYTE 0
.	.
.	.
IND	BYTE 127

Following the descriptor, 128 data bytes are read into the sector buffer. If more than 128 bytes are required for one descriptor, they will be read from the next sequential sector and track. This will be communicated by another OTD after receipt of the 128th byte from the sector buffer.

It should be noted that complete status reads should not be made until the device is completely selected.

Diagnostic Read

This command has the following description:

Control Byte	Data Byte
SIO	-
OTD	Diag. Read
IND	R1
IND	R2
.	
.	
.	
IND	R15
IND	R0

As can be seen from above, the 16 working registers can be read in, starting from R1. Reading can terminate at any time. If more than 16 registers are read, a wraparound will occur, that is, R1 ... etc., will be read again.

WRITE COMMANDS

Copy Write

This command is similar to the normal write except the data is the current contents of the sector buffer.

CRC is updated for this command as the track and sector bytes are included in the CRC calculations.

Write Unformatted

This command allows the writing of unformatted disks. The main difference between this and a normal write is that the read before write is omitted.

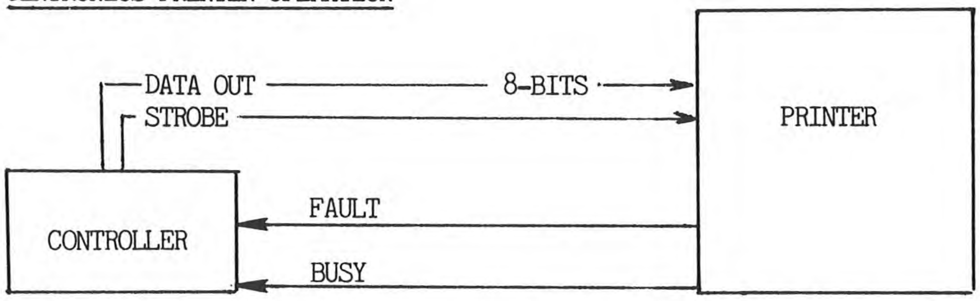
Normal Write

The sequence of bytes in the descriptor is as follows:

CONTROL BYTE		DATA BYTE
SIO	-	-
OTD	-	COMMAND (WRITE TYPE)
OTD	-	DRIVE NUMBER
OTD	-	SECTOR NUMBER
OTD	-	TRACK NUMBER
OTD	-	DATA BYTE 0
.		.
.		.
.		.
OTD	-	DATA BYTE 127

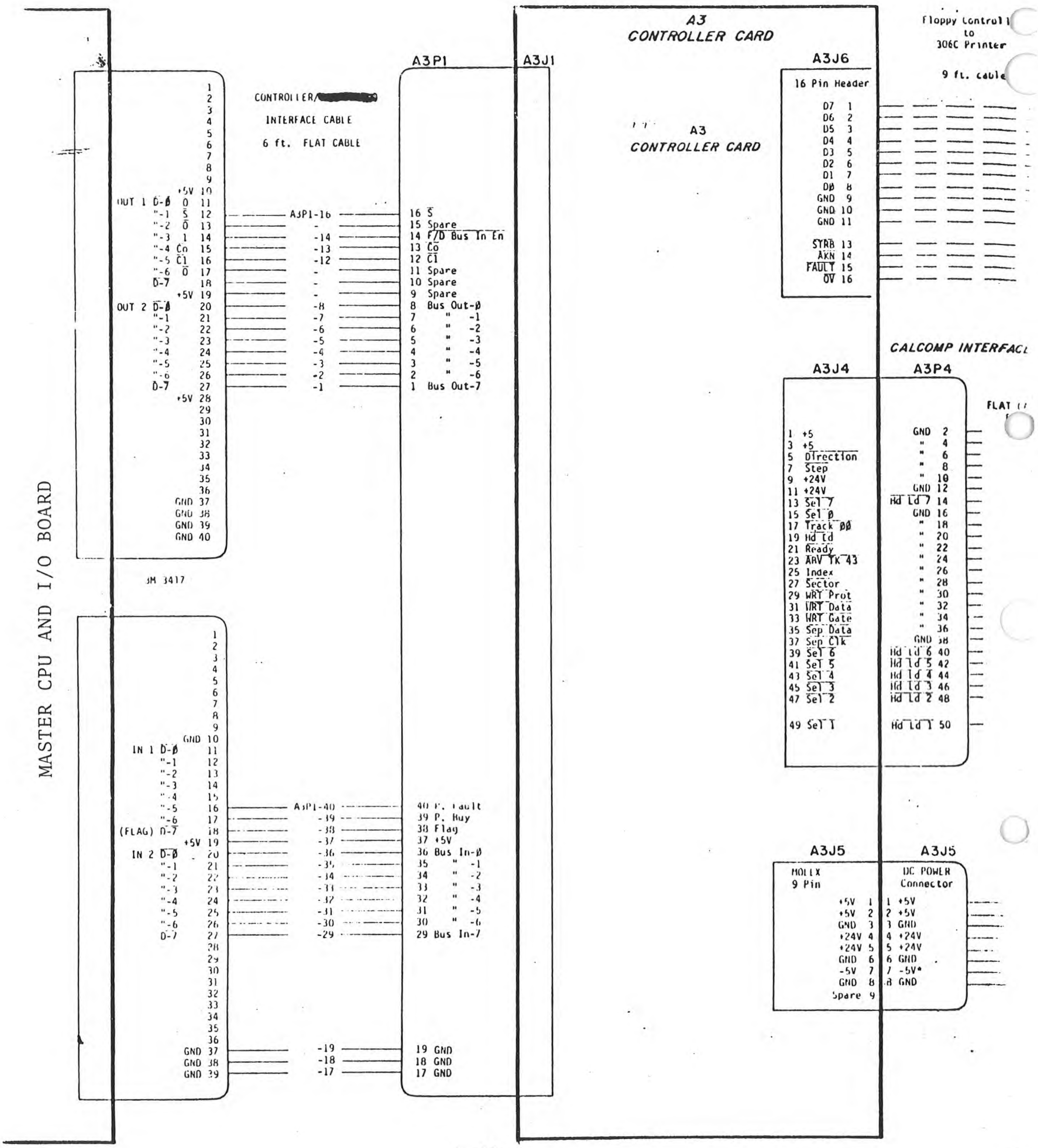
After receipt of the 128th byte, the controller will write the data to the disk. If subsequent bytes are presented to the controller without a new task descriptor, they will be written on the next sequential sector and track. <u>It should be noted that the data blocks must come in multiples of 128</u>. A read ID is performed by the controller prior to the write to ensure correct track position.

<u>CENTRONICS PRINTER OPERATION</u>



The interface to the Centronics Printer contains 8 Data output lines, a data Strobe line, and two printer status lines, Fault and Busy.

The functional description and codes required to operate the printer are given in section 4 of the technical manual Centronics Model 306.


MASTER CPU AND I/O BOARD
A3 CONTROLLER CARD
A3P1
A3J1
CONTROLLER/[redacted] INTERFACE CABLE
6 ft. FLAT CABLE
OUT 1 D-Ø … D-7
OUT 2 D-Ø … D-7
+5V 10
+5V 19
+5V 28
GND 37
GND 38
GND 39
GND 40
3M 3417
A3P1-16
16 S̄
15 Spare
14 F/D Bus In En
13 C̄o
12 C̄1
11 Spare
10 Spare
9 Spare
8 Bus Out-Ø
7 " -1
6 " -2
5 " -3
4 " -4
3 " -5
2 " -6
1 Bus Out-7
IN 1 D-Ø … (FLAG) D-7
IN 2 D-Ø … D-7
GND 10
+5V 19
A3P1-40
40 P. Fault
39 P. Buy
38 Flag
37 +5V
36 Bus In-Ø
35 " -1
34 " -2
33 " -3
32 " -4
31 " -5
30 " -6
29 Bus In-7
19 GND
18 GND
17 GND
Floppy Controll to 306C Printer
9 ft. cable
A3J6
16 Pin Header
D7 1
D6 2
D5 3
D4 4
D3 5
D2 6
D1 7
DØ 8
GND 9
GND 10
GND 11
STRB 13
AKN 14
FAULT 15
OV 16
CALCOMP INTERFACE
A3J4
A3P4
FLAT
1 +5
3 +5
5 Direction
7 Step
9 +24V
11 +24V
13 Sel 7
15 Sel Ø
17 Track ØØ
19 Hd Ld
21 Ready
23 ARV TK 43
25 Index
27 Sector
29 WRT Prot
31 WRT Data
33 WRT Gate
35 Sep Data
37 Sep Clk
39 Sel 6
41 Sel 5
43 Sel 4
45 Sel 3
47 Sel 2
49 Sel 1
GND 2
" 4
" 6
" 8
" 10
GND 12
Hd Ld 7 14
GND 16
" 18
" 20
" 22
" 24
" 26
" 28
" 30
" 32
" 34
" 36
GND 38
Hd Ld 6 40
Hd Ld 5 42
Hd Ld 4 44
Hd Ld 3 46
Hd Ld 2 48
Hd Ld 1 50
A3J5
MOLEX 9 Pin
DC POWER Connector
+5V 1
+5V 2
GND 3
+24V 4
+24V 5
GND 6
-5V 7
GND 8
Spare 9
1 +5V
2 +5V
3 GND
4 +24V
5 +24V
6 GND
7 -5V*
8 GND

APPENDIX B

SERVICE REQUESTS

A program running under the Slave CPU can issue one of six SVC's (Supervisor Call); SVC1, SVC2, etc, to effect I/O or a SDOS service function. The SVC is actually an extended I/O instruction. Associated with each SVC is a pointer to a Service Request Block (SRB). The SRB pointers must be placed by the user program in fixed memory locations in the first 16K bytes of common memory.

HEX LOCATION	CONTENTS	DEVICE ADDRESS
40-41	SRB POINTER FOR SVC 1	F7
42-43	SRB POINTER FOR SVC 2	F6
44-45	SRB POINTER FOR SVC 3	F5
46-47	SRB POINTER FOR SVC 4	F4
48-49	SRB POINTER FOR SVC 5	F3
4A-4B	SRB POINTER FOR SVC 6	F2

When the program running under the Slave CPU issues an SVC, the Slave CPU is halted and the Master CPU is interrupted (re-started). Each one of the six SVC's is vectored thru a unique location in Master Memory where the SRB pointer is picked up by SDOS then vectored to a common SDOS SVC processor where the SVC interrupts are serviced. All of the information required to perform the function desired by the caller is contained in the SRB for the SVC which was issued.

SERVICE REQUEST BLOCK (SRB)

A Service Request Block (SRB) consists of eight contiguous bytes located in the first 16K page of Slave memory. The contents of an SRB are as follows:

BYTE	SYMBOL	CONTENT
1	SFC	SVC FUNCTION CODE
2	SCH	CHANNEL NUMBER
3	STAT	STATUS
4	SDAT	SINGLE BYTE DATA
5	BCNT	I/O BYTE COUNT
6	BMAX	I/O BUFFER LENGTH
7-8	BPTR	I/O BUFFER POINTER

A description of each entry in the SRB is given below.

SVC FUNCTION CODE (SFC)

The SVC function code specifies the I/O or service function which is to be performed by the SVC call. A list of functions supported by SDOS is given at the end of this appendix.

CHANNEL NUMBER (SCH)

A logical channel number must be specified for all I/O functions. The channel number must be in the range 0-7. When a logical channel is "ASSIGNED" to a physical device or file, the channel stays "connected" to that device or file until a "CLOSE" function is issued on the channel. The channel number can be used in more than one SRB.

STATUS (STAT)

For all I/O operations, an indication of the result of the I/O request is returned in STAT. If a READ AND PROCEED, or WRITE AND PROCEED function is requested, STAT will first be set to indicate that the I/O operation is in progress. When the I/O operation completes, STAT will be set to indicate the result of the operation.

SINGLE BYTE DATA (SDAT)

This entry is used to return single byte data requested by a non-I/O SVC function. For I/O requests, this is the physical status of the device.

I/O BYTE COUNT (BCNT)

This entry is used to return the actual number of bytes input or output by a READ or WRITE operation. For line oriented ASCII read or writes, this count is the actual number of characters including the end of line character in the line. For BINARY read or writes, the count is the actual number of bytes. It is also used in conjunction with SDAT to return double byte data requested by a non-I/O SVC function.

I/O BUFFER LENGTH (BMAX)

The maximum number of bytes of data to be input to or output from the user's buffer must be given in this byte. Once set by the user, it is not disturbed by SDOS.

I/O BUFFER POINTER (BPTR)

For all I/O operations and for some non-I/O SVC functions, the user must provide the starting address of a buffer in the first 16K page of Slave memory.

Unless otherwise specified in the SVC function description, data transfers to or from the user program are performed through the buffer pointed to by BPTR.

AUTOMATIC BLOCKING

All application program I/O operations on the Signetics TWIN are performed by SDOS running under the Master CPU. Data to be input from floppy disk files are first read into SDOS buffers and then deblocked to the user's buffer as required. Data to be output to floppy disk files are accumulated in SDOS buffers and then output to the file as full sectors. Input/output operations on devices other than floppy disk files are performed directly through the user's buffer.

SVC FUNCTION DESCRIPTIONS

Following are descriptions of the SVC functions available to an application program.

ASSIGN CHANNEL

An application program running under the Slave CPU has eight logical channels through which it can perform I/O.

Any logical channel can be assigned to any physical device attached to the system. A floppy disk file is treated as an independent physical device.

The physical device or floppy disk file to which a logical channel is to be assigned is given as a string of ASCII characters terminated by an EOL (carriage return) character.

To assign a channel, SCH must contain a channel number in the range 0-7 and BPTR must contain the starting address of the device name string.

If a channel is assigned to a floppy disk file which does not exist, the file will be automatically created and the STAT entry in the SRB will be set to a 1 to indicate that it is a new file.

READ/WRITE ASCII

A line is defined as a string of ASCII characters terminated by an End-Of-Line character, which is a normal ASCII carriage return. An application program running under the Slave CPU can input or output a line through a channel which has been previously "ASSIGNED" to a floppy disk file or other I/O device. The required settings for the SRB are minimal.

To read a line, BPTR must contain the starting address of a buffer into which the line is to be input. BMAX must contain the maximum number of characters the user wants placed in the buffer in the event an EOL character is not in the data stream. The user's buffer must be able to contain BMAX+1 bytes of data because, if an EOL character is not found in the data stream and BMAX bytes have been placed in the buffer, an EOL character will be placed in the buffer at the BMAX+1 position. This procedure assures the user that an ASCII input line will always be terminated by an EOL.

To write a line, BPTR must contain the starting address of the line to be output. BMAX must contain the maximum number of ASCII characters to be output in the event the line is not terminated by an EOL character.

After the I/O function is completed, BCNT will contain the actual number of ASCII characters including the EOL which were input or output to or from the user's buffer.

The actual number of characters input or output may be less than the maximum specified by BMAX as a result of encountering the normal EOL character, an end-of-file on a READ before finding a EOL, or an end-of-device on a WRITE. All but the first return a non-zero status.

The maximum length of a line supported by SDOS is 255 characters plus the EOL character for a total of 256. Thus, BMAX must be greater than or equal to 1 and less than or equal to 255.

The four functions, READ AND WAIT, READ AND PROCEED, WRITE AND WAIT, WRITE AND PROCEED, provide control over the action of the operating system when reading from or writing into the user's buffer. In the "WAIT" options, the operating system outputs a H '7F' to the status byte to indicate that the read or write is complete. The applications program must test the status byte to determine completion before proceeding with another read or write operation. In the "PROCEED" options, the operating system returns to the applications program as soon as a read or write is complete. In the "WAIT" options, for example, the applications program can be written to allow two buffers to be used alternately. When the first buffer is full, a read can commence and a second buffer can be filled. When the status byte test shows that the first buffer has been read, the second buffer can then be read while the first is being filled.

READ/WRITE BINARY

An application program running under the Slave CPU can input or output a block of binary bytes through a channel which has been previously "ASSIGNED" to a floppy disk file or other I/O device.

To read a block of binary bytes, BPTR must contain the starting address of a buffer into which the data is to be input. BMAX must contain the number of bytes to be input to the buffer.

To write a block of binary bytes, BPTR must contain the starting address of the buffer from which the data is to be output. BMAX must contain the number of bytes to be output from the buffer.

After the I/O function is completed, BCNT will contain the actual number of bytes which were input or output to or from the user's buffer.

Binary reads and writes are performed strictly under count control. A user may input or output up to 256 bytes of data. In the case of read or write binary, a BMAX value of 0 is taken to mean 256.

CLOSE FILE

The close function "disconnects" the given channel from the floppy disk file or other I/O device to which it was "ASSIGNED".

If the channel was "ASSIGNED" to a floppy disk write file, the SDOS buffer used for the file will be output and a logical end-of-file will be recorded on the file before it is "disconnected". Subsequently, when the file is read, the End-Of-File condition will be sensed and indicated in the status byte of the SRB.

For physical devices other than the floppy disk, an appropriate clearing action will be taken.

REWIND FILE

The REWIND function applies only to floppy disk files. It has the effect of positioning a file at its beginning.

If a device other than a floppy disk file has been "ASSIGNED" to the channel, the function will be treated as a NOP.

To rewind a file, SCH in the SRB for the SVC request must contain the channel number to which the floppy disk file has been "ASSIGNED".

When a floppy disk file is "REWOUND" it is treated the same as if it had just been "ASSIGNED". If the first I/O operation for the rewound file is a READ, data will be input from the file in the normal manner. If the first I/O operation for the rewound file is a WRITE, the sectors previously allocated for the file will be deallocated and the file will be treated as if it were a new file.

DELETE FILE

The DELETE function causes the floppy disk file assigned to the given channel to be deleted from the directory of the diskette and, as a consequence, causes the channel to be "disconnected" from the file.

If a device instead of a floppy disk file has been "ASSIGNED" to the channel, the function will be treated the same as "CLOSE".

RENAME FILE

To RENAME the floppy disk which has been "ASSIGNED" to the given channel, BPTR must contain the starting address of the name, given as an ASCII line, to which the file is to be renamed.

A file which is to be renamed must not be in the process of being read or written, i.e., the file must have just been "ASSIGNED" or "REWOUND".

If a device other than a floppy disk file has been "ASSIGNED" to the channel, the function will be treated as a NOP.

GET PARAMETER

If an application program running under the Slave CPU was invoked as a part of a command file from the system console, there may have been parameters in the command line. This SVC gets a particular parameter for the application program.

If parameters are present, they will be stored in the SDOS procedure parameter buffer in Master memory. The parameters are identified by number according to the order in which they appeared in the command line and exist as strings of ASCII characters terminated by an EOL character.

The desired parameter is requested as a number of SDAT. The parameter is returned to the user as an ASCII line starting at the location contained in BPTR.

When a command file command line is entered, parameters are delimited by a space, comma, or EOL character. A comma or space delimiter will be replaced with an EOL character before the parameter is stored in the parameter buffer. A parameter may be omitted from the ordered sequence by two consecutive commas. If a parameter has been omitted, the first character in the user's buffer will be an EOL character.

If the given parameter number is greater than the number of parameters included in the command line, the first byte of the user's buffer will be a -1, and the SRB status will indicate that the given parameter number was greater than the number of parameters included in the command line.

LOAD OVERLAY

Overlays are stored on disk as load modules complete with memory beginning, end, and start addresses. The program loads an overlay by setting the SRB and issuing the SVC. The file name of the overlay is given as an ASCII string terminated by an EOL. The BPTR field in the SRB must point to this string. The header information in the load module determines where the overlay is to be loaded in memory. The result of the load operation is returned in STAT. The overlay is not started, and control remains with the requesting program.

EXECUTE OVERLAY

This SVC function is called and performed in the same way as the "LOAD OVERLAY" function with the exception that the overlay is executed after it is loaded at the starting address given in the Header of the load module. This SVC provides the capability of chaining separate programs, as distinct from overlays.

SUSPEND EXECUTION

This SVC function will cause the requesting program to be suspended at the place the SVC is issued. The action is similar to an I/O and wait. The program can be started executing again by an operator command issued from the system console.

EXIT

This SVC function will cause the program running under the Slave CPU to be terminated. The channels previously assigned by or for the program will not be closed.

ABORT

This SVC function will cause the program running under the Slave CPU to be terminated. The channels previously assigned by or for the program, if not already closed, will be closed.

GET TIME

The cumulative time in milliseconds since system start-up is returned in SDAT and BCNT.

GET OVERLAY ADDRESS

The memory bounds of the last overlay loaded into slave memory and the execution address of the overlay are stored in six consecutive bytes starting at the address given in BPTR. The first two bytes will contain the low load address, the next two bytes will contain the high load address, and the last two bytes will contain the execution address.

GET DEVICE STATUS

The status of the device assigned to the given logical channel (SCH), as obtained from the physical device, is returned in SDAT. A default of zero will be returned if there is no physical status available.

GET DEVICE TYPE

The identification number of the device assigned to the channel number in SCH is returned in SDAT and the device type is returned in BCNT.

GET LAST CONSOLE INPUT CHAR

The last character input from the system console is returned in SDAT. If sensed in a loop while performing extensive calculations or I/O, it provides the user program with a way to respond to a request for attention or other action by the operator.

STATUS AND FUNCTION CODES

Tables B-1 through B-3 list service request status and function codes.

TABLE B-1

SVC FUNCTION CODES

HEX CODE	FUNCTION
01	READ ASCII and WAIT
81	READ ASCII and PROCEED
41	READ BINARY and WAIT
C1	READ BINARY and PROCEED
02	WRITE ASCII and WAIT
82	WRITE ASCII and PROCEED
42	WRITE BINARY and WAIT
C2	WRITE BINARY and PROCEED
03	CLOSE device or file on channel
04	REWIND file on channel
05	DELETE file on channel
06	RENAME file on channel
10	ASSIGN channel to device or file
11	GET TIME (milliseconds)
12	GET OVERLAY ADDRESSES
13	GET PARAMETER
14	GET DEVICE TYPE
15	GET DEVICE STATUS
16	GET LAST CONSOLE INPUT CHAR.
17	LOAD OVERLAY
18	EXECUTE OVERLAY
19	SUSPEND EXECUTION
1A	EXIT
1F	ABORT

TABLE B-2

DEVICE NAME	DEVICE I.D.	DEVICE TYPE
CONI (Console Input)	1	1
CONO (Console Output)	2	2
LPT1 (Line Printer)	3	2
HSPT (H.S. P/T READER)	4	1
TTYR (TTY P/T READER)	6	1
DISK FILE	-1	43

DEVICE TYPES

1	ASCII READ
41	BINARY READ
2	ASCII WRITE
42	BINARY WRITE
3	ASCII READ/WRITE
43	BINARY READ/WRITE

The device types specified for the SDOS I/O devices represent the way in which the SDOS commands treat the devices in normal usage. A user application program can read from any input device in either ASCII or Binary mode and can write to any output device in either ASCII or Binary mode.

The CONO, CONI, and the floppy disk subsystem are sharable devices which can be assigned to more than one channel. The LPT1, HSPT, and TTYR are non-sharable devices and can only be assigned to one channel at any given time.

A user application program can have a maximum of seven channels assigned to floppy disk files.

TABLE B-3

SRB STATUS CODES

00	-	FUNCTION COMPLETE / NO ERROR
01	-	CHANNEL ASSIGNED TO NEW FILE
02	-	ILLEGAL CHANNEL NUMBER
03	-	CHANNEL NOT ASSIGNED
04	-	CHANNEL BUSY
05	-	ILLEGAL FUNCTION CODE
06	-	NO EOL ON ASCII READ
07	-	NO EOL ON ASCII WRITE
08	-	ILLEGAL DRIVE NUMBER
09	-	FILE IN USE
0A	-	DEVICE NOT OPERATIONAL
0B	-	DEVICE NOT AVAILABLE
0C	-	DEVICE NOT READY
0D	-	DEVICE IN USE
0E	-	DIRECTORY READ ERROR
0F	-	DIRECTORY WRITE ERROR
10	-	DIRECTORY FULL
11	-	DEVICE READ ERROR
12	-	DEVICE WRITE ERROR
13	-	CODE NOT ASSIGNED
14	-	CODE NOT ASSIGNED
15	-	FILE NAME IN USE
16	-	ILLEGAL FILE NAME
17	-	FILE IN R/W PROGRESS
18	-	CHANNEL ALREADY ASSIGNED
19	-	INCORRECT DISKETTE
7F	-	I/O IN PROGRESS
FF	-	END OF FILE OR END OF DEVICE

APPENDIX C

TELETYPE INSTALLATION

The standard ASR-33 teletypewriter must be modified for use with the TWIN system. The modification changes the loop current to 20 mA and adds a tape reader control circuit. Use the procedure following:

Internal Modifications

1. Change the current source resistor value to 1450 ohms. This is done by moving a single wire (see Figure C-5).

2. Create a full duplex hookup internally by moving two wires on a terminal strip (see Figure C-4 and C-6).

3. Change the receiver current level to 20 mA by moving a single wire (see Figure C-4 and C-6).

4. Install a tape reader control circuit consisting of a relay, a resistor, a diode, and a contact protection network. This requires mounting of a small board with the relay circuit on it. It may be mounted in the teletypewriter using the two tapped holes in the base plate (see Figure C-1). The relay circuit may be added without alteration of the existing circuit (see Figures C-2, C-3, and C-4). Wire "A", shown in Figure C-6, to be connected to the brown wire in Figure C-2, may be spliced into the brown wire near its connector. The "line" and "local" wires must then be connected to the mode switch (see Figures C-3 and C-6).

External Connections

1. Create a two-wire receive loop by connecting two wires between the system and the teletypewriter, as shown in Figure C-6.

2. Create a two-wire send loop by connecting a second set of wires between the system and the teletypewriter, as shown in Figure C-6.

3. Create a two-wire tape reader loop by connecting the system to the teletypewriter as shown in Figure C-6.

The teletype is connected to the TWIN PDC console connector J108 via a 25-pin EIA connector. Cable wiring for connecting a current loop TTY to the system are shown in Figure C-7. A TTY equipped with an RS-232 interface option connects to the system using the standard cable supplied with the TWIN system. These connections are shown in Figure C-8 for reference.

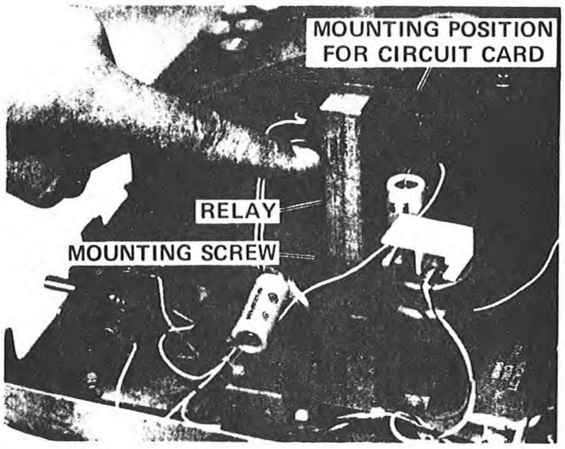


FIGURE C-1

Relay Circuit (Alternate)

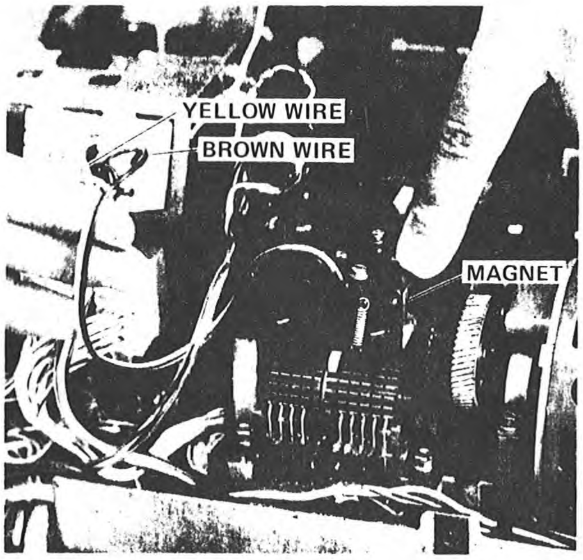


FIGURE C-2

Distributor Trip Magnet

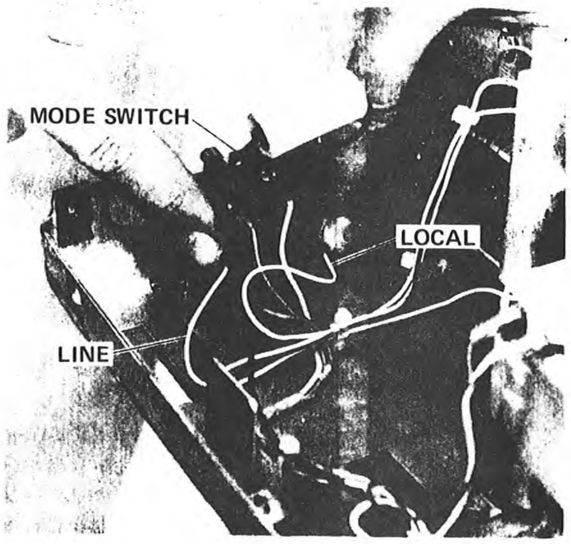


FIGURE C-3

Mode Switch

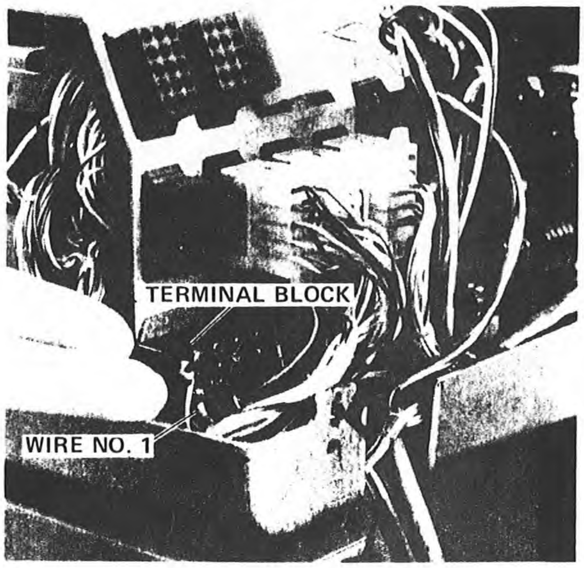


FIGURE C-4

Terminal Block

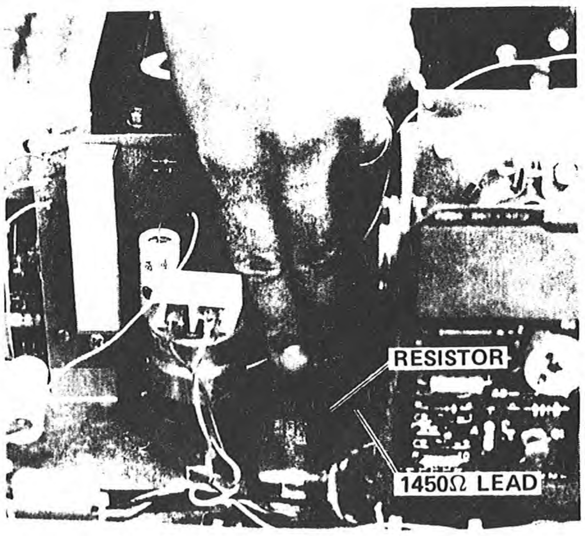


FIGURE C-5

Current Source Resistor

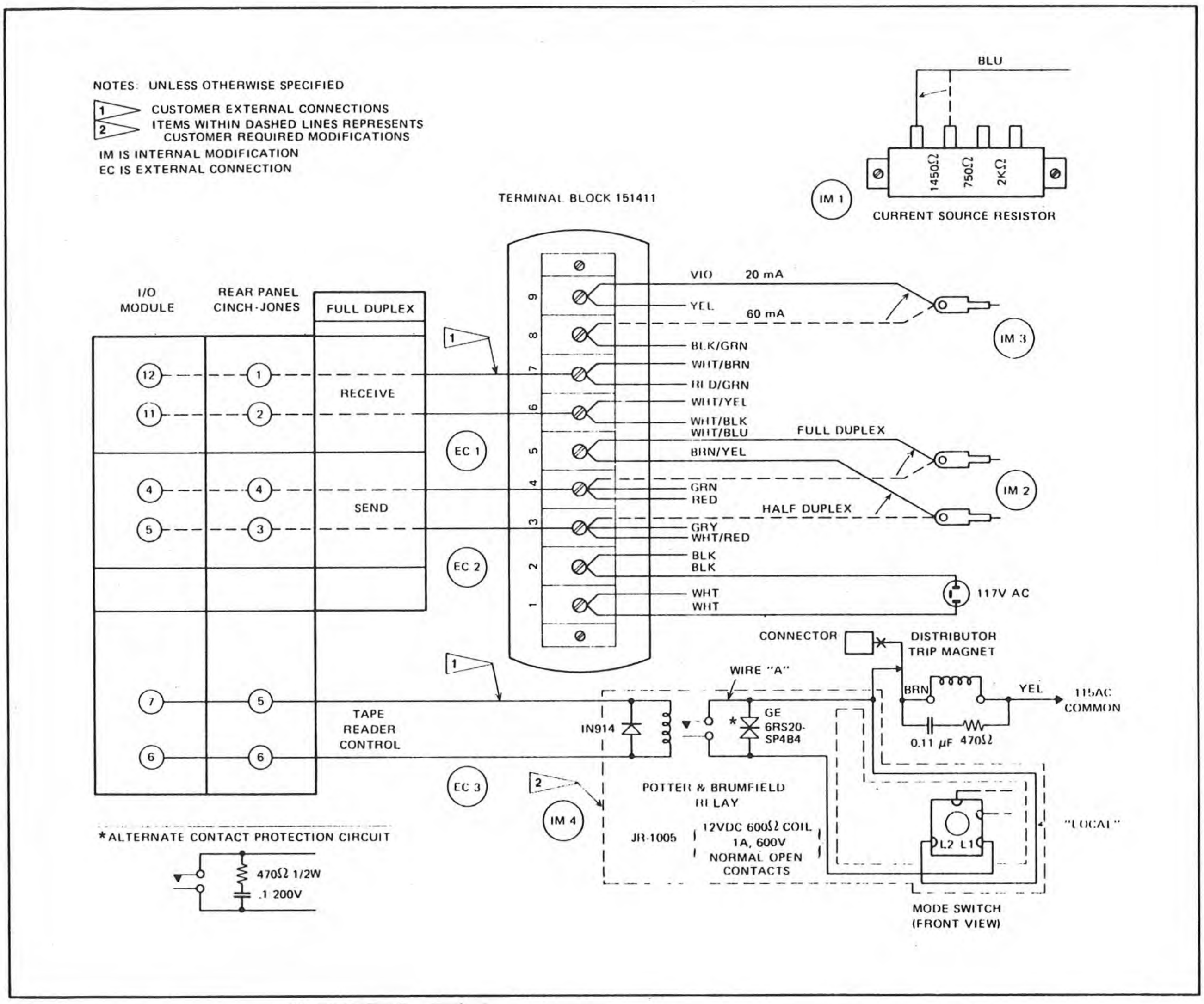


FIGURE C-6

TTY Modification

20 ma Loop

PIN		Description
15 ~~2~~	RCV DATA	Receive Data
16 ~~3~~	XMIT DATA	Transmitted Data
13	TTY XMIT+	TTY Current Loop Output
14	TTY XMIT-	TTY Current Loop Output
17	TTY RCV+	TTY Current Loop Input
18	TTY RCV-	TTY Current Loop Input

FIGURE C-7

CURRENT LOOP TTY TO TWIN CABLE

RS-232-C

PIN		Description
1	GND	Protective Ground
2	EIA REV DATA	Serial Input
3	EIA XMIT DATA	Serial Output
4	REQ TO SND	Request to Send Input
5	CLR TO SND	Clear to Send Output
6	DATA SET RDY	Data Set Ready Output
7	EIA SIG GND	Signal Ground
8	SIG DET	Carrier Detected Output
20	DATA TERM RDY	Data Terminal Ready

FIGURE C-8

RS-232 TERMINAL TO TWIN CABLING

APPENDIX D

ADDING A DEVICE DRIVER TO SDOS

This description assumes the reader is familiar with the two separate handler and interrupt service portions of a driver and serves to show how to incorporate them (merge them into) the SDOS load module.

The merging scheme is very simple. The user assembles his logic as a separate object module (in hexadecimal format), and then performs the following steps using SDOS functions:

1. LOAD SDOS/0
   Load the SDOS load module into Slave memory.

2. RHEX FILENAME (of the object module)
   Merge the new driver logic into Slave memory with SDOS.

3. MODULE SDOS/1 80 3FFF 100 (COMMENT)
   Make a new SDOS module on a different diskette (advisable until the new driver is debugged).

A description of how and where to merge the driver logic and its related device tables is given in the example below.

The High Speed Paper Tape Reader driver (device Name HSPT) was added to SDOS using the procedure described below. At the end of this description is a copy of the source listing used.

The HSPT listing shows what is needed in the way of linkages and tables for the driver; briefly the needs are these:

1. Linkages to SDOS routines SAVR, RESR, IOC3, SDCB, and DISP (DISPATCHER). The user should be familiar with the functions of these routines (briefly annotated on the listing);

2. FCB (File Control Block) linkage address to the first of the 22 SDOS FCB's. There is no "dedicated" FCB for a particular handler, but the driver is provided with an "Active FCB Index" in the driver-dedicated DCB described below (entry DFCB) when I/O is initiated. Upon entry to the "handler" the Active FCB Index is in register R1.

3. Shown next in the listing is the creation of linkages to the new driver's DCB (Device Control Block). Note that no data need be merged into the DCB, but linkages must be provided.

Note that for the HSPT, the device index chosen is PTDV (see DEVICE CONTROL BLOCK INDICES). A new user must choose an available index from the range USR1=7 to USR8=15. It should be clear by the listing, how to set up a new DCB.

4. Now that a device index has been chosen, the corresponding Device Definition Table (DDT) can be set up to define the new device, by name and characteristics, in the SDOS data base. The user must use the DDT entry point linkage addresses shown in the listing (DHAN, DTST, etc.) and use the new device index for an offset in the "ORG's" in place of the PTDV index (HSPT).

   The data which must be provided in the entries for this table should reflect the new device. Note that the DHAN entry is an "unconditional absolute branch" to the device "handler" portion of the new driver.

5. Add the transfer vector for the "interrupt service" portion of the driver. An appropriate slot must be chosen among those in the available hardware interrupt locations which start at location H'1000'. Those slots available are: H'1006', H'1018, H'1024', H'1057, H'105A', and H'105D'.

6. Now all that is left is to add the driver logic itself to SDOS. The origin chosen must be within the available memory as shown here:

   a. 4-drive system: H'3A00' to H'3DFF'
   b. 8-drive system: H'3AF2' to H'3DFF'

```
TWIN ASSEMBLER VER 1.0      HSPT VER R.2 7/30/76          PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0002                 *
0003                 *
0004                 *
0005                 *              ****************************
0006                 *              *                          *
0007                 *              *                          *
0008                 *              *  HIGH SPEED PAPER TAPE   *
0009                 *              *      READER DRIVER       *
0010                 *              *                          *
0011                 *              *                          *
0012                 *              ****************************
0013                 *
0014                 *
0015                 *
0016                 *
0017                 *
0018                 *   REGISTER EQUATES
0019                 *
0020 0000            R0     EQU         0         REGISTER 0
0021 0001            R1     EQU         1         REGISTER 1
0022 0002            R2     EQU         2         REGISTER 2
0023 0003            R3     EQU         3         REGISTER 3
0024                 *
0025                 *   CONDITION CODES
0026                 *
0027 0001            P      EQU         1         POSITIVE RESULT
0028 0000            Z      EQU         0         ZERO RESULT
0029 0002            N      EQU         2         NEGATIVE RESULT
0030 0002            LT     EQU         2         LESS THAN
0031 0000            EQ     EQU         0         EQUAL TO
0032 0001            GT     EQU         1         GREATER THAN
0033 0003            UN     EQU         3         UNCONDITIONAL
0034                 *
0035                 *   PSW LOWER EQUATES
0036                 *
0037 0080            CCI    EQU         H'80'
0038 0040            CC0    EQU         H'40'
0039 0010            RS     EQU         H'10'
0040 0008            WC     EQU         H'08'
0041 0004            OVF    EQU         H'04'
0042 0002            COM    EQU         H'02'
0043 0001            C      EQU         H'01'
0044                 *
0045                 *   PSW UPPER EQUATES
0046                 *
0047 0080            SENS   EQU         H'80'
0048 0040            FLAG   EQU         H'40'
0049 0020            II     EQU         H'20'
0050 0004            SP2    EQU         H'04'
0051 0002            SP1    EQU         H'02'
0052 0001            SP0    EQU         H'01'
0053                 *
0054                 *   HIGH SPEED PAPER TAPE READER INPUT AND CONTROL PORTS
0055                 *
0056 00D0            PCPT   EQU         H'D0'     HSPT CONTROL PORT
```

```
LINE ADDR  OBJECT  E SOURCE

0057 00D1               PDPT   EQU         H'D1'     HSPT DATA PORT
0058 00EE               BSPT   EQU         H'EE'     COMMON MEMORY BANK SELECT
0059                    *
0060                    *
0061                    *
0062                    *
0063                    *
0064                    **********************************
0065                    *                               *
0066                    *  GENERAL CONTROL CHARACTERS   *
0067                    *                               *
0068                    **********************************
0069                    *
0070 000D               EOL    EQU         H'0D'     END OF LINE
0071 000D               CR     EQU         H'0D'     CARRIAGE RETURN
0072 000A               LF     EQU         H'0A'     LINE FEED
0073 001A               CTLZ   EQU         H'1A'     CONTROL Z
0074 007F               RUB    EQU         H'7F'     RUBOUT
0075 001B               ESC    EQU         H'1B'     ESCAPE
0076 0020               SPAC   EQU         H'20'     SPACE
0077 002F               SLSH   EQU         A'/'
0078 0000               NULL   EQU         0         NULL
0079 0011               XON    EQU         H'11'     XON FOR TTY READER
0080 0013               XOF    EQU         H'13'     XOFF FOR TTY READER
0081                    *
0082                    *  JCB STATES
0083                    *
0084 0000               JS0    EQU         0         JOB IDLE (NO JOB)
0085 0001               JS1    EQU         1         JOB LOADED
0086 0002               JS2    EQU         2         JOB READY TO START
0087 0003               JS3    EQU         3         JOB EXECUTING
0088 0004               JS4    EQU         4         JOB IN I/O WAIT
0089 0005               JS5    EQU         5         JOB I/O COMPLETE
0090 0006               JS6    EQU         6         JOB SUSPENDED
0091 0007               JS7    EQU         7         JOB BEING ABORTED
0092 0008               JS8    EQU         8         JOB SELF PAUSED
0093                    *
0094                    *  DCB STATES
0095                    *
0096 0000               DRDY   EQU         0         DEVICE READY
0097 0001               DBSY   EQU         1         DEVICE BUSY
0098 0002               DVDN   EQU         2         DEVICE DOWN
0099                    *
0100                    *  DSB STATES
0101                    *
0102 0000               DSBU   EQU         0         DRIVE STATE UNKNOWN
0103 0001               DSBS   EQU         1         DRIVE DSB SET
0104 0002               DSBD   EQU         2         DRIVE DOWN
0105                    *
0106                    *  FCB STATES
0107                    *
0108 0000               FREE   EQU         0         FREE
0109 0001               FASS   EQU         1         ASSIGNED
0110 0002               FOPN   EQU         2         OPEN
0111 0003               FBSY   EQU         3         BUSY
0112 0004               FRDY   EQU         4         READY
```

```
LINE ADDR  OBJECT  E SOURCE

     0005            FEOF   EQU       5        EOF/EOD
     0005            FEOD   EQU       FEOF
     0006            FABT   EQU       6        ABORT
                     *
                     *
                     *
                     *
                     *
                     *
                     *
                     *
                     ************************
                     *                      *
                     *   SRB STATUS CODES   *
                     *                      *
                     ************************
                     *
     0000            SE00   EQU       0        FUNCTION COMPLETE / NO ERROR
     0001            SE01   EQU       1        NEW FILE
     0002            SE02   EQU       2        ILLEGAL CHANNEL NUMBER
     0003            SE03   EQU       3        CHANNEL NOT ASSIGNED
     0004            SE04   EQU       4        CHANNEL BUSY
     0005            SE05   EQU       5        ILLEGAL FUNCTION CODE
     0006            SE06   EQU       6        SHORT READ
     0007            SE07   EQU       7        SHORT WRITE
     0008            SE08   EQU       8        ILLEGAL DRIVE NUMBER
     0009            SE09   EQU       9        FILE IN USE
     000A            SE10   EQU       10       DEVICE NOT OPERATIONAL
     000B            SE11   EQU       11       DEVICE NOT AVAILABLE
     000C            SE12   EQU       12       DEVICE NOT READY
     000D            SE13   EQU       13       DEVICE IN USE
     000E            SE14   EQU       14       DIRECTORY READ ERROR
     000F            SE15   EQU       15       DIRECTORY WRITE ERROR
     0010            SE16   EQU       16       DIRECTORY FULL
     0011            SE17   EQU       17       DEVICE READ ERROR
     0012            SE18   EQU       18       DEVICE WRITE ERROR
     FFFF            SE19   EQU       -1       NOT ASSIGNED
     FFFF            SE20   EQU       -1       NOT ASSIGNED
     0015            SE21   EQU       21       FILE NAME IN USE
     0016            SE22   EQU       22       ILLEGAL FILE NAME
     0017            SE23   EQU       23       FILE IN R/W PROGRESS
     0018            SE24   EQU       24       CHANNEL ALREADY ASSIGNED
     0019            SE25   EQU       25       INCORRECT DISKETTE
     007F            SE7F   EQU       H'7F'    I/O IN PROGRESS
     00FF            SEFF   EQU       H'FF'    END OF FILE / END OF DEVICE
                     *
                     ****************************
                     *                          *
                     *   SVC FUNCTION OP CODES  *
                     *                          *
                     ****************************
                     *
     0001            RDOP   EQU       H'01'    READ
     0002            WTOP   EQU       H'02'    WRITE
     0003            CLOP   EQU       H'03'    CLOSE
     0004            RWOP   EQU       H'04'    REWIND
```

```
LINE ADDR  OBJECT  E SOURCE

0169 0005             DEOP   EQU      H'05'    DELETE
0170 0006             RNOP   EQU      H'06'    RENAME
0171 0007             DIOP   EQU      H'07'    DIRECT I/O
0172 0010             ASOP   EQU      H'10'    ASSIGN
0173 0061             RBOP   EQU      H'61'    READ BLOCK
0174 0062             WBOP   EQU      H'62'    WRITE BLOCK
0175                  *
0176                  *
0177                  *
0178                  *
0179                  *
0180                  *
0181                  *
0182                  *
0183                  *
0184                  ************************************
0185                  *                                  *
0186                  *   DEVICE CONTROL BLOCK INDICES   *
0187                  *                                  *
0188                  ************************************
0189                  *
0190 0000             CIDV   EQU      0        CONSOLE INPUT
0191 0001             CODV   EQU      1        CONSOLE OUTPUT
0192 0002             LPDV   EQU      2        LINE PRINTER
0193 0003             PTDV   EQU      3        PAPER TAPE READER
0194 0004             ERDV   EQU      4        ERROR PSUEDO DEVICE
0195 0005             TTYR   EQU      5        TTY READER
0196 0006             LPT2   EQU      6        SECOND LINE PRINTER
0197 0007             USR1   EQU      7        USER 1 INDEX
0198 0008             USR2   EQU      8        USER 2 INDEX
0199 0009             USR3   EQU      9        USER 3 INDEX
0200 000A             USR4   EQU      10       USER 4 INDEX
0201 000B             USR5   EQU      11       USER 5 INDEX
0202 000C             USR6   EQU      12       USER 6 INDEX
0203 000D             USR7   EQU      13       USER 7 INDEX
0204 000E             USR8   EQU      14       USER 8 INDEX
0205 000F             FDDV   EQU      15       FLOPPY DISK
0206                  *
0207                  *   (NOTE: USER INDICIES ARE 7-14)
0208                  *
0209                  *******************************************
0210                  *                                         *
0211                  *   LINKS TO SDOS ROUTINES AND DATA BASE  *
0212                  *                                         *
0213                  *******************************************
0214                  *                  *
0215                  *   SDOS ROUTINES  *
0216                  *                  *
0217                  ********************
0218                  *
0219 2003             SAVR   EQU      H'2003'  SAVE REGISTERS
0220 2006             RESR   EQU      H'2006'  RESTORE REGISTERS
0221 136C             IOC3   EQU      H'136C'  SET CHANNEL I/O COMPLETE
0222 136A             SDCB   EQU      H'136A'  SET-UP DCB FROM THE FCB
0223 200F             DISP   EQU      H'200F'  DISPATCHER
0224                  *
```

```
LINE ADDR  OBJECT  E SOURCE

0225                 *****************************
0226                 *                           *
0227                 *   FCB ENTRIES (1ST FCB)   *
0228                 *                           *
0229                 *****************************
0230                 *
0231 2150            FCS    EQU        H'2150'   FCB STATE
0232 2166            FJCB   EQU        FCS+22    JCB INDEX
0233 217C            FBNK   EQU        FJCB+22   BANK
0234 2192            FCH    EQU        FBNK+22   CHANNEL NUMBER
0235 21A8            FCOD   EQU        FCH+22    SVC FUNCTION CODE
0236 21BE            FPDS   EQU        FCOD+22   PHYSICAL DEVICE STATUS
0237 21D4            FSTA   EQU        FPDS+22   I/O COMPLETE STATUS
0238 21EA            FCNT   EQU        FSTA+22   I/O BUFFER BYTE COUNT
0239                 *
0240                 *
0241                 *
0242                 *
0243                 *
0244                 ***************************
0245                 *                         *
0246                 *   DEFINE THE HSPT DCB   *
0247                 *                         *
0248                 ***************************
0249                 *
0250 2346            DCS    EQU        H'2346'   1ST DCB ADDRESS (DEVICE INDEX 0)
0251 2349            PTCS   EQU        DCS+PTDV  DEVICE STATE FOR HSPT
0252 2359            PTTA   EQU        PTCS+16   DEVICE STATUS
0253 2369            PTDX   EQU        PTTA+16   I/O BUFFER INDEX
0254 2379            PTNT   EQU        PTDX+16   I/O BUFFER COUNT
0255 2389            PTCO   EQU        PTNT+16   I/O BUFFER ECHO COUNT
0256 2399            PTCB   EQU        PTCO+16   ACTIVE FCB INDEX
0257 23A9            PTOD   EQU        PTCB+16   SVC FUNCTION CODE
0258 23B9            PTNK   EQU        PTOD+16   BANK SWITCH
0259 23CC            PTOB   EQU        PTNK+16+PTDV  I/O BUFFER POINTER (DOUBLE-BYTE)
0260                 *
0261                 ***************************
0262                 *                         *
0263                 *   DEFINE THE HSPT DDT   *
0264                 *                         *
0265                 ***************************
0266                 *
0267                 *
0268                 *
0269 2436            DHAN   EQU        H'2436'   1ST 'DHAN' ENTRY (THREE-BYTE)
0270 0000                   ORG        DHAN+PTDV+PTDV+PTDV
0271 243F 1F3EA2            BCTA,UN    HPTH      ADD XFER VECTOR FOR HSPT HANDLER
0272                 *
0273                 *
0274                 *
0275 2416            DTST   EQU        H'2416'   1ST 'DTST' ENTRY
0276 2442                   ORG        DTST+PTDV HSPT DEVICE AVAILABILITY
0277 2419 00                DATA       H'0'      SET HSPT UP AND NON-SHAREABLE
0278                 *
0279                 *
0280                 *
```

```
LINE ADDR  OBJECT  E SOURCE

2426                DTTP   EQU      H'2426'   1ST 'DTTP' ENTRY
241A                       ORG      DTTP+PTDV HSPT DEVICE I/O TYPE
2429 01                    DATA     1         SET 'READ ONLY' IN TABLE
                    *
                    *
                    *
2406                DTID   EQU      H'2406'   1ST 'DTID' ENTRY
242A                       ORG      DTID+PTDV HSPT DEVICE ID
2409 04                    DATA     PTDV+1    SET DEVICE ID
                    *
                    *
                    *
2466                DTNM   EQU      H'2466'   1ST 'DTNM' ENTRY (FOUR-BYTE)
240A                       ORG      DTNM+PTDV+PTDV+PTDV+PTDV
2472 48535054              DATA     A'HSPT'   ADD DEVICE NAME
                    *
                    *
                    *
                    *
                    *
                    *
                    *
                    *
                    ************************************
                    *                                  *
                    *   ADD THE HSPT INTERRUPT VECTOR  *
                    *                                  *
                    ************************************
                    *
2476                       ORG      H'1006'
1006 1F3E00                BCTA,UN  HISR      ADD VECTOR TO VECTOR AREA
                    *                         IN AN AVAILABLE SLOT
                    *
                    ***********************************
                    *                                 *
                    *   HIGH SPEED PAPER TAPE READER  *
                    *     INTERRUPT SERVICE ROUTINE   *
                    *                                 *
                    ***********************************
                    *
                    *
1009                       ORG      H'3E00'   ORG INTO AVAILABLE MEMORY AREA
                    *
                    *
3E00 3F2003         HISR   BSTA,UN  SAVR      SAVE MACHINE STATE
3E03 0D0399                LODA,R1  PTCB      GET ACTIVE FCB
                    *
3E06 0C0349                LODA,R0  PTCS      GET CURRENT DEVICE STATE
3E09 E401                  COMI,R0  DBSY      CHECK FOR BUSY
3E0B 9C3E7D                BCFA,EQ  HIS2      NO - DISMISS INTERRUPT
                    *
3E0E 070A                  LODI,R3  SE10      ASSUME DEVICE NOT OPERATIONAL
3E10 54D0           HIS8   REDE,R0  PCPT      GET STATUS
3E12 CC0359                STRA,R0  PTTA      SAVE
3E15 247F                  EORI,R0  H'7F'
3E17 9C3E95                BCFA,Z   HIS5      BRANCH FOR DEVICE DOWN
```

```
LINE ADDR  OBJECT  E SOURCE

0337                 *
0338 3E1A 54D1              REDE,R0    PDPT      READ DATA
0339                 *
0340 3E1C 0F03A9            LODA,R3    PTOD      GET OP CODE
0341 3E1F 4740              ANDI,R3    H'40'     TEST FOR BINARY
0342 3E21 9822              BCFR,Z     HIS6      YES
0343                 *
0344 3E23 447F              ANDI,R0    H'7F'     STRIP PARITY
0345 3E25 1806              BCTR,Z     HIS9      SKIP NULLS
0346                 *
0347 3E27 E413              COMI,R0    XOF       SKIP XOFFS
0348 3E29 1802              BCTR,EQ    HIS9
0349                 *
0350 3E2B E40A              COMI,R0    LF        SKIP LINE FEEDS
0351 3E2D 1C3E86     HIS9   BCTA,EQ    HIS7
0352                 *
0353 3E30 E47F              COMI,R0    RUB       TEST FOR RUBOUT
0354 3E32 9809              BCFR,EQ    HISA      NO
0355                 *
0356 3E34 0E0369            LODA,R2    PTDX      GET INDEX
0357 3E37 1819              BCTR,Z     HISB      BOUNDARY
0358                 *
0359 3E39 A601              SUBI,R2    1         BACK UP A CHAR
0360 3E3B 1B15              BCTR,UN    HISB      BOUNDARY
0361                 *
0362                 *
0363 3E3D E41A       HISA   COMI,R0    CTLZ      CHECK FOR EOF
0364 3E3F 9804              BCFR,EQ    HIS6      NO
0365 3E41 040D              LODI,R0    EOL
0366 3E43 0705              LODI,R3    FEOF
0367                 *
0368 3E45 0E03B9     HIS6   LODA,R2    PTNK      SWITCH IN PROPER BANK
0369 3E48 D6EE              WRTE,R2    BSPT
0370 3E4A 0E0369            LODA,R2    PTDX      GET I/O BUFFER INDEX
0371 3E4D CEE3CC            STRA,R0    *PTOB,R2  STORE DATA
0372                 *
0373 3E50 8601              ADDI,R2    1         INCREMENT BUFFER INDEX
0374 3E52 CE0369     HISB   STRA,R2    PTDX
0375 3E55 F740              TMI,R3     H'40'     CHECK FOR BINARY
0376 3E57 1828              BCTR,Z     HIS3      YES - CHECK COUNT
0377                 *
0378 3E59 E40D              COMI,R0    EOL       NO - ASCII
0379 3E5B 9824              BCFR,EQ    HIS3      GO CHECK COUNT
0380                 *
0381 3E5D 03                LODZ       R3        CHECK FOR EOF
0382 3E5E 1835              BCTR,Z     HIS5      NO
0383 3E60 0700              LODI,R3    0         SET STAT O.K.
0384                 *
0385 3E62 CD6150     HIS1   STRA,R0    FCS,R1    STORE CURRENT STATE
0386 3E65 03                LODZ       R3        STORE STATUS IN FCB
0387 3E66 CD61D4            STRA,R0    FSTA,R1
0388                 *
0389 3E69 0C0369            LODA,R0    PTDX      STORE BYTE COUNT IN FCB
0390 3E6C CD61EA            STRA,R0    FCNT,R1
0391 3E6F 0C0359            LODA,R0    PTTA      STORE PHYSICAL STATUS READ
0392 3E72 CD61BE            STRA,R0    FPDS,R1
```

```
LINE ADDR  OBJECT  E SOURCE

                     *
3E75 0400                  LODI,R0     DRDY       SET DEVICE READY
3E77 CC0349                STRA,R0     PTCS
3E7A 3F936C                BSTA,UN     *IOC3      SET I/O COMPLETE
                     *
3E7D 3F2006          HIS2  BSTA,UN     RESR
3E80 37                    RETE,UN
                     *
3E81 EE0379          HIS3  COMA,R2     PTNT       CHECK FOR MAX COUNT
3E84 1804                  BCTR,EQ     HIS4       YES - PROCESS
3E86 3B11            HIS7  BSTR,UN     HSIO       START INPUT
3E88 1B73                  BCTR,UN     HIS2
                     *
3E8A 2740            HIS4  EORI,R3     H'40'      CHECK FOR BINARY
3E8C 1807                  BCTR,Z      HIS5       YES - GOOD STATUS
3E8E 0706                  LODI,R3     SE06       NO - SET SHORT READ
3E90 040D                  LODI,R0     EOL
3E92 CEE3CC                STRA,R0     *PTOB,R2
                     *
3E95 0404            HIS5  LODI,R0     FRDY       SET FCB READY
3E97 1B49                  BCTR,UN     HIS1
                     *
3E99 0418            HSIO  LODI,R0     H'18'      TURN OFF HSPT
3E9B D4D0                  WRTE,R0     PCPT
3E9D 0488                  LODI,R0     H'88'      ENABLE AND DRIVE LEFT
3E9F D4D0                  WRTE,R0     PCPT
3EA1 17                    RETC,UN
                     *
                     *
                     *
                     *
                     ******************************
                     *                            *
                     *   HIGH SPEED PAPER TAPE    *
                     *       READER HANDLER       *
                     *                            *
                     ******************************
                     *
                     *
3EA2 01              HPTH  LODZ        R1         GET FCB INDEX
3EA3 1A31                  BCTR,N      HPT5       RETURN TO DISPATCHER IF DQ
                     *
3EA5 20                    EORZ        R0
3EA6 C2                    STRZ        R2
3EA7 C3                    STRZ        R3
                     *
3EA8 0D6150                LODA,R0     FCS,R1     GET FCB STATE
3EAB 2406                  EORI,R0     FABT       CHECK FOR ABORT
3EAD 181C                  BCTR,EQ     HPT3       YES - FREE FCB
                     *
3EAF C3                    STRZ        R3         SET I/O COMPLETE FLAG
3EB0 E403                  COMI,R0     3          CHECK FOR FCS = EOF
3EB2 9804                  BCFR,EQ     HPT1       JIF NOT EOF
                     *
3EB4 06FF                  LODI,R2     SEFF       SET EOF ERROR CODE
3EB6 1B16                  BCTR,UN     HPT4
```

```
TWIN ASSEMBLER VER 1.0       HSPT VER R.2 7/30/76          PAGE 0009

LINE ADDR  OBJECT  E SOURCE

0449                 *
0450 3EB8 0D61A8     HPT1   LODA,R0     FCOD,R1   GET FUNCTION CODE
0451 3EBB 4407              ANDI,R0     7
0452 3EBD 180A              BCTR,Z      HPT2      JIF ASSIGN
0453 3EBF 2403              EORI,R0     CLOP
0454 3EC1 1808              BCTR,Z      HPT3      JIF CLOSE
0455 3EC3 E402              COMI,R0     WTOP      REALLY READ
0456 3EC5 1812              BCTR,EQ     HPT6      JIF READ
0457 3EC7 0605              LODI,R2     SE05      SET ILLEGAL FUNCTION CODE
0458 3EC9 0404       HPT2   LODI,R0     FRDY      SET FCB RDY CODE
0459 3ECB CD6150     HPT3   STRA,R0     FCS,R1    STORE FCB STATE
0460 3ECE 02         HPT4   LODZ        R2        GET STATUS
0461 3ECF CD61D4            STRA,R0     FSTA,R1   STORE
0462 3ED2 03                LODZ        R3        GET CALL IOC3 INDICATOR
0463 3ED3 BC936C            BSFA,Z      *IOC3     CALL IF INDICATED
0464                 *
0465 3ED6 1F200F     HPT5   BCTA,UN     DISP      GO TO DISPATCHER
0466                 *
0467 3ED9 060A       HPT6   LODI,R2     SE10      ASSUME DEVICE NOT OPERATIONAL
0468 3EDB 54D0              REDE,R0     PCPT      GET DEVICE STATUS
0469 3EDD CD61BE            STRA,R0     FPDS,R1
0470 3EE0 247F              EORI,R0     H'7F      CHECK STATUS
0471 3EE2 9865              BCFR,EQ     HPT2      DEVICE NOT OPERATIONAL
0472                 *
0473 3EE4 0603              LODI,R2     PTDV      GET HSPT DEVIVE INDEX
0474 3EE6 3F936A            BSTA,UN     *SDCB     SET UP DEVICE CONTROL BLOCK
0475                 *
0476 3EE9 7620              PPSU        H'20'     DISABLE INTERRUPTS
0477                 *
0478 3EEB 3F3E99            BSTA,UN     HSIO      START INPUT
0479 3EEE 1B66              BCTR,UN     HPT5      GO TO DISPATCHER
0480 0000                   END         0

 TOTAL ASSEMBLY ERRORS = 0000
```

signetics

a subsidiary of **U.S. Philips Corporation**

Signetics Corporation
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700