

**iSBC 80/30  
SINGLE BOARD COMPUTER  
HARDWARE  
REFERENCE MANUAL**

Manual Order Number: 9800611A

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Intel Corporation.

The following are trademarks of Intel Corporation and may be used only to describe Intel products:

ICE  
INSITE  
INTEL  
INTELLEC  
iSBC

LIBRARY MANAGER  
MCS  
MEGACHASSIS  
MICROMAP  
MULTIBUS

PROMPT  
RMX  
UPI  
μSCOPE



This manual provides general information, installation, programming information, principles of operation, and service information for the Intel iSBC 80/30 Single Board Computer. Additional information is available in the following documents:

- *Intel MCS-85 User's Manual*, Order No. 98-366
- *Intel UPI-41 User's Manual*, Order No. 9800504
- *Intel 8080/8085 Assembly Language Programming Manual*, Order No. 9800301
- *Intel 8255A Programmable Peripheral Interface*, Application Note AP-15
- *Intel 8251 Universal Synchronous/Asynchronous Receiver/Transmitter*, Application Note AP-16
- *Intel MULTIBUS Interfacing*, Application Note AP-28
- *Intel 8259 Programmable Interrupt Controller*, Application Note AP-31



# CONTENTS

## CHAPTER 1

### GENERAL INFORMATION

	PAGE
Introduction .....	1-1
Description .....	1-1
System Software Development .....	1-3
Equipment Supplied .....	1-4
Equipment Required .....	1-4
Specifications .....	1-4

## CHAPTER 2

### PREPARATION FOR USE

Introduction .....	2-1
Unpacking and Inspection .....	2-1
Installation Considerations .....	2-1
User-Furnished Components .....	2-1
Power Requirements .....	2-1
Cooling Requirement .....	2-1
Physical Dimensions .....	2-1
Component Installation .....	2-3
ROM/EPROM Chips .....	2-3
Universal Peripheral Interface .....	2-4
Line Drivers and I/O Terminators .....	2-4
Rise Time/Noise Capacitors .....	2-4
Jumper Configuration .....	2-4
ROM/EPROM Configuration .....	2-4
On-Board RAM Addresses .....	2-4
On-Board 8085A Access .....	2-7
System Access .....	2-7
Priority Interrupts .....	2-8
8251A Port Configuration .....	2-12
8255A Port Configuration .....	2-12
8041/8741A Port Configuration .....	2-12
Multibus Configuration .....	2-15
Signal Characteristics .....	2-15
Serial Priority Resolution .....	2-23
Parallel Priority Resolution .....	2-24
Power Fail/Memory Protect Configuration .....	2-24
Parallel I/O Cabling .....	2-27
Serial I/O Cabling .....	2-27
Board Installation .....	2-29

## CHAPTER 3

### PROGRAMMING INFORMATION

Introduction .....	3-1
Failsafe Timer .....	3-1
Memory Addressing .....	3-1
I/O Addressing .....	3-2
System Initialization .....	3-2
8251A USART Programming .....	3-3
Mode Instruction Format .....	3-3
Sync Characters .....	3-3
Command Instruction Format .....	3-3
Reset .....	3-4
Addressing .....	3-4
Initialization .....	3-4

Operation .....	3-5
Data Input/Output .....	3-5
Status Read .....	3-6
8253 PIT Programming .....	3-6
Mode Control Word and Count .....	3-7
Addressing .....	3-10
Initialization .....	3-10
Operation .....	3-11
Counter Read .....	3-11
Clock Frequency/Divide Ratio Selection .....	3-11
Rate Generator/Interval Timer .....	3-12
Interrupt Timer .....	3-12
8255A PPI Programming .....	3-13
Control Word Format .....	3-13
Addressing .....	3-14
Initialization .....	3-14
Operation .....	3-14
Read Operation .....	3-14
Write Operation .....	3-14
8259A PIC Programming .....	3-14
Interrupt Priority Modes .....	3-14
Fully Nested Mode .....	3-14
Auto-Rotating Mode .....	3-15
Specific Rotating Mode .....	3-15
Interrupt Mask .....	3-15
Status Read .....	3-15
Initialization Command Words .....	3-15
Operation Command Words .....	3-16
Addressing .....	3-16
Initialization .....	3-16
Operation .....	3-16
8041/8741A UPI Programming .....	3-17
Interrupt Handling .....	3-22
TRAP Input .....	3-22
RST 7.5, 6.5, 5.5 Inputs .....	3-23
INTR Input .....	3-23

## CHAPTER 4

### PRINCIPLES OF OPERATION

Introduction .....	4-1
Functional Description .....	4-1
Clock Circuits .....	4-1
Central Processor Unit .....	4-1
Interval Timer .....	4-1
Serial I/O .....	4-1
Parallel I/O .....	4-1
Universal Peripheral Interface .....	4-2
Interrupt Control .....	4-2
ROM/EPROM Configuration .....	4-2
RAM Configuration .....	4-2
Bus Interface .....	4-2
Dual-Port Control .....	4-2
Circuit Analysis .....	4-2



# CONTENTS (Continued)

	PAGE		PAGE
Initialization .....	4-2	On-Board I/O Operation .....	4-13
Clock Circuits .....	4-3	System I/O Operation .....	4-13
8085A CPU Timing .....	4-3	ROM/EPROM Operation .....	4-13
Instruction Timing .....	4-3	RAM Operation .....	4-14
Opcode Fetch Timing .....	4-4	RAM Controller .....	4-14
Memory Read Timing .....	4-6	On-Board Read/Write Operation .....	4-14
I/O Read Timing .....	4-6	Bus Read/Write Operation .....	4-14
Memory Write Timing .....	4-6	Interrupt Operation .....	4-15
I/O Write Timing .....	4-7		
Interrupt Acknowledge Timing .....	4-7	<b>CHAPTER 5</b>	
Address Bus .....	4-9	<b>SERVICE INFORMATION</b>	
Bus Time Out .....	4-9	Introduction .....	5-1
Data Bus .....	4-9	Replaceable Parts .....	5-1
Read/Write Signal Generation .....	4-9	Service Diagrams .....	5-1
I/O Control Signals .....	4-9	Service and Repair Assistance .....	5-1
Memory Control Signals .....	4-9		
Dual Port Control Logic .....	4-9	<b>APPENDIX A</b>	
Bus Access Timing .....	4-12	<b>8085 INSTRUCTION SET</b>	
CPU Access Timing .....	4-12		
Multibus Interface .....	4-12	<b>APPENDIX B</b>	
I/O Operation .....	4-13	<b>TELETYPEWRITER MODIFICATIONS</b>	



# TABLES

TABLE	TITLE	PAGE	TABLE	TITLE	PAGE
1-1.	Specifications .....	1-4	2-15.	iSBC 80/30 AC Characteristics (Master Mode) .....	2-21
2-1.	User-Furnished and Installed Components .....	2-2	2-16.	iSBC 80/30 AC Characteristics (Slave Mode) .....	2-21
2-2.	User-Furnished Connector Details .....	2-3	2-17.	Auxiliary Connector P2 Pin Assignments .....	2-25
2-3.	Line Driver and I/O Terminator Locations .....	2-4	2-18.	Auxiliary Signal (Connector P2) DC Characteristics .....	2-26
2-4.	Jumper Selectable Options .....	2-5	2-19.	Connector J1 Pin Assignments .....	2-27
2-5.	ROM/EPROM Configuration Jumpers .....	2-7	2-20.	Connector J2 Pin Assignments .....	2-27
2-6.	Jumper Configuration for On-Board 8085A CPU Access of On-Board RAM .....	2-7	2-21.	Parallel I/O Signal (Connector J1/J2) DC Characteristics .....	2-28
2-7.	65K Page System Memory Selection .....	2-8	2-22.	Connector J3 Vs. RS232C Pin Correspondence .....	2-29
2-8.	8K/16K Block Selection Within 65K Page .....	2-8	3-1.	On-Board Memory Address (for CPU Access) .....	3-1
2-9.	Priority Interrupt Jumper Matrix .....	2-11	3-2.	I/O Address Assignments .....	3-2
2-10.	Connector J3 Pin Assignments Vs. Jumper Configuration .....	2-12	3-3.	Typical USART Mode or Command Instruction Subroutine .....	3-5
2-11.	8255A Port Configuration Jumpers .....	2-13	3-4.	Typical USART Data Character Read Subroutine .....	3-6
2-12.	Multibus Connector P1 Pin Assignments .....	2-16			
2-13.	Multibus Signal Functions .....	2-17			
2-14.	iSBC 80/30 DC Characteristics .....	2-18			



# TABLES (Continued)

TABLE	TITLE	PAGE	TABLE	TITLE	PAGE
3-5.	Typical USART Data Character Write Subroutine .....	3-6	3-18.	Typical PIC Initialization Subroutine .....	3-17
3-6.	Typical USART Status Read Subroutine .....	3-7	3-19.	PIC Operation Procedures .....	3-19
3-7.	PIT Counter Operation Vs. Gate Inputs .....	3-10	3-20.	Typical PIC Interrupt Request Register Read Subroutine .....	3-20
3-8.	Typical PIT Control Word Subroutine .....	3-10	3-21.	Typical PIC In-Service Register Read Subroutine .....	3-20
3-9.	Typical PIT Count Value Load Subroutine ..	3-11	3-22.	Typical PIC Set Mask Register Subroutine ..	3-20
3-10.	Typical PIT Counter Read Subroutine .....	3-11	3-23.	Typical PIC Mask Register Read Subroutine ..	3-21
3-11.	PIT Count Value Vs. Rate Multiplier for Each Baud Rate .....	3-12	3-24.	Typical PIC End-of-Interrupt Command Subroutine .....	3-21
3-12.	PIT Rate Generator Frequencies and Timer Intervals .....	3-13	3-25.	Typical UPI Data Byte Read Subroutine .....	3-22
3-13.	PIT Time Intervals Vs. Timer Counts .....	3-13	3-26.	Typical UPI Data Byte Write Subroutine .....	3-22
3-14.	Typical PPI Initialization Subroutine .....	3-14	3-27.	8085A CPU Restart Interrupt Vectors .....	3-22
3-15.	Typical PPI Port Read Subroutine .....	3-14	4-1.	CPU Status and Control Lines .....	4-4
3-16.	Typical PPI Port Write Subroutine .....	3-15	5-1.	Replaceable Parts .....	5-1
3-17.	PIC Device Address Insertion .....	3-16	5-2.	List of Manufacturers' Codes .....	5-3



# ILLUSTRATIONS

FIGURE	TITLE	PAGE	FIGURE	TITLE	PAGE
1-1.	iSBC 80/30 Single Board Computer .....	1-1	3-12.	PPI Port C Bit Set/Reset Control Word Format .....	3-15
2-1.	Jumper Configuration for Multibus Access of On-Board RAM (16-Bit Address System) ..	2-9	3-13.	PIC Device Interrupt Addresses .....	3-17
2-2.	Jumper Configuration for Multibus Access of On-Board RAM (20-Bit Address System) ..	2-10	3-14.	PIC Initialization Command Word Formats ..	3-17
2-3.	Bus Exchange Timing (Master Mode) .....	2-22	3-15.	PIC Operation Control Word Formats .....	3-18
2-4.	Bus Exchange Timing (Slave Mode) .....	2-23	3-16.	UPI Data Bus Buffer and Status Registers .....	3-21
2-5.	Serial Priority Resolution Scheme .....	2-24	4-1.	iSBC 80/30 Input/Output and Interrupt Block Diagram .....	4-17
2-6.	Parallel Priority Resolution Scheme .....	2-25	4-2.	iSBC 80/30 ROM/EPROM and Dual Port RAM Block Diagram .....	4-19
3-1.	USART Synchronous Mode Instruction Word Format .....	3-3	4-3.	Typical CPU Instruction Cycle .....	4-4
3-2.	USART Synchronous Mode Transmission Format .....	3-3	4-4.	Opcode Fetch Machine Cycle .....	4-5
3-3.	USART Asynchronous Mode Instruction Word Format .....	3-3	4-5.	Opcode Fetch Machine Cycle (With Wait) .....	4-5
3-4.	USART Asynchronous Mode Transmission Format .....	3-4	4-6.	Memory Read (or I/O Read) Machine Cycles ..	4-6
3-5.	USART Command Instruction Word Format ..	3-4	4-7.	Memory Write (or I/O Write) Machine Cycles ..	4-7
3-6.	Typical USART Initialization and I/O Data Sequence .....	3-5	4-8.	Interrupt Acknowledge Machine Cycles .....	4-8
3-7.	USART Status Read Format .....	3-7	4-9.	Dual Port Control Bus Access Timing With CPU Lockout .....	4-10
3-8.	PIT Mode Control Word Format .....	3-8	4-10.	Dual Port Control CPU Access Timing With Bus Lockout .....	4-11
3-9.	PIT Programming Sequence Examples .....	3-9	5-1.	iSBC 80/30 Parts Locations Diagram .....	5-5
3-10.	PIT Counter Register Latch Control Word Format .....	3-12	5-2.	iSBC 80/30 Schematic Diagram .....	5-7
3-11.	PPI Control Word Format .....	3-13	5-3.	iSBC 604 Schematic Diagram .....	5-25
			5-4.	iSBC 614 Schematic Diagram .....	5-27



# CHAPTER 1 GENERAL INFORMATION

## 1-1. INTRODUCTION

The iSBC 80/30 Single Board Computer, which is a member of Intel's complete line of iSBC 80 computer products, is a computer system on a single printed-circuit assembly. The iSBC 80/30 includes a central processor unit (CPU), 16K bytes of dynamic random access memory (RAM), one serial and three parallel I/O ports, a programmable timer, priority interrupt logic, and Multibus control logic. Also included is dual port control logic to allow the iSBC 80/30 to act as a slave RAM device to other bus masters in the system. Provision is made for user-installation of masked or programmable read only memory (ROM or EPROM) and an Intel 8041 or 8741A Universal Peripheral Interface.

## 1-2. DESCRIPTION

The iSBC 80/30 Single Board Computer (figure 1-1) is controlled by an Intel 8085A Microprocessor (CPU), which includes six 8-bit general-purpose registers and an accumulator. The six general-purpose registers may be addressed individually or in pairs, which allows both single precision and double precision operations. The CPU has a 16-bit program counter which allows direct

addressing of up to 65K of memory. An external stack, located within any portion of read/write memory, may be used as a last-in/first-out storage area for the contents of the program counter, flags, accumulator, and all six general-purpose registers. A 16-bit stack pointer controls the addressing of this external stack, which allows subroutine nesting that is bounded only by the 65K address limitation.

The iSBC 80/30 has an internal bus for all on-board memory and I/O operations and accesses the system bus (Multibus) for all external memory and I/O operations. Hence, local (on-board) operations do not involve the Multibus and allow true parallel processing when several bus masters (e.g., DMA devices and other single board computers) are used in a multimaster scheme.

The 16K of dynamic RAM is implemented with eight Intel 2717 chips and an Intel 8202 Dynamic RAM Controller. Dual port control logic is included to interface this 16K RAM with the Multibus so that the iSBC 80/30 can function as a slave RAM device when not in control of the Multibus. The CPU has priority when accessing on-board RAM. After the CPU completes its read or write operation, the controlling bus master is allowed to access RAM and complete its operation. Where both the CPU and the controlling bus master have the need to write or read

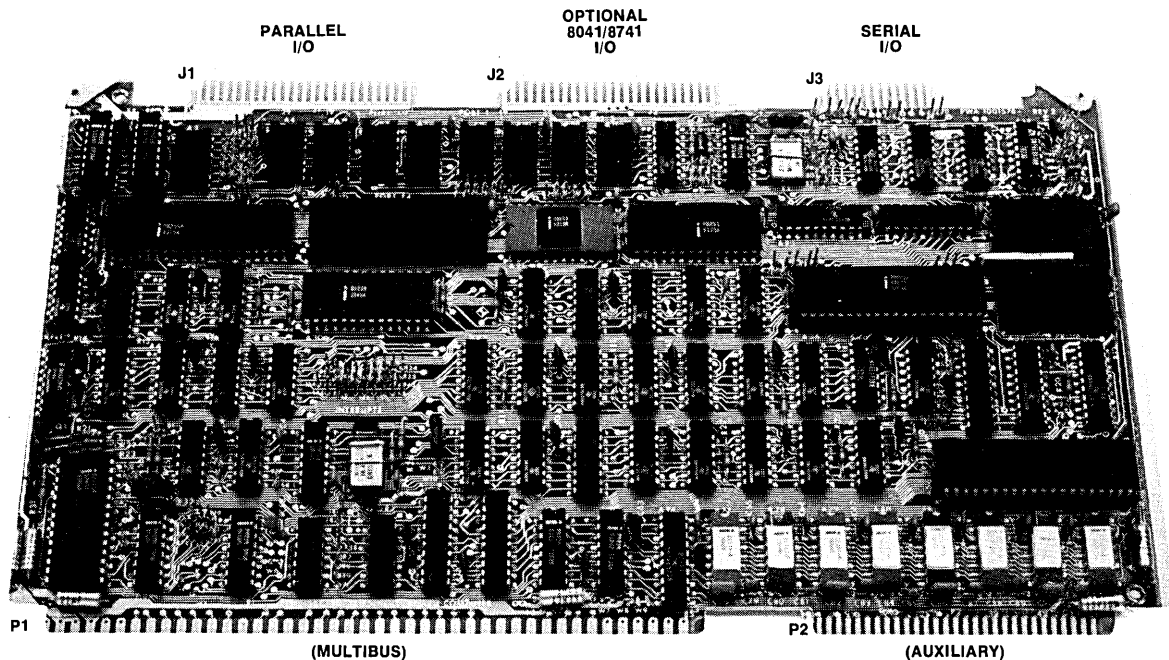


Figure 1-1. iSBC 80/30 Single Board Computer

several words to or from on-board RAM, their operations are interleaved. The slave RAM decode logic allows extended Multibus addressing so that bus masters having a 20-bit address capability can partition the iSBC 80/30 RAM into any 8K or 16K segment in a 1-megabyte address space. The CPU, however, has only 16 address lines and memory must therefore reside in the 0-65K byte address space. There is no conflict in assigning RAM addresses for CPU access and slave access since separate decoding logic is used.

Jumpers are included to allow the user to reserve 8K bytes of on-board RAM for use by the 8085A CPU only. This reserved RAM address space is not accessible via the Multibus and does not occupy any system address space.

Two IC sockets are included to accommodate up to 8K of user-installed ROM or EPROM. Configuration jumpers allow ROM or EPROM to be installed in 1K, 2K, or 4K increments. All on-board ROM/EPROM operations are performed at maximum processor speed.

The iSBC 80/30 includes 24 programmable parallel I/O lines implemented by means of an Intel 8255A Programmable Peripheral Interface (PPI). The system software is used to configure the I/O lines in any combination of unidirectional input/output and bidirectional ports. The I/O interface may be customized to meet specific peripheral requirements and, in order to take full advantage of the large number of possible I/O configurations, IC sockets are provided for interchangeable I/O line drivers and terminators. Hence, the flexibility of the parallel I/O interface is further enhanced by the capability of selecting the appropriate combination of optional line drivers and terminators to provide the required sink current, polarity, and drive/termination characteristics for each application. The 24 programmable I/O lines and signal ground lines are brought out to a 50-pin edge connector (J1) that mates with flat, woven, or round cable.

Sockets are provided for a user-supplied Intel 8041/8741A Universal Programmable Interface (UPI) and associated line drivers and terminators. The 8041/8741A is a single-chip microcomputer which contains a CPU, 1K bytes of ROM (8041) or EPROM (8741), 64 bytes of RAM, 16 programmable I/O lines, and an 8-bit timer. Special interface registers are included in the chip which enable the UPI to function as a slave processor to the 8085A CPU. The UPI allows the user to specify algorithms for controlling user peripherals directly in the chip, thereby relieving the 8085A CPU for other system functions. An RS232C driver and an RS232C receiver are included so that the UPI may optionally be used to handle a simple serial I/O interface. In addition to providing the capability of user-supplied algorithms for the 8041/8741A the iSBC 80/30 supports all the preprogrammed 8041/8741A devices such as the Intel 8278 Keyboard Encoder, 8294 Data Encryption Controller, and 8295 Matrix Printer Driver.

The RS232C compatible serial I/O port is controlled and interfaced by an Intel 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter) chip. The USART is individually programmable for operation in most synchronous or asynchronous serial data transmission formats (including IBM Bi-Sync).

In the synchronous mode the following are programmable:

- a. Character length,
- b. Sync character (or characters), and
- c. Parity.

In the asynchronous mode the following are programmable:

- a. Character length,
- b. Baud rate factor (clock divide ratios of 1, 16, or 64).
- c. Stop bits, and
- d. Parity.

In both the synchronous and asynchronous modes, the serial I/O port features half- or full-duplex, double-buffered transmit and receive capability. In addition, USART error detection circuits can check for parity, overrun, and framing errors. The USART transmit and receive clock rates are supplied by a programmable baud rate/time generator. These clocks may optionally be supplied from an external source. The RS232C command lines, serial data lines, and signal ground lines are brought out to a 50-pin edge connector (J3) that mates with flat or round cable.

Three independent, fully programmable 16-bit interval timer/event counters are provided by an Intel 8253 Programmable Interval Timer (PIT). Each counter is capable of operating in either BCD or binary modes; two of these counters are available to the systems designer to generate accurate time intervals under software control. Routing for the outputs and gate/trigger inputs of two of these counters is jumper-selectable; the outputs of these two counters may be independently routed to the 8259A Programmable Interrupt Controller (PIC), the I/O line drivers associated with the 8255A Programmable Peripheral Interface (PPI), the 8041/8741A Universal Programmable Interface, or used as inputs to the 8255A PPI and 8041/8741A (UPI). The gate/trigger inputs of the two counters may be routed to I/O terminators associated with the 8255A PPI or as output connections from the 8255A PPI. The third counter is used as a programmable baud rate generator for the serial I/O port. In utilizing the iSBC 80/30, the systems designer simply configures, via software, each counter independently to meet system requirements. Whenever a given time delay or count is needed, software commands to the 8253 PIT select the desired function. The contents of each counter may be read at any time during system operation with simple operations for event counting applications, and special commands are included so that the contents of each counter can be read "on the fly."



The iSBC 80/30 provides vectoring for 12 interrupt levels, four of which are handled directly by the interrupt processing capability of the 8085A CPU. These four levels (TRAP, RST 7.5, RST 6.5, and RST 5.5) represent (in decreasing order of priority) the four highest priority interrupts of the iSBC 80/30. These four interrupts generate the following unique memory address: TRAP (24H), RST 7.5 (3CH), RST 6.5 (34H), and RST 5.5 (2CH). An 8085A JUMP instruction at each of these addresses then provides linkage to interrupt service routines located independently anywhere in the lower 65K bytes of memory. All interrupt inputs with the exception of TRAP may be masked via software. The TRAP interrupt should be used for conditions such as power-down sequences which require immediate attention by the 8085A CPU.

An Intel 8259A Programmable Interrupt Controller (PIC) provides vectoring for the next eight interrupt levels. The PIC treats each true input signal condition as an interrupt request. After resolving the interrupt priority, the PIC issues a single interrupt request to the CPU. Interrupt priorities are independently programmable under software control. Similarly, an interrupt can be masked under software control. The programmable interrupt priority modes are:

- a. Fully Nested Priority. Each interrupt request has a fixed priority: input 0 is highest, input 7 is lowest.
- b. Auto-Rotating Priority. Each interrupt request has equal priority. Each level, after receiving service, becomes the lowest priority level until the next interrupt occurs.
- c. Specific Priority. Software assigns lowest priority. Priority of all other levels is in numerical sequence based on lowest priority.

The PIC, which can be programmed to respond to edge-sensitive or level-sensitive inputs, generates a unique memory address for each interrupt level. These addresses are equally spaced at intervals of 4 to 8 (software selectable) bytes. This 32- or 64-byte block may be located to begin at any 32- or 64-byte boundary in the 65,536 byte memory space. A single 8085A JUMP instruction at each of these addresses then provides linkage to locate each interrupt service routine independently anywhere in memory.

Interrupt requests may originate from 18 sources. Two jumper-selectable interrupt requests can be automatically generated by the Programmable Peripheral Interface (PPI) when a byte of information is ready to be transferred to the 8085A CPU (i.e., input buffer is full) or a byte of information has been transferred to a peripheral device (i.e., output buffer is empty). Two jumper-selectable interrupt requests can be automatically generated by the USART when a character is ready to be transferred to the 8085A CPU (i.e., receive channel buffer is full) or when a character is ready to be transmitted (i.e., transmit channel data buffer is empty). A jumper-selectable interrupt

request can be generated by two of the programmable counters and by the Universal Peripheral Interface (UPI). Eight additional interrupt request lines are available to the user for direct interfaces to user designated peripheral devices via the Multibus, and two interrupt request lines may be jumper routed directly from peripherals via the parallel I/O driver/ terminator section.

Control logic is also included for generation of a Power-Fail Interrupt, which works in conjunction with an AC LOW signal from an Intel iSBC 635 Power Supply or equivalent.

The iSBC 80/30 includes the resources for supporting a variety of OEM system requirements. For those applications requiring additional processing capacity and the benefits of multiprocessing (i.e., several CPU's and/or controllers logically sharing systems tasks with communication over the Multibus), the iSBC 80/30 provides full bus arbitration control logic. This control logic allows up to three bus masters (e.g., any combination of iSBC 80/30, iSBC 80/20, DMA controller, diskette controller, etc.) to share the Multibus in serial (daisy-chain) fashion or up to 16 bus masters to share the Multibus using an external parallel priority resolving network.

The Multibus arbitration logic operates synchronously with the bus clock, which is derived either from the iSBC 80/30 or can be optionally generated by some other bus master. Data, however, is transferred via a handshake between the controlling master and the addressed slave module. This arrangement allows different speed controllers to share resources on the same bus, and transfers via the bus proceed asynchronously. Thus, the transfer speed is dependent on transmitting and receiving devices only. This design prevents slower master modules from being handicapped in their attempts to gain control of the bus, but does not restrict the speed at which faster modules can transfer data via the same bus. The most obvious applications for the master-slave capabilities of the bus are multiprocessor configurations, high-speed direct memory access (DMA) operations, and high-speed peripheral control, but are by no means limited to these three.

### 1-3. SYSTEM SOFTWARE DEVELOPMENT

Intel's RMX/80 Real-Time Multitasking Software, specifically designed for Intel iSBC 80 single board computers, provides the capability to monitor and control multiple asynchronous external events. The RMX/80 Executive, which synchronizes and controls the execution of multiple tasks, is provided as a linkable and relocatable module that requires only 2K bytes of memory. Optional linkable and relocatable modules for

teletypewriter and CRT control, diskette file system, high-speed mathematics unit, and analog subsystems are also available.

The development cycle of iSBC 80/30 based products may be significantly reduced using an Intel Intellec Microcomputer Development System. The resident macroassembler, text editor, and system monitor greatly simplify the design, development, and debug of iSBC 80/30 system software. An optional diskette operating system provides a relocating macroassembler, relocating loader and linkage editor, and a library manager. A unique In-Circuit Emulator (ICE-85) option provides the capability of developing and debugging software directly on the iSBC 80/30.

Intel's high level programming language, PL/M, is also available as a resident Intellec Microcomputer Development System option. PL/M provides the capability to program in a natural, algorithmic language and eliminates the need to manage register usage or allocate memory. PL/M programs can be written in a much shorter time than assembly language programs for a given application.

**1-4. EQUIPMENT SUPPLIED**

The following are supplied with the iSBC 80/30 Single Board Computer:

- a. Schematic diagram, dwg no. 2002132
- b. Assembly drawing, dwg no. 1001576

**1-5. EQUIPMENT REQUIRED**

Because the iSBC 80/30 is designed to satisfy a variety of applications, the user must purchase and install only those components required to satisfy his particular needs. A list of components required to configure all the intended applications of the iSBC 80/30 is provided in table 2-1.

**1-6. SPECIFICATIONS**

Specifications of the iSBC 80/30 Single Board Computer are listed in table 1-1.

**Table 1-1. Specifications**

<b>WORD SIZE</b>	
Instruction:	8, 16, or 24 bits.
Data:	8 bits.
<b>CYCLE TIME:</b>	1.44 $\mu$ sec $\pm$ 0.1% for fastest executable instruction; i.e., four clock cycles.
<b>MEMORY CAPACITY</b>	
On-Board ROM/EPROM:	Up to 8K bytes; user installed in 1K, 2K, or 4K increments.
On-Board RAM:	16K bytes of dynamic RAM; integrity maintained during power failure with user-furnished batteries.
Off-Board Expansion:	Up to 65K bytes of user-specified combinations of RAM, ROM, and EPROM.
<b>MEMORY ADDRESSING</b>	
On-Board ROM/EPROM:	0-07FF (using 2708 or 2758 EPROM's or 8308 ROM's); 0-0FFF (using 2716 EPROM's or 2316E ROM's); 0-1FFF (using 2332 ROM's).
On-Board RAM (CPU Access):	Jumpers allow on-board CPU to access either 8K or 16K. For 8K RAM access, addresses may be set on 8K boundaries 2000, 4000, ... E000. For 16K access, addresses may be set on 16K boundaries 4000, 8000, or C000. One or both 8K segments may be reserved for CPU use only.
On-Board RAM (Multibus Access):	Jumpers allow board to act as slave for 8K or 16K RAM access by another bus master; 16-bit or 20-bit addressing is accommodated and addresses are irrespective of addresses used for CPU access. For 16-bit addressing, boundaries may be set on any 8K or 16K segment of 65K byte address space. For 20-bit addressing, boundaries may be set on any 8K or 16K segment of 1M byte address space.

Table 1-1. Specifications (Continued)

## SERIAL COMMUNICATIONS

## Synchronous:

5-, 6-, 7-, or 8-bit characters.  
Internal; 1 or 2 sync characters.  
Automatic sync insertion.

## Asynchronous:

5-, 6-, 7-, or 8-bit characters.  
Break character generation.  
1, 1½, or 2 stop bits.  
False start bit detection.

## Sample Baud Rate:

Frequency <sup>1</sup> (kHz, Software Selectable)	Baud Rate (Hz) <sup>2</sup>		
	Synchronous	Asynchronous	
		÷ 16	÷ 64
153.6	—	9600	2400
76.8	—	4800	1200
38.4	38400	2400	600
19.2	19200	1200	300
9.6	9600	600	150
4.8	4800	300	75
2.4	2400	150	—
1.76	1760	110	—

Notes: 1. Frequency selected by I/O writes of appropriate 16-bit frequency factor to Baud Rate Register.

2. Baud rates shown here are only a sample subset of possible software-programmable rates available. Any frequency from 18.75 Hz to 614.4 kHz may be generated utilizing on-board crystal oscillator and 16-bit Programmable Interval Timer (used here as frequency divider).

## INTERVAL TIMER AND BAUD RATE GENERATOR

## Input Frequency (selectable):

2.46 MHz  $\pm$ 0.1% (0.41  $\mu$ sec period nominal),  
1.23 MHz  $\pm$ 0.1% (0.82  $\mu$ sec period nominal), and  
153.6 kHz  $\pm$ 0.1% (6.5  $\mu$ sec period nominal).

## Output Frequencies:

Function	Single Timer		Dual Timers (Two Timers Cascaded)	
	Min.	Max.	Min.	Max.
Real-Time Interrupt Interval	1.63 $\mu$ sec	426 msec	3.26 $\mu$ sec	465 28 minutes
Rate Generator (Frequency)	2.348 Hz	614.4 kHz	0.000036 Hz	307.2 kHz

## SYSTEM CLOCK (8085A CPU):

2.7648 MHz  $\pm$ 0.1%.

**Table 1-1. Specifications (Continued)**

<b>I/O ADDRESSING:</b>	All communication to Parallel I/O and Serial I/O Ports, Timer, and Interrupt Controller is via read and write commands from on-board 8085A CPU. Refer to table 3-2.																										
<b>INTERFACE COMPATIBILITY</b> Serial I/O:	EIA Standard RS232C signals provided and supported: <table border="0"> <tr> <td>Carrier Detect</td> <td>Receive Data</td> </tr> <tr> <td>Clear to Send</td> <td>Ring Indicator</td> </tr> <tr> <td>Data Set Ready</td> <td>Secondary Receive Data</td> </tr> <tr> <td>Data Terminal Ready</td> <td>Secondary Transmit Data</td> </tr> <tr> <td>Request to Send</td> <td>Transmit Clock</td> </tr> <tr> <td>Receive Clock</td> <td>Transmit Data</td> </tr> </table>	Carrier Detect	Receive Data	Clear to Send	Ring Indicator	Data Set Ready	Secondary Receive Data	Data Terminal Ready	Secondary Transmit Data	Request to Send	Transmit Clock	Receive Clock	Transmit Data														
Carrier Detect	Receive Data																										
Clear to Send	Ring Indicator																										
Data Set Ready	Secondary Receive Data																										
Data Terminal Ready	Secondary Transmit Data																										
Request to Send	Transmit Clock																										
Receive Clock	Transmit Data																										
Parallel I/O:	24 programmable lines (8 lines per port); one port includes bidirectional bus driver. IC sockets included for user installation of line drivers and/or I/O terminators as required for interface ports. Refer to table 2-1.																										
Optional I/O:	IC socket included for Intel 8041/8741 Universal Peripheral Interface (UPI). Also included are IC sockets for user installation of line drivers and/or I/O terminators as required for interface. Refer to table 2-1.																										
<b>INTERRUPTS:</b>	8085A CPU includes five interrupt inputs, each of which vectors processor to the following unique memory locations for entry point to service routine:																										
	<table border="1"> <thead> <tr> <th>Interrupt Input</th> <th>Vector Address</th> <th>Priority</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>TRAP</td> <td>24H</td> <td>Highest</td> <td>Non-maskable</td> </tr> <tr> <td>RST 7.5</td> <td>3CH</td> <td rowspan="4" style="text-align: center;">↑ ↓</td> <td>Maskable</td> </tr> <tr> <td>RST 6.5</td> <td>34H</td> <td>Maskable</td> </tr> <tr> <td>RST 5.5</td> <td>2CH</td> <td>Maskable</td> </tr> <tr> <td>INTR</td> <td>Note</td> <td>Lowest</td> <td>Maskable</td> </tr> <tr> <td colspan="4" style="text-align: center;">Note: INTR input provided by 8259A PIC, which is programmable to provide vector CALL address of service routine for interrupting device.</td> </tr> </tbody> </table>	Interrupt Input	Vector Address	Priority	Type	TRAP	24H	Highest	Non-maskable	RST 7.5	3CH	↑ ↓	Maskable	RST 6.5	34H	Maskable	RST 5.5	2CH	Maskable	INTR	Note	Lowest	Maskable	Note: INTR input provided by 8259A PIC, which is programmable to provide vector CALL address of service routine for interrupting device.			
Interrupt Input	Vector Address	Priority	Type																								
TRAP	24H	Highest	Non-maskable																								
RST 7.5	3CH	↑ ↓	Maskable																								
RST 6.5	34H		Maskable																								
RST 5.5	2CH		Maskable																								
INTR	Note		Lowest	Maskable																							
Note: INTR input provided by 8259A PIC, which is programmable to provide vector CALL address of service routine for interrupting device.																											
	Jumpers allow selection of 12 priority interrupts from 18 interrupt sources. PIC may be programmed to respond to edge-sensitive or level-sensitive inputs.																										
<b>COMPATIBLE CONNECTORS/CABLES:</b>	Refer to table 2-2 for compatible connector details. Refer to paragraphs 2-29 and 2-30 for recommended types and lengths of I/O cables.																										
<b>ENVIRONMENTAL REQUIREMENTS</b> Operating Temperature:	0° to 55°C (32° to 131°F).																										
Relative Humidity:	To 90% without condensation.																										
<b>PHYSICAL CHARACTERISTICS</b> Width:	30.48 cm (12.00 inches).																										
Depth:	17.15 cm (6.75 inches).																										
Thickness:	1.27 cm (0.50 inch).																										
Weight:	425 gm (15 ounces).																										

Table 1-1. Specifications (Continued)

POWER REQUIREMENTS:				
CONFIGURATION	$V_{CC} = +5V \pm 5\%$	$V_{DD} = +12V \pm 5\%$	$V_{BB} = -5V \pm 5\%$	$V_{AA} = -12V \pm 5\%$
Without EPROM <sup>1</sup>	3.5A	220 mA	—	50 mA
With 8041/8741 UPI <sup>2</sup>	3.6A	220 mA	—	50 mA
RAM Only <sup>3</sup>	350 mA	20 mA	2.5 mA	—
With iSBC 530 <sup>4</sup>	3.5A	320 mA	—	150 mA
With 2K EPROM <sup>5</sup> (using 8708)	4.4A	350 mA	95 mA	40 mA
With 2K EPROM <sup>5</sup> (Using 8758)	4.6A	220 mA	—	50 mA
With 4K EPROM <sup>5</sup> (Using 2716)	(4.6A)	(220 mA)	—	(50 mA)
With 8K ROM <sup>5</sup> (Using 2332)	4.6A	220 mA	—	50 mA

Notes: 1. Does not include power for optional ROM/EPROM, 8041/8741 UPI, I/O drivers, and I/O terminators.  
 2. Does not include power required for optional ROM/EPROM, I/O drivers and I/O terminators.  
 3. RAM chips powered via auxiliary power bus.  
 4. Does not include power for optional ROM/EPROM, 8041/8741 UPI, I/O drivers, and I/O terminators. Power for iSBC 530 is supplied via serial port connector.  
 5. Includes power required for two ROM/EPROM chips, 8041/8741 UPI, and I/O terminators installed for 34 I/O lines; all terminator inputs low.



## 2-1. INTRODUCTION

This chapter provides instructions for preparing the iSBC 80/30 Single Board Computer for use in the user-defined environment. It is advisable that the contents of Chapters 1 and 3 be fully understood before beginning the configuration and installation procedures provided in this chapter.

## 2-2. UNPACKING AND INSPECTION

Inspect the shipping carton immediately upon receipt for evidence of mishandling during transit. If the shipping carton is severely damaged or waterstained, request that the carrier's agent be present when the carton is opened. If the carrier's agent is not present when the carton is opened and the contents of the carton are damaged, keep the carton and packing material for the agent's inspection.

For repairs to a product damaged in shipment, contact the Intel Technical Support Center (see paragraph 5-4) to obtain a Return Authorization Number and further instructions. A purchase order will be required to complete the repair. A copy of the purchase order should be submitted to the carrier with your claim.

It is suggested that salvageable shipping cartons and packing material be saved for future use in the event the product must be reshipped.

## 2-3. INSTALLATION CONSIDERATIONS

The iSBC 80/30 is designed for use in one of the following configurations:

- Standalone (single-board) system.
- Bus master in a single bus master system.
- Bus master in a multiple bus master system.

Important criteria for installing and interfacing the iSBC 80/30 in the above environments are presented in following paragraphs.

## 2-4. USER-FURNISHED COMPONENTS

Because the iSBC 80/30 is designed to satisfy a variety of applications, the user need purchase and install only those components required to satisfy his particular configuration. A list of components required to configure all the intended applications of the iSBC 80/30 are listed in table 2-1. Table 2-2 lists details, types, and vendors of those connectors referenced in table 2-1.

## 2-5. POWER REQUIREMENTS

The iSBC 80/30 requires +5V, -5V, +12V, and -12V power supply inputs. The currents required from these supplies are listed in table 1-1. (The -5V supply is *mandatory* only if Intel 2708 EPROM chips are installed; an on-board regulator that operates off the -12V supply can otherwise supply the -5V power.)

## 2-6. COOLING REQUIREMENT

The iSBC 80/30 dissipates 401 gram-calories/minute (1.62 Btu/minute) and adequate circulation of air must be provided to prevent a temperature rise above 55°C (131°F). The System 80 enclosures and the Intellec System include fans to provide adequate intake and exhaust of ventilating air.

## 2-7. PHYSICAL DIMENSIONS

Physical dimensions of the iSBC 80/30 are as follows:

- Width: 30.48 cm (12.00 inches).
- Height: 17.15 cm (6.75 inches).
- Thickness: 1.25 cm (0.50 inch).

Table 2-1. User-Furnished and Installed Components

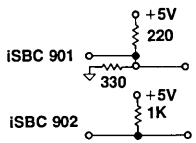
Item No.	Item	Description	Use		
1	iSBC 604	Modular Backplane and Cardcage. Includes four slots with bus terminators.  (See figure 5-3.)	Provides power input pins and Multibus signal interface between iSBC 80/30 and three additional boards in a multiple board system.		
2	iSBC 614	Modular Backplane and Cardcage. Includes four slots without bus terminators. (See figure 5-4.)	Provides four-slot extension of iSBC 604.		
3	Connector (mates with P1)	See Multibus connector details in table 2-2.	Power inputs and Multibus signal interface Not required if iSBC 80/30 is installed in an iSBC 604/614.		
4	Connector (mates with P2)	See Auxiliary connector details in table 2-2.	Auxiliary backup battery inputs and associated memory protect functions.		
5	Connector (mates with J1)	See parallel I/O connector details in table 2-2.	Interfaces parallel I/O ports to Intel 8255A Programmable Peripheral Interface (PPI).		
6	Connector (mates with J2)	See parallel I/O connector details in table 2-2.	Interfaces I/O ports to optional Intel 8041/8741A Universal Peripheral Interface (UPI).		
7	Connector (mates with J3)	See serial I/O connector details in table 2-2.	Interfaces serial I/O port to Intel 8251A Programmable Communications Interface (USART).		
8	ROM/EPROM Chips	One or two each of the following Intel ROM/EPROM chips:	On-board UV erasable PROM for program development and/or dedicated program use. Compatible ROM chips can also be employed. Use either ROM or EPROM; do not mix.		
		ROM		EPROM	BITS
		8308 — 2316E 2332		2708 2758 2716 —	1K × 8 1K × 8 2K × 8 4K × 8
9	Intel 8041/8741A	Universal Peripheral Interface (UPI).	Single chip microcomputer with program memory, data memory, CPU, event timer, I/O, and clock oscillator. Interfaces two 8-bit I/O ports; two additional input bits (T0 and T1) for conditional branch and event timer functions.		
10	Line Drivers	Type	Current	Used for interface to Intel 8255A and optional Intel 8041/8741. Requires two line driver IC's for each 8-bit parallel output port. (Exception: refer to paragraph 2-11.)	
		SN7403 I, OC SN7400 I SN7408 NI SN7409 NI, OC	16 mA 16 mA 16 mA 16 mA		
		Types selected as typical; I = inverting, NI = noninverting, and OC = open collector.			
11	I/O Terminators	Intel iSBC 901 Divider or iSBC 902 Pull-Up:  	Used for interface to Intel 8255A and optional Intel 8041/8741A. Requires two 901's or two 902's for each 8-bit parallel input port. (Exception: refer to paragraph 2-11.) Additional 901 or 902 required for 8041/8741 if T0 and T1 inputs are used for conditional branch or event timer functions.		
12	Capacitors	Seven capacitors as required.	Rise time/noise capacitors for serial I/O port.		

Table 2-2. User-Furnished Connector Details

Function	No. Of Pairs/Pins	Centers (Inches)	Connector Type	Vendor	Vendor Part No.	Intel Part No.
Parallel I/O Connector	25/50	0.1	Flat Crimp	3M 3M AMP ANSLEY SAE	3415-0000 WITH EARS 3415-0001 W/O EARS 88083-1 609-5015 SD6750 SERIES	iSBC 956 Cable Set
Parallel I/O Connector	25/50	0.1	Soldered	AMP VIKING TI	2-583485-6 3VH25/1JV5 H312125	N/A
Parallel I/O Connector	25/50	0.1	Wirewrap <sup>1</sup>	TI VIKING CDC <sup>3</sup> ITT CANNON	H311125 3VH25/1JND5 VPB01B25D00A1 EC4A050A1A	N/A
Serial I/O Connector	13/26	0.1	Flat Crimp	3M AMP ANSLEY SAE	3462-0001 88106-1 609-2615 SD6726 SERIES	iSBC 955 Cable Set
Serial I/O Connector	13/26	0.1	Soldered	TI AMP	H312113 1-583485-5	N/A
Serial I/O Connector	13/26	0.1	Wirewrap <sup>1</sup>	TI	H311113	N/A
Multibus Connector	43/86	0.156	Soldered <sup>1</sup>	CDC <sup>3</sup> MICRO PLASTICS ARCO VIKING	VPB01E43D00A1 MP-0156-43-BW-4 AE443WP1 LESS EARS 2VH43/1AV5	N/A
Multibus Connector	43/86	0.156	Wirewrap <sup>1,2</sup>	CDC <sup>3</sup> CDC <sup>3</sup> VIKING	VFB01E43D00A1 or VPB01E43A00A1 2VH43/1AV5	MDS 985
Auxiliary Connector	30/60	0.1	Soldered <sup>1</sup>	TI VIKING	H312130 3VH30/1JN5	N/A
Auxiliary Connector	30/60	0.1	Wirewrap <sup>1,2</sup>	CDC <sup>3</sup> TI	VPB01B30A00A2 H311130	N/A
NOTES:						
1. Connector heights are not guaranteed to conform to OEM packaging equipment.						
2. Wirewrap pin lengths are not guaranteed to conform to OEM packaging equipment.						
3. CDC VPB01 ..., VPB02 ..., VPB04 ..., etc. are identical connectors with different electroplating thicknesses or metal surfaces.						

## 2-8. COMPONENT INSTALLATION

Instructions for installing the optional ROM/EPROM, Intel 8041/8741A Universal Peripheral Interface, line drivers, I/O terminators, and rise time/noise capacitors are given in following paragraphs. When installing the optional chips, be sure to orient pin 1 of the chip adjacent to the white dot located near pin 1 of the associated IC socket. The grid location on figure 5-1 (parts location diagram) and figure 5-2 (schematic diagram) are specified for each user-installed component. Because the schematic diagram consists of nine sheets, grid references to figure 5-2

consist of four alphanumeric characters. For example, grid reference 6ZD4 signifies sheet 6 Zone D4.

## 2-9. ROM/EPROM CHIPS

Install the ROM/EPROM chips in IC sockets A25 and A37. (Refer to figure 5-1 zone ZC3 and figure 5-2 zone 3ZA3.) Sockets A25 and A37, respectively, accommodate the low order and high-order addresses of the ROM/EPROM chip pair. For instance, if two Intel 2716 EPROM chips are installed, the chip installed in IC socket A25 is assigned addresses 0000-07FF; the chip installed in IC socket A37 is assigned addresses 0800-0FFF.



The default (factory connected) jumpers are configured for Intel 2316E ROM or 2716 EPROM. If different type chips are installed, reconfigure the jumpers as described in paragraph 2-14.

## 2-10. UNIVERSAL PERIPHERAL INTERFACE

Install the optional Intel 8041/8741A Universal Peripheral Interface (UPI) chip in socket A20. (Refer to figure 5-1 zone ZC6 and figure 5-2, zone 5ZC4.)

## 2-11. LINE DRIVERS AND I/O TERMINATORS

Table 2-3 lists the I/O ports and the location of associated IC sockets for installing either line drivers or I/O terminators. (Refer to table 2-1 items 10 and 11.) Port E8 is factory equipped with Intel 8226 Bidirectional Bus Drivers and requires no additional components. (Refer to paragraphs 2-22 and 2-23.)

## 2-12. RISE TIME/NOISE CAPACITORS

Eye pads are provided so that rise time/noise capacitors may be installed as required on the individual serial I/O pins. The selection of capacitor values is at the option of the user and is normally a function of the particular environment. The location of these eye pads are as follows:

Capacitor	Fig. 5-1	Fig. 5-2
C11	ZD4	6ZD4
C12	ZD4	6ZB3
C13	ZD3	6ZC4
C14	ZD4	6ZD4
C16	ZD3	6ZD6
C17	ZD3	6ZC6
C18	ZD3	6ZD6

## 2-13. JUMPER CONFIGURATION

The iSBC 80/30 includes a variety of jumper-selectable options to allow the user to configure the board for his particular application. Table 2-4 summarizes these jumper-selectable options and lists the grid reference locations of the jumpers as shown in figure 5-1 (parts location diagram) and figure 5-2 (schematic diagram). Because the schematic consists of nine sheets, grid references to figure 5-2 consists of four alphanumeric characters. For example, grid reference 3ZB7 signifies sheet 3 Zone B7.

Study table 2-5 carefully while making reference to figures 5-1 and 5-2. If the default (factory configured) jumper wiring is appropriate for a particular function, no further actions is required for that function. If, however, a different configuration is required, remove the default jumper(s) and install an optional jumpers(s) as specified. For most options, the information in table 2-4 is sufficient for proper configuration. Additional information, where necessary for clarity, is described in subsequent paragraphs.

## 2-14. ROM/EPROM CONFIGURATION

Table 2-5 lists the jumper configurations and associated address block for the various types of compatible Intel ROM/EPROM chips.

## 2-15. ON-BOARD RAM ADDRESSES

This on-board RAM can be accessed by the on-board 8085A microprocessor (CPU) as well as by other bus masters in the system via the Multibus. Addresses for on-board 8085A access and for system access are assigned as described in paragraphs 2-16 and 2-17, respectively.

Table 2-3. Line Driver and I/O Terminator Locations

	I/O Port	Bits	Driver/ Terminator	Fig. 5-1 Grid Ref.	Fig. 5-2 Grid Ref
8255A PPI Interface	E8	0-7	None Required	—	—
	E9	0-3 4-7	A5 A6	ZD6 ZD6	4ZA3 4ZA3
	EA	0-3 4-7	A4 A3	ZD7 ZD7	4ZC3 4ZB3
8041/8741 UPI Interface (Optional)	1	0-3 4-7	A7 A8	ZD6 ZD6	5ZD3 5ZC3
	2	0-3 4-7	A10 A11	ZD5 ZD5	5ZB3 5ZB3
	—	T0, T1	A9	ZD5	5ZC3

Table 2-4. Jumper Selectable Options

Function	Fig. 5-1 Grid Ref.	Fig. 5-2 Grid Ref.	Description
ROM/EPROM Configuration	ZC3 ZC3 ZC3 ZD2 ZD2 ZD2	3ZB7 3ZC6 3ZB4 3ZA3 3ZA4 3ZA3	Six jumpers accommodate one of four types of user-installed ROM or EPROM chips. One jumper required between two posts in each of the following six groups of posts:  112 thru 114 157 thru 159 153 thru 156 84 thru 88 99 thru 101 102 thru 105 } Default jumpers accommodate Intel 2716 EPROM or 2316E ROM chips. Refer to paragraph 2-14 if reconfiguration is required.
On-Board RAM (On-Board Access)	ZD2	3ZD7, 3ZD6	One jumper wire selects 8K or 16K access and one jumper wire selects base address for on-board 8085A access of on-board RAM. Jumper W1 default position *A-B selects 16K access and default jumper *98-92 selects base address 4000H. Refer to paragraphs 2-15 and 2-16 if reconfiguration is required.
On-Board RAM (System Access)	ZB7 ZB7 ZB7	5ZC6 5ZB6 5ZB5	16-Bit Address System: Three jumper wires select base address for system access:  W5 (one): Must be set to position *K-L. W4 (one): 8K or 16K access. 171 thru 180 (one): Base address. Refer to paragraphs 2-17 and 2-18.
	ZB7 ZB7 ZB7 ZB7	5ZC7 5ZC6 5ZB6 5ZB5	20-Bit Address System: Five jumper wires select base address for system access:  W6 (two): Upper or lower 512K space. W5 (one): 65K page select. W4 (one): 8K or 16K access. 171 thru 180 (one): Base address. Refer to paragraphs 2-17 and 2-19.
Bus Clock	ZB7	5ZA4	Default jumper *165-166 routes Bus Clock signal BCLK/ to the Multibus. (Refer to table 2-13.) Remove this jumper <i>only</i> if another bus master supplies this signal.
Constant Clock	ZB7	5ZA4	Default jumper *167-168 routes Constant Clock signal CCLK/ to the Multibus. (Refer to table 2-13.) Remove this jumper <i>only</i> if another bus master supplies this signal.
Bus Priority Out	ZB7	8ZD2	Default jumper *169-170 routes Bus Priority Out signal BPRO/ to the Multibus (Refer to table 2-13.) Remove this jumper <i>only</i> in those systems employing a parallel priority bus resolution scheme. (Refer to paragraph 2-27.)
Bus Priority Resolution	ZB7	8ZD6	One jumper defines one of two modes of resolving bus contention in a multiple bus master system:  *162-163: Can request Multibus as needed. 163-164: Always requesting Multibus; should only be used when iSBC 80/30 has lowest priority.
Auxiliary Backup Batteries	ZB5	1ZC5, 1ZB5	If auxiliary backup batteries are employed to sustain memory during ac power outages, remove default jumpers *W8, *W9, and *W10.
On-Board -5V Regulator	ZB6	1ZB2	The 80/30 requires a -5V supply for Intel 2708 EPROM chips and a -5V AUX input for the on-board RAM chips. The system -5V supply is mandatory only if Intel 2708 EPROM chips are employed. If Intel 2708 EPROM chips are not employed and no system -5V supply is available, the -5 AUX input to the on-board RAM chips can be supplied by the on-board -5V regulator or by an auxiliary backup battery. (The on-board -5V regulator operates from the system -12V supply.) If neither a system -5V supply nor an auxiliary backup battery is used, enable the -5V regulator by connecting jumper *W10.

Table 2-4. Jumper Selectable Options (Continued)

Function	Fig. 5-1 Grid Ref.	Fig. 5-2 Grid Ref.	Description
Failsafe Timer	ZC8	1ZC6	If the on-board 8085A CPU addresses either a system or an on-board memory or I/O device and that device does not return an acknowledge signal, the 8085A will hang up in a wait state. A failsafe timer is triggered during $T_1$ of every machine cycle and, if not retriggered within 10 milliseconds, the resultant time-out pulse can be used to allow the 8085A to exit the wait state. If this feature is desired, connect jumper 115-116.
RAM Refresh	ZC4	2ZA3	The 8202 Memory Controller for on-board RAM is jumper selectable as follows to select the automatic or invisible refresh mode: *110-111: Automatic refresh (normal) mode. 110-106: Invisible refresh (on request) mode.  In the <i>automatic</i> refresh mode, a wait state can be imposed on the on-board 8085A if a memory access occurs while a memory refresh cycle is in progress. In this case, the 8085A is forced to wait until the refresh cycle is complete. In the <i>invisible</i> refresh mode, the 8202 works around memory accesses by refreshing memory during the instruction decode clock cycle which follows each instruction fetch.
Timer Input Frequency	ZD5  ZD5  ZD5	7ZB3  7ZA4  7ZB5	Input frequencies to the 8253 Programmable Interval Timer counters are jumper selectable as follows: <u>Counter 0</u> (8041/8741 Event Clock) *47-52: Same as Counter 2. 47-51: 153.6 kHz. 47-48: External Event Clock (from Port EA). <u>Counter 1</u> (Count Out) *46-54: 1.2288 MHz. 46-52: Same as Counter 2. 46-50: Counter 0 output. 46-48: External Event Clock (from Port EA). Jumper 46-50 effectively connects Counter 0 and 1 in series in which the output of Counter 0 serves as the input clock to Counter 1. This permits lower clock rates to Counter 1 and, in turn, Counter 1 provides longer time intervals. <u>Counter 2</u> (8251A Baud Rate Clock) *53-54: 1.2288 MHz. 53-49: 2.4576 MHz.
Event Clock	ZD5	7ZB4	The Event Clock is a jumper option for Port C (Port EA) of the 8255A PPI. The Event Clock may be provided by the following optional sources: 48-50: Same as PIT Counter 0 (8041/8741 Event Clock). 48-51: 153.6 kHz. 48-52: Same as PIT Counter 2 clock.
Priority Interrupt	ZC6 ZC6 ZB6	Sheet 7	A jumper matrix provides a wide selection of interrupts to be interfaced to the on-board 8085A and the Multibus. The matrix includes the following jumpers (refer to paragraph 2-20): 117 thru 134 136 thru 152 181 thru 190
8251A Serial I/O Port Configuration	—	Sheet 6	Jumpers are used to input the serial I/O port transmit clock, receive clock, ring indicator, carrier detect, etc., from several sources. Refer to paragraph 2-21.
8255A Parallel I/O Port Configuration	—	Sheet 4	Jumpers are used to configure Ports E8, E9, and EA for the desired operating mode. Refer to paragraph 2-22.
8041/8741A Universal Peripheral Interface	—	Sheet 5	Jumpers are used to select various input and/or output signals depending on the application. Refer to paragraph 2-23.
*Default jumper connected at the factory.			

**Table 2-5. ROM/EPROM Configuration Jumpers**

ROM/EPROM Type	Jumpers	Address Block
2708 EPROM or 8308 ROM	112-113, 158-159, 100-101, 104-105, 155-154, 86-84	0000-07FF
2758 EPROM	112-113, 158-159, 100-101, 104-103, 155-154, 86-87	0000-07FF
*2716 EPROM or *2316E ROM	112-113, 157-158, 100-101, 104-103, 155-156, 86-85	0000-0FFF
2332 ROM	112-114, 157-158, 100-99, 104-102, 155-153, 86-85	0000-1FFF
*Default jumpers are connected for type 2716 EPROM or 2316E ROM. Disconnect and reconfigure jumpers as necessary if installing different type ROM/EPROM.		

**2-16. ON-BOARD 8085A ACCESS.** Jumper W1 position B-C allows the on-board 8085A to access 8K of the 16K RAM; jumper W1 default position A-B allows access to all 16K. For 8K access, the RAM address block may be configured on 8K boundaries 2000, 4000, . . . E000; for 16K access, the RAM address block may be configured on 16K boundaries 4000, 8000, or C000. (Address boundary 0000 is reserved for ROM/EPROM.) Default jumper 98-92 selects boundary 4000 for both 8K and 16K access. If only 8K access is desired, remove jumper W1 from position A-B and install it in position B-C. If a different address boundary is desired, reconfigure the jumpers as listed in table 2-6.

**2-17. SYSTEM ACCESS.** The on-board RAM can be shared by other bus masters in the system via the Multibus. If one or more bus masters have a 20-bit address capability, the extended addressing jumpers allow the onboard RAM to reside anywhere within a 1-megabyte address space. (The on-board 8085A can only access memory in the lower 65K address space.) If it is not desired to have the on-board RAM shared by the system, leave default jumper 180-179 installed. Otherwise, configure the jumpers for 16-bit addressing or 20-bit addressing as described in following paragraphs.

**NOTE**

Addresses for system access of on-board RAM are completely independent of addresses for on-board 8085A access of on-board RAM.

**Table 2-6. Jumper Configuration for On-Board 8085A Access of On-Board RAM**

Jumper	RAM Address Block <sup>1</sup>	
	8K Access <sup>2</sup>	16K Access <sup>3</sup>
98-91	2000-3FFF	
*98-92	4000-5FFF	4000-7FFF
98-93	6000-7FFF	
98-94	8000-9FFF	8000-BFFF
98-95	A000-BFFF	
98-96	C000-DFFF	C000-FFFF
98-97	E000-FFFF	
NOTES:		
1. Address block 0000-1FFF is reserved for ROM/EPROM.		
2. Requires jumper W1 position B-C installed.		
3. Requires jumper W1 position *A-B installed.		
*Default jumper; disconnect if reconfiguration is required.		

**2-18. 16-BIT ADDRESS SYSTEMS.** For system access, the on-board RAM is divided into eight 65K pages. In 16-bit address systems, there is no page selection capability and jumper W5 default position K-L restricts the addresses to Page 0 (the only page in a 16-bit system). (Refer to table 2-7.)

As shown in table 2-8, the system can access either 8K or 16K of the on-board RAM. Jumper W4 default position B-A limits the system to 8K access; jumper W4 position B-C allows access of all 16K of on-board RAM.

One jumper wire places the on-board RAM in the desired 8K or 16K segment of the selected 65K page. To access 8K, for example, the 8K segment can be placed on any 8K boundary 0000 (1st 8K), 2000 (2nd 8K), 4000 (3rd 8K), . . . E000 (8th 8K). To access 16K, the 16K segment can be placed on any 16K boundary 0000 (1st 16K), 4000 (2nd 16K), 8000 (3rd 16K), or C0000 (4th 16K). Figure 2-1 illustrates a step-by-step sequence for establishing RAM addresses for 16-bit address systems.

**2-19. 20-BIT ADDRESS SYSTEMS.** In 20-bit address systems, the on-board RAM can reside anywhere within a 1-megabyte address space. As shown in table 2-7, the RAM is first placed in the lower or upper 524K by jumper W6. Jumper W5 then selects one of eight 65K pages within the upper or lower 524K bytes.

Next, referring to table 2-8, the system can access either 8K or 16K of the on-board RAM. Default jumper W4 position B-A limits the system to 8K access; jumper W4 position B-C allows access of all 16K of on-board RAM.

**Table 2-7. 65K Page System Memory Selection**

Low/(High) <sup>1</sup> System Memory	65K Page No.	Address Range <sup>1</sup>
N/A (Note 2)	0 W5: *K-L	0000-FFFF
0-524K (525-1048K)  W6: *B-C *D-E  W6: (B-E) (D-A)	0 W5: K-A	00000-0FFFF (80000-8FFFF)
	1 W5: K-B	10000-1FFFF (90000-9FFFF)
	2 W5: K-C	20000-2FFFF (A0000-AFFFF)
	3 W5: K-D	30000-3FFFF (B0000-BFFFF)
	4 W5: K-E	40000-4FFFF (C0000-CFFFF)
	5 W5: K-F	50000-5FFFF (D0000-DFFFF)
	6 W5: K-G	60000-6FFFF (E0000-EFFFF)
	7 W5: K-H	70000-7FFFF (F0000-FFFFF)
<p>NOTES:</p> <ol style="list-style-type: none"> <li>Notation in parentheses applies to high (upper 524K) bytes of 20-bit system address space.</li> <li>Systems without 20-bit address capability must use jumper W5 in position *K-L.</li> </ol> <p>*Default Jumper; disconnect if reconfiguration is required. N/A = not applicable.</p>		

**Table 2-8. 8K/16K Block Selection Within 65K Page**

8K or 16K System Access	Address Block Within 65K Page (Refer to Table 2-7)	
	Jumper	Block
8K W4: *B-A	180-172	1st 8K
	180-173	2nd 8K
	180-174	3rd 8K
	180-175	4th 8K
	180-176	5th 8K
	180-177	6th 8K
	180-178	7th 8K
	180-179	8th 8K
	*180-171	No Access
16K W4: B-C	180-173	1st 16K
	180-175	2nd 16K
	180-177	3rd 16K
	180-179	4th 16K
	*180-171	No Access
*Default jumper; disconnect if reconfiguration is desired.		

Finally, one jumper wire places the on-board RAM in the desired 8K or 16K segment of the selected 65K page. To access an 8K segment in Page 4 of the lower 524K, for example, the 8K segment can be placed on any 8K boundary 40000 (1st 8K), 42000 (2nd 8K), 44000 (3rd 8K), . . . 4E000 (8th 8K). To access a 16K segment in Page 4 of the lower 524K, the 16K segment can be placed on any 16K boundary 40000 (1st 16K), 44000 (2nd 16K), 48000 (3rd 16K), or 4C000 (4th 16K). Figure 2-2 illustrates a step-by-step sequence for establishing RAM addresses for a 20-bit address system.

**2-20. PRIORITY INTERRUPTS**

Table 2-9 lists the source (from) and destination (to) of the interrupt matrix shown in figure 5-2 sheet 7. For example, note that the 8259A Programmable Interrupt Controller (PIC) can handle eight positive-true interrupt requests and, after resolving any priority contention, outputs an interrupt request to the INTR input of the 8085A micro-processor.

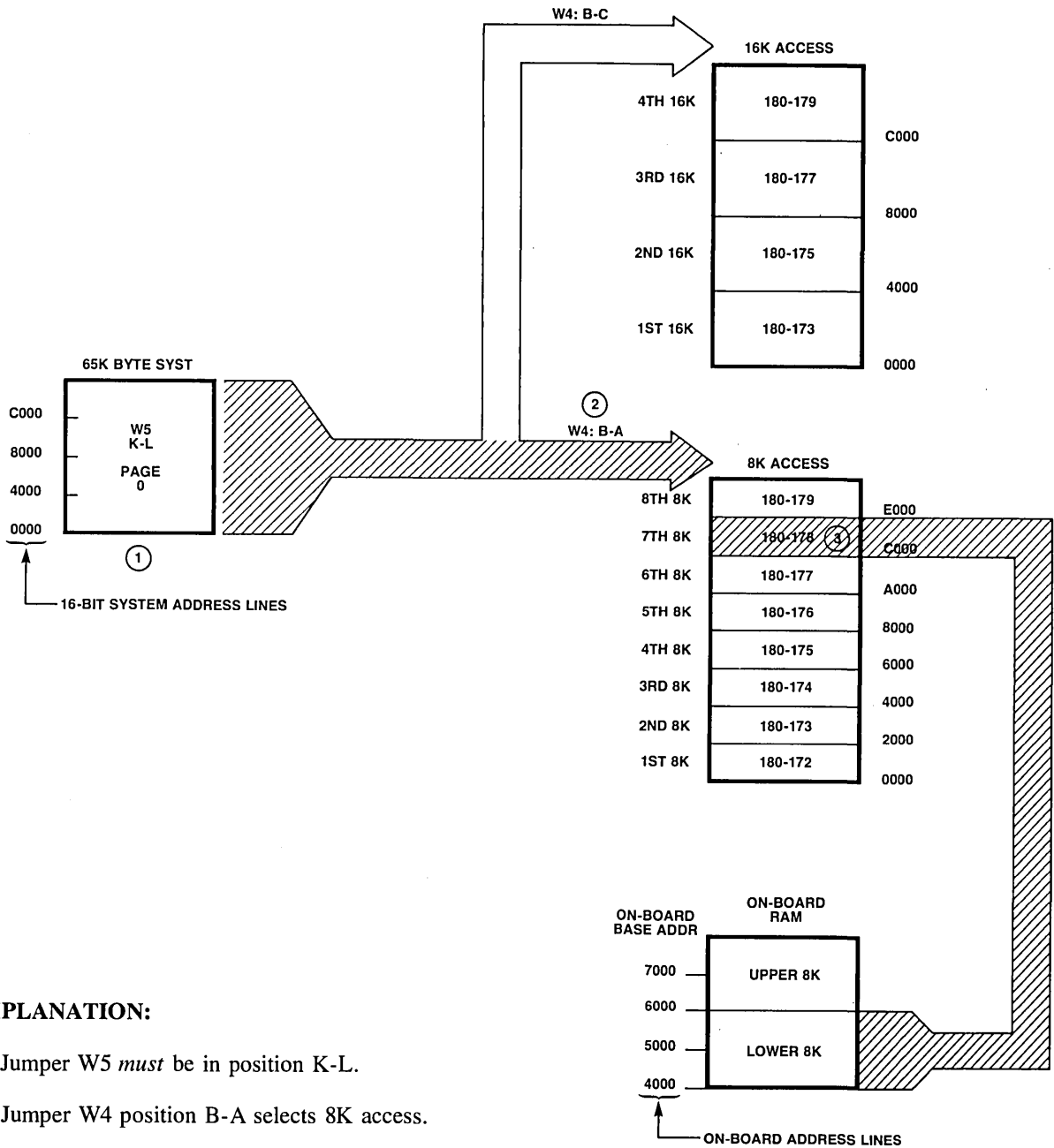
Study table 2-9 carefully while making reference to figure 5-2 sheet 7 before deciding on a definite priority configuration for the iSBC 80/30. There are two areas that require some explanation: the 8085A TRAP and RST 7.5 interrupts.

Default jumper 137-145 grounds the TRAP interrupt input to prevent the possibility of false interrupts being generated by noise spikes. Since the TRAP interrupt is not maskable, cannot be disabled by the program, and has the highest priority, it should be used only to detect a catastrophic event such as a power failure or a bus failure.

The RST 7.5 interrupt input is edge-sensitive only and is default jumpered to the COUNT OUT output of Counter 1. If it is desired to interrupt the 8085A if the Failsafe Timer times out, remove jumper 123-138 (COUNT OUT) and connect jumper 122-138 (BUS TIME OUT). (The BUS TIME OUT signal is generated when the Failsafe Timer is retriggered after timing out.)

**NOTE**

The 8259A PIC can be programmed to respond to either edge-sensitive or level-sensitive interrupt requests. If the PIC is programmed to respond to edge-sensitive interrupt requests, the PIC will respond only to a low-to-high transition on any one of the individual IR input lines.



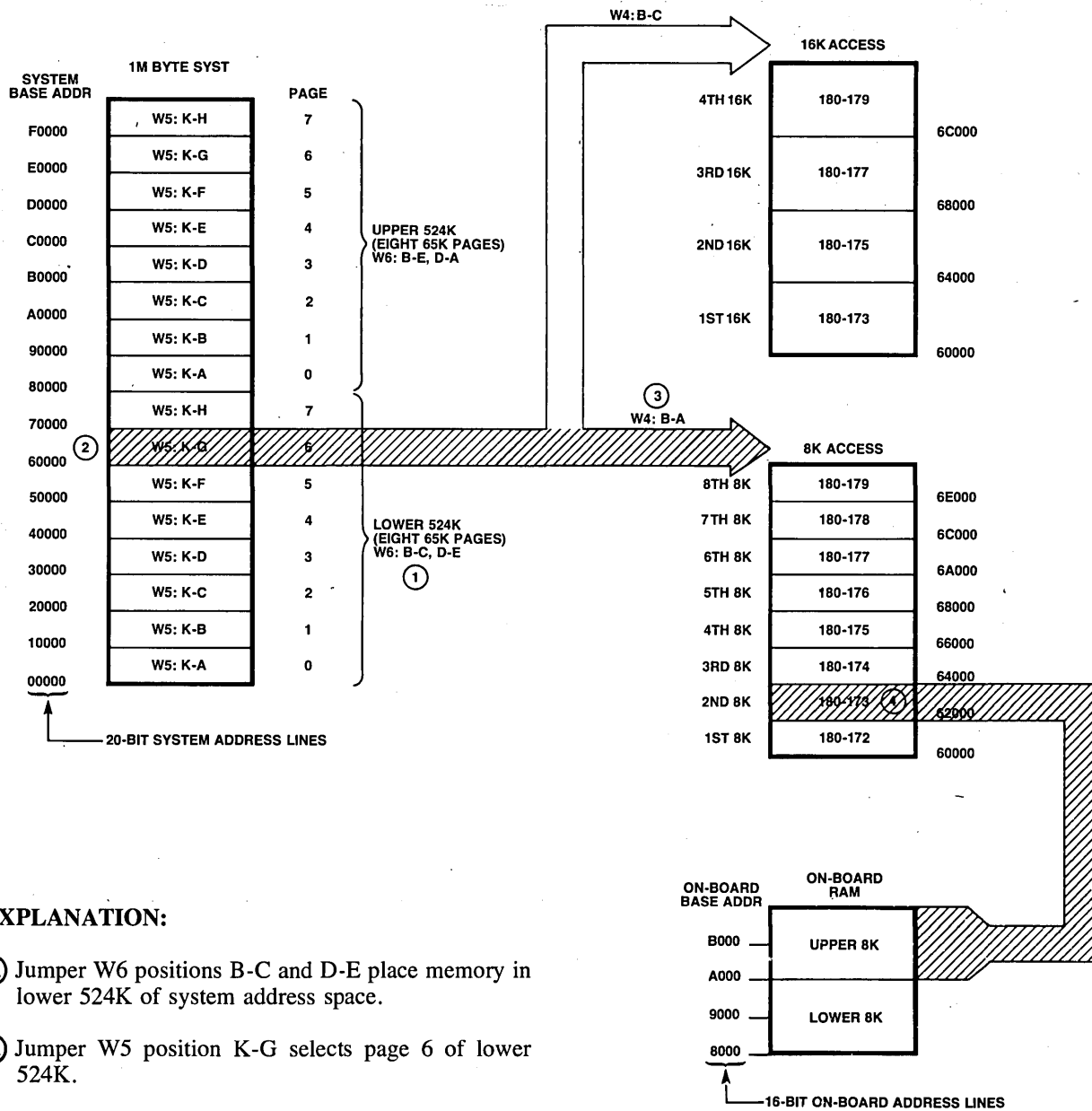
**EXPLANATION:**

- ① Jumper W5 *must* be in position K-L.
- ② Jumper W4 position B-A selects 8K access.
- ③ Jumper 180-178 selects 7th 8K segment of 65K page 0 (the only page in a 16-bit address system).

Thus, the lower 8K of on-board memory is assigned *system* addresses C000 thru DFFF. In this example, this same 8K of memory has *on-board 8085A* addresses A000 thru 5FFF.

Note that for 8K access, the 1st, 3rd, 5th, and 7th 8K bytes access the lower 8K of on-board RAM. The 2nd, 4th, 6th, and 8th 8K bytes access the upper 8K of on-board RAM. With jumper W4 in position B-C, all 16K of on-board RAM is accessed by the system; addresses for system use are established by jumpers 180-173 (1st 16K), 180-175 (2nd 16K), etc.

611-2 Figure 2-1. Jumper Configuration for Multibus Access of On-Board RAM (16-Bit Address System)



**EXPLANATION:**

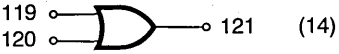
- ① Jumper W6 positions B-C and D-E place memory in lower 524K of system address space.
- ② Jumper W5 position K-G selects page 6 of lower 524K.
- ③ Jumper W4 position B-A selects 8K access of page 6.
- ④ Jumper 180-173 selects 2nd 8K segment of page 6.

Thus, the upper 8K of on-board memory is assigned *system* addresses 62000 thru 63FFF. In this example, this same 8K of memory has *on-board 8085A* addresses A000 thru BFFF.

Note that for 8K access, the 1st, 3rd, 5th, and 7th 8K bytes access the lower 8K of on-board RAM. The 2nd, 4th, 6th, and 8th 8K bytes access the upper 8K of on-board RAM. With jumper W4 in position B-C, all 16K of on-board RAM is accessed by the system; addresses for system use are established by the jumpers 180-173 (1st 16K), 180-175 (2nd 16K), etc.

611-3 **Figure 2-2. Jumper Configuration for Multibus Access of On-Board RAM (20-Bit Address System)**

**Table 2-9. Priority Interrupt Jumper Matrix**

Interrupt Request (From)			Interrupt Request (To)			
Source	Signal	Post	Device	Signal	Post	
Multibus	INT0/ (1)	148	8259A PIC (7)	IR0 IR1 IR2 IR3 IR4 IR5 IR6 IR7	All level sensitive or all edge sensitive (7)	133
	INT1/ (1)	147				132
	INT2/ (1)	152				131
	INT3/ (1)	151				130
	INT4/ (1)	150				129
	INT5/ (1)	149				128
	INT6/ (1)	146				127
	INT7/ (1)	136				126
External Via J2-50	EXT INTR1/ (1)	144	8085A CPU	TRAP (8) RST 7.5 (9) RST 6.5 (10) RST 5.5 (10) INTR (11)	137	
External Via J1-50	EXT INTR0/ (1)	125			138	
Power Fail Logic Via P2-19	PFI/ (1)(2)	134			139	
Failsafe Timer	BUS TIME OUT/ (3)	122			140 None	
8255A PPI Port A (Port E8) Port B (Port E9)	55PAI (1)(4) 55PBI (1)(4)	117 118	Multibus	INTO/ (12) INT1/ (12) INT2/ (12) INT3/ (12) INT4/ (12) INT5/ (12) INT6/ (12) INT7/ (12) ADR10/ (12)	181	
8251A USART	51TXR (1) 51RXR (1)	142 143			182	
8253 PIT Counter 0 Out Counter 1 Out	8041/8741 EVENT CLK (1) COUNT OUT (1)(5)	141			183	
		123			184	
8259A PIC	INTR (6)	None			187	
8041/8741A UPI	41INTR (1)	124			188	
8255A PPI	INTR0UT (13)	185			189	
						

**NOTES:**

- (1) Signal is positive-true at associated jumper post.
- (2) Disconnect 137-145 and connect 134-137 (8085A TRAP).
- (3) Disconnect 123-138 and connect 122-138 (8085A RST 7.5); jumper 115-116 must also be connected. See note (5).
- (4) Used primarily in strobed I/O applications.
- (5) Default jumper 123-138 connects signal to 8085A RST 7.5.
- (6) Applied directly to 8085A INTR input.
- (7) May be programmed for either edge-sensitive or level-sensitive input; IR lines are not individually programmable.
- (8) Default jumper 137-145 disables (grounds) input. TRAP is highest priority, non-maskable, and is both level and edge sensitive.
- (9) Default jumper 123-138 connects input to COUNT OUT. RST 7.5 is second highest priority and is edge sensitive only.
- (10) RST 6.5 and RST 5.5, respectively, are third and fourth highest priority. Both inputs are level sensitive.
- (11) INTR is connected directly to INTR output of 8259A PIC.
- (12) To system via Multibus; requires ground-true signal.
- (13) Signal is ground-true at associated jumper post.
- (14) One two-input OR gate is provided in interrupt matrix.



## 2-21. 8251A PORT CONFIGURATION

Table 2-10 lists the signals, signal functions, and the jumpers required (if necessary) to input or output a particular signal to or from the serial I/O port (Intel 8251A Programmable Communication Interface).

## 2-22. 8255A PORT CONFIGURATION

Table 2-11 lists the jumper configuration for three parallel I/O ports. Note that each of the three ports (E8, E9, and EA) can be configured in a variety of ways to suit the

individual requirement. Recommended line drivers and I/O terminators for user applications are listed in table 2-1.

## 2-23. 8041/8741A PORT CONFIGURATION

The optional Intel 8041/8741A Universal Peripheral Interface can be programmed and, hence, configured to perform serial or parallel I/O functions. Refer to figure 5-2 sheet 5 for jumper details. Applications for the 8041/8741 are presented in the *Intel UPI-41 User's Manual*, Order No. 9800504.

**Table 2-10. Connector J3 Pin Assignments Vs. Jumper Configuration**

Pin <sup>1</sup>	Signal	Function	Jumper In	Jumper Out
2	PROTECTIVE GND	Protective Ground	69-70	—
3	TRANS SIG ELE TIMING (IN)	8251A TXC in (Note 2)	56-57	*55-56
4	TRANSMITTED DATA	8251A RXD in or 8251A TXD out	*82-83 81-82	— *82-83, *80-81
5	SEC REC DATA	8255A STXD out <sup>3</sup> or 8741A RS232 out <sup>4</sup>	76-79, 73-71 76-79, 73-74	— —
6	RECEIVED DATA	8251A TXD out or 8251A RXD in	*80-81 80-83	— *80-81, *82-83
7	REC SIG ELE TIMING	8251A RXC in (Note 2)	59-60	*58-59
8	REQUEST TO SEND	8251A CTS in (Note 5)	—	—
—	(Note 5)	8251A RTS out to CTS in	67-68	—
10	CLEAR TO SEND	8251A RTS out (Note 5)	—	—
—	(Note 5)	8251A RTS out to CTS in	67-68	—
12	DATA SET READY	8251A DTR out	—	—
13	DATA TERMINAL RDY	8251A DSR in	—	—
14	GND	Ground	—	—
16	REG LINE SIG DET	8255A Carrier Detect in <sup>3</sup>	—	—
17	RING INDICATOR	8255A Ring Indicator in <sup>3</sup>	—	—
19	-12V	-12V out	64-65	—
21	TRANS SIG ELE TIMING (OUT)	Same as 8251A TXC in	76-75, 72-73	—
22	+12V	+12V out	63-66	—
23	+5V	+5V out	61-62	—
25	GND	Ground	—	—
26	SEC CLEAR TO SEND	8255A STXD in <sup>3</sup>	76-77	—

**NOTES:**

- All odd-numbered pins (1, 3, 5, . . . 25) are on component side of the board. Pin 1 is the right-most pin when viewed from the component side of the board with the extractors at the top.
- Default jumpers \*55-56 and \*58-59 connect 8253 BAUD RATE CLK to 8251 TXC and RXC inputs, respectively. See Timer Input Frequency (Counter 2) in table 2-4.
- Optional jumper-selected output function for Intel 8255 Programmable Peripheral Interface. Refer to figure 5-2 sheet 4.
- Optional jumper-selected output function for 8041/8741 Universal Peripheral Interface. Refer to figure 5-2 sheet 5.
- Jumper 67-68 connects 8251A RTS output back to CTS input for those applications without CTS capability.

\*Default jumpers connected at the factory.

Table 2-11. 8255A Port Configuration Jumpers

Port	Mode	Driver (D)/ Terminator (T)	Jumper Configuration			Port	Restrictions
			Delete	Add	Effect		
E8	0 Input	8226: A1,A2	None	*7-8	8226 = input enabled.	E9	None; can be in Mode 0 or 1, input or output.
						EA	None; can be in Mode 0, input or output, unless Port E9 is in Mode 1.
E8	0 Output (latched)	8226: A1,A2	*7-8	4-8	8226 = output enabled.	E9	None; can be in Mode 0 or 1, input or output.
						EA	None; can be in Mode 0, input or output, unless Port E9 is in Mode 1.
E8	1 Input (strobed)	8226: A1,A2 T: A3 D: A4	*15-16 and *17-18	*7-8	8226 = input enabled.	E9	None; can be in Mode 0 or 1, input or output.
				*21-22 15-17	Connects J1-26 to STB <sub>A</sub> / input. Connects IBF <sub>A</sub> output to J1-18.	EA	Port EA bits perform the following: <ul style="list-style-type: none"> <li>• Bits 0,1,2 — Control for Port E9 if in Mode 1.</li> <li>• Bit 3 — Port E8 Interrupt (55PAI) to interrupt jumper matrix.</li> <li>• Bit 4 — Port E8 Strobe (STB/) input.</li> <li>• Bit 5 — Port E8 Input Buffer Full (IBF) output.</li> <li>• Bits 6,7 — Port EA input or output (both must be in same direction).</li> </ul>
E8	1 Output (latched)	8226: A1,A2 T: A3 D: A4	*7-8	4-8	8226: output enabled.	E9	None; can be in Mode 0 or 1, input or output.
						*13-14 9-15	Connects J1-30 to ACK <sub>A</sub> / input. Connects OBF <sub>A</sub> / output to J1-18.

Table 2-11. 8255A Port Configuration Jumpers (Continued)

Port	Mode	Driver (D)/ Terminator (T)	Jumper Configuration			Port	Restrictions
			Delete	Add	Effect		
E8	2 (bidirectional)	8226: A1,A2 T: A3 D: A4	*7-8	8-13	Allows ACK <sub>A</sub> / input to control 8226 in/out direction.	E9	None.
			*17-18 and *25-26	*21-22	Connects J1-26 to STB <sub>A</sub> / input.	EA	Port EA bits perform the following: <ul style="list-style-type: none"> <li>• Bit 0 — Can only be used for jumper option (see figure 5-2 zone 4ZC4).</li> <li>• Bits 1,2 — Can be used for input or output if Port E9 is in Mode 0.</li> <li>• Bit 3 — Port E8 Interrupt (55PAI) to interrupt jumper matrix.</li> <li>• Bit 4 — Port E8 Strobe (STB/) input.</li> <li>• Bit 5 — Port E8 Input Buffer Full (IBF) output.</li> <li>• Bit 6 — Port E8 Acknowledge (ACK/) input.</li> <li>• Bit 7 — Port E8 Output Buffer Full (OBF/) output.</li> </ul>
E9	0 Input	T: A5,A6	None	None		E8	None.
						EA	None; Port EA can be in Mode 0, input or output, if Port E8 is also in Mode 0.
E9	0 Output (latched)	D: A5,A6	None	None		E8	None.
						EA	None; Port EA can be in Mode 0, input or output, if Port E8 is also in Mode 0.
E9	1 Input (strobed)	T: A3,A5,A6 D: A4	*19-20 and *9-10	*23-24	Connects IBF <sub>B</sub> output to J1-22.	E8	None.
				10-20	Connects J1-32 to STB <sub>B</sub> / input.	EA	Port EA bits perform the following: <ul style="list-style-type: none"> <li>• Bit 0 — Port E9 Interrupt (55PBI) to interrupt jumper matrix.</li> <li>• Bit 1 — Port E9 Input Buffer Full (IBF) output.</li> <li>• Bit 2 — Port E9 Strobe (STB/) input.</li> <li>• Bit 3 — If Port E8 is in Mode 0, bit 3 can be input or output. Otherwise, bit 3 is reserved.</li> <li>• Bits 4,5 — Depends on Port E8 mode.</li> <li>• Bits 6,7 — Input or output (both must be in same direction).</li> </ul>

Table 2-11. 8255A Port Configuration Jumpers (Continued)

Port	Mode	Driver (D)/ Terminator (T)	Jumper Configuration			Port	Restrictions
			Delete	Add	Effect		
E9	1 Output (latched)	T: A3 D: A4,A5,A6	*25-26	*23-24	Connects OBF <sub>B</sub> / output J1-22.	E8	None.
			*19-20 and *9-10	10-20	Connects J1-32 to ACK <sub>B</sub> / input.	EA	Port EA bits perform the following: <ul style="list-style-type: none"> <li>• Bit 0 — Port E9 Interrupt (55PBI) to interrupt jumper matrix.</li> <li>• Bit 1 — Port E9 Output Buffer Full (OBF/) output.</li> <li>• Bit 2 — Port E9 Acknowl- edge (ACK/) input.</li> <li>• Bit 3 — If Port E8 is in Mode 0, bit 3 can be input or output. Otherwise, bit 3 is reserved.</li> <li>• Bits 4,5 — Input or output (both must be in same direction).</li> <li>• Bits 6,7 — Depends on Port E8 mode.</li> </ul>
EA (upper)	0 Input	T: A3	None	*21-22 *17-18 *13-14 *9-10	Connects bit 4 to J1-26. Connects bit 5 to J1-28. Connects bit 6 to J1-30. Connects bit 7 to J1-32.	E8	Port E8 must be in Mode 0 for all four bits to be available.
						E9	Port E9 must be in Mode 0 for all four bits to be available.
EA (lower)	0 Input	T: A4	None	*25-26 *23-24 *19-20 *15-16	Connects bit 0 to J1-24. Connects bit 1 to J1-22. Connects bit 2 to J1-20. Connects bit 3 to J1-18.	E8	Port E8 must be in Mode 0 for all four bits to be available.
						E9	Port E9 must be in Mode 0 for all four bits to be available.
EA (upper)	0 Output (latched)	D: A3	None	Same as for Port EA (upper) Mode 0 Input.		EA8	Same as for Port EA (upper) Mode 0 Input.
EA (lower)	0 Output (latched)	D: A4	None	Same as for Port EA (lower) Mode 0 Input.		E9	Same as for Port EA (lower) Mode 0 Input.

\*Default jumper connected at the factory.

## 2-24. MULTIBUS CONFIGURATION

For systems applications, the iSBC 80/30 is designed for installation in a standard Intel iSBC 604/614 Modular Backplane and Cardcage. (Refer to table 2-1 items 1 and 2.) Alternatively, the iSBC 80/30 can be interfaced to a user-designed system backplane by means of an 86-pin connector. (Refer to table 2-1 item 3.) Multibus signal characteristics and methods of implementing a serial or parallel priority resolution scheme for resolving bus contention in a multiple bus master system are described in the following paragraphs.



Always turn off the system power supply before installing the board in or removing the board

from the backplane. Failure to observe this precaution can cause damage to the board.

## 2-25. SIGNAL CHARACTERISTICS

As shown in figure 1-1, connector P1 interfaces the iSBC 80/30 to the Multibus. Connector P1 pin assignments are listed in table 2-12 and descriptions of the signal functions are provided in table 2-13.

The dc characteristics of the iSBC 80/30 bus interface signals are provided in table 2-14. The ac characteristics of the iSBC 80/30 when operating in the master mode and slave mode are provided in tables 2-15 and 2-16, respectively. Bus exchange timing diagrams are provided in figures 2-3 and 2-4.

Table 2-12. Multibus Connector P1 Pin Assignments

Pin*	Signal	Function	Pin*	Signal	Function
1	GND	} Ground	44	ADRF/	} Address bus
2	GND		45	ADRC/	
3	+5V	} Power input	46	ADRD/	
4	+5V		47	ADRA/	
5	+5V		48	ADRB/	
6	+5V		49	ADR8/	
7	+12V		50	ADR9/	
8	+12V		51	ADR6/	
9	-5V	} Ground	52	ADR7/	
10	-5V		53	ADR4/	
11	GND	} Address bus	54	ADR5/	
12	GND		55	ADR2/	
13	BCLK/		56	ADR3/	
14	INIT/		57	ADR0/	
15	BPRN/		58	ADR1/	
16	BPRO/		59		
17	BUSY/		60		
18	BREQ/		61		
19	MRDC/		62		
20	MWTC/		63		
21	IORC/		64		
22	IOWC/		65		
23	XACK/	66			
24	INH1/	67	DAT6/	} Data Bus	
25		68	DAT7/		
26		69	DAT4/		
27		70	DAT5/		
28	ADR10/	71	DAT2/		
29	CBRQ/	72	DAT3/		
30	ADR11/	73	DAT0/		
31	CCLK/	74	DAT1/		
32	ADR12/	75	GND	} Ground	
33		76	GND		
34	ADR13/	77		} Power input	
35	INT6/	78			
36	INT7/	79	+12V		
37	INT4/	80	-12V		
38	INT5/	81	+5V		
39	INT2/	82	+5V		
40	INT3/	83	+5V		
41	INT0/	84	+5V	} Ground	
42	INT1/	85	GND		
43	ADRE/		86	GND	

\*All odd-numbered pins (1, 3, 5 . . . 85) are on component side of the board. Pin 1 is the left-most pin when viewed from the component side of the board with the extractors at the top. All unassigned pins are reserved.

Table 2-13. Multibus Signal Functions

Signal	Functional Description
ADR0/-ADRF/ ADR10/-ADR13/	<i>Address.</i> These 20 lines transmit the address of the memory location or I/O port to be accessed. ADRF/ is the most-significant bit except where ADR10/ through ADR13/ are used. ADR10/ through ADR13/ are transmitted only by those bus masters capable of addressing beyond 65K of memory. In this case, ADR13/ is the most-significant bit.
BCLK/	<i>Bus Clock.</i> Used to synchronize the bus contention logic on all bus masters. When generated by the iSBC 80/30, BCLK/ has a period of 108 nanoseconds (9.22 MHz) with a 35-65 percent duty cycle.
BPRN/	<i>Bus Priority In.</i> Indicates to a particular bus master that no higher priority bus master is requesting use of the bus. BPRN/ is synchronized with BCLK/.
BPRO/	<i>Bus Priority Out.</i> In serial (daisy chain) priority resolution schemes, BPRO/ must be connected to the BPRN/ input of the bus master with the next lower bus priority.
BREQ/	<i>Bus Request.</i> In parallel priority resolution schemes, BREQ/ indicates that a particular bus master requires control of the bus for one or more data transfers. BREQ/ is synchronized with BCLK/.
BUSY/	<i>Bus Busy.</i> Indicates that the bus is in use and prevents all other bus masters from gaining control of the bus. BUSY/ is synchronized with BCLK/.
CBRQ/	<i>Common Bus Request.</i> Indicates that a bus master wishes control of the bus but does not presently have control. As soon as control of the bus is obtained, the requesting bus controller raises the CBRQ/ signal.
CCLK/	<i>Constant Clock.</i> Provides a clock signal of constant frequency for use by other system modules. When generated by the iSBC 80/30, CCLK/ has a period of 108 nanoseconds (9.22 MHz) with a 35-65 percent duty cycle.
DAT0/-DAT7/	<i>Data.</i> These eight bidirectional data lines transmit and receive data to and from the addressed memory location or I/O port. DAT7/ is the most-significant bit.
INH1/	<i>Inhibit Ram.</i> Prevents system access to on-board RAM; allows system addresses to overlay on-board RAM for certain applications. This signal has no effect on on-board access of on-board RAM.
INIT/	<i>Initialization.</i> Resets the entire system to a known internal state.
INT0/-INT7/	<i>Interrupt.</i> These eight lines are for inputting interrupt requests to the iSBC 80/30. INT0/ has the highest priority; INT7/ has the lowest priority.
IORC/	<i>I/O Read Command.</i> Indicates that the address of an I/O port is on the Multibus address lines and that the output of that port is to be read (placed) onto the Multibus data lines.
IOWC/	<i>I/O Write Command.</i> Indicates that the address of an I/O port is on the Multibus address lines and that the contents on the Multibus data lines are to be accepted by the addressed port.
MRDC/	<i>Memory Read Command.</i> Indicates that the address of a memory location is on the Multibus address lines and that the contents of that location are to be read (placed) on the Multibus data lines.
MWTC/	<i>Memory Write Command.</i> Indicates that the address of a memory location is on the Multibus address lines and that the contents on the Multibus data lines are to be written into that location.
XACK/	<i>Transfer Acknowledge.</i> Indicates that the addressed memory location has completed the specified read or write operation. That is, data has been placed onto or accepted from the Multibus data lines.

Table 2-14. iSBC 80/30 DC Characteristics

Signals	Symbol	Parameter Description	Test Conditions	Min.	Max.	Units
ADR0/-ADRC/	V <sub>OL</sub>	Output Low Voltage	I <sub>OL</sub> = 50 mA		0.5	V
	V <sub>OH</sub>	Output High Voltage	I <sub>OH</sub> = -10 mA	2.4		V
	V <sub>IL</sub>	Input Low Voltage			0.8	V
	V <sub>IH</sub>	Input High Voltage		2.0		V
	I <sub>IL</sub>	Input Current at Low V	V <sub>IN</sub> = 0.45V		-0.25	mA
	I <sub>IH</sub>	Input Current at High V	V <sub>IN</sub> = 5.25V		80	μA
	I <sub>LH</sub>	Output Leakage High	V <sub>O</sub> = 5.25V		200	μA
	I <sub>LL</sub>	Output Leakage Low	V <sub>O</sub> = 0.45V		200	μA
	*C <sub>L</sub>	Capacitive Load			18	pF
ADRD/-ADRF	V <sub>OL</sub>	Output Low Voltage	I <sub>OL</sub> = 50 mA		0.5	V
	V <sub>OH</sub>	Output High Voltage	I <sub>OH</sub> = 10 mA	2.4		V
	V <sub>IL</sub>	Input Low Voltage			0.8	V
	V <sub>IH</sub>	Input High Voltage		2.0		V
	I <sub>IL</sub>	Input Current at Low V	V <sub>IN</sub> = 0.45V		-0.5	mA
	I <sub>IH</sub>	Input Current at High V	V <sub>IN</sub> = 5.25V		200	μA
		*C <sub>L</sub>	Capacitive Load			25
ADR10/-ADR13/	V <sub>IL</sub>	Input Low Voltage			0.85	V
	V <sub>IH</sub>	Input High Voltage		2.0		V
	I <sub>IL</sub>	Input Current at Low V	V <sub>IN</sub> = 0.45V		-0.25	mA
	I <sub>IH</sub>	Input Current at High V	V <sub>IN</sub> = 5.25V		10	μA
		*C <sub>L</sub>	Capacitive Load			18
BCLK/	V <sub>OL</sub>	Output Low Voltage	I <sub>OL</sub> = 59.5 mA		0.5	V
	V <sub>OH</sub>	Output High Voltage	I <sub>OH</sub> = -3 mA	2.7		V
	V <sub>IL</sub>	Input Low Voltage			0.8	V
	V <sub>IH</sub>	Input High Voltage		2.0		V
	I <sub>IL</sub>	Input Current at Low V	V <sub>IN</sub> = 0.45V		-0.5	mA
	I <sub>IH</sub>	Input Current at High V	V <sub>IN</sub> = 5.25V		100	μA
		*C <sub>L</sub>	Capacitive Load			15
BPRN/	V <sub>IL</sub>	Input Low Voltage			0.8	V
	V <sub>IH</sub>	Input High Voltage		2.0		V
	I <sub>IL</sub>	Input Current at Low V	V <sub>IN</sub> = 0.4V		-1.6	mA
	I <sub>IH</sub>	Input Current at High V	V <sub>IN</sub> = 2.4V		40	μA
		*C <sub>L</sub>	Capacitive Load			18

\*Capacitive load values are approximations.

Table 2-14. iSBC 80/30 DC Characteristics (Continued)

Signals	Symbol	Parameter Description	Test Conditions	Min.	Max.	Units	
BPRO/	$V_{OL}$	Output Low Voltage	$I_{OL} = 3.2 \text{ mA}$	2.4	0.45	V	
	$V_{OH}$	Output High Voltage	$I_{OH} = -400 \mu\text{A}$		V		
	$I_{LH}$	Output Leakage High	$V_O = 5.25\text{V}$		100	$\mu\text{A}$	
	$I_{LL}$	Output Leakage Low	$V_O = 0.45\text{V}$		-100	$\mu\text{A}$	
	* $C_L$	Capacitive Load			15	pF	
BREQ/	$V_{OL}$	Output Low Voltage	$I_{OL} = 20 \text{ mA}$	2.4	0.45	V	
	$V_{OH}$	Output High Voltage	$I_{OH} = -400 \mu\text{A}$		V		
	$V_{IL}$	Input Low Voltage	$V_O = 5.25\text{V}$		100	$\mu\text{A}$	
	$V_{IH}$	Input High Voltage	$V_O = 0.45\text{V}$		-100	$\mu\text{A}$	
	* $C_L$	Capacitive Load			15	pF	
BUSY/, CBRQ/, INTROUT/ (OPEN COLLECTOR)	$V_{OL}$	Output Low Voltage	$I_{OL} = 20 \text{ mA}$		0.45	V	
	* $C_L$	Capacitive Load			20	pF	
CCLK/	$V_{OL}$	Output Low Voltage	$I_{OL} = 60 \text{ mA}$	2.7	0.5	V	
	$V_{OH}$	Output High Voltage	$I_{OH} = -3 \text{ mA}$		V		
	* $C_L$	Capacitive Load			15	pF	
DAT0/-DAT7/	$V_{OL}$	Output Low Voltage	$I_{OL} = 50 \text{ mA}$	2.4	0.6	V	
	$V_{OH}$	Output High Voltage	$I_{OH} = -10 \text{ mA}$		V		
	$V_{IL}$	Input Low Voltage		2.0	0.95	V	
	$V_{IH}$	Input High Voltage			V		
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.45\text{V}$		-0.25	mA	
	$I_{LH}$	Output Leakage High	$V_O = 5.25\text{V}$		100	$\mu\text{A}$	
	$I_{LL}$	Output Leakage Low	$V_O = 0.45\text{V}$		100	$\mu\text{A}$	
	* $C_L$	Capacitive Load			18	pF	
INH1/	$V_{IL}$	Input Low Voltage		2.0	0.8	V	
	$V_{IH}$	Input High Voltage			V		
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.5\text{V}$			-2.0	mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.7\text{V}$			50	$\mu\text{A}$
	* $C_L$	Capacitive Load				18	pF
INIT/ (SYSTEM RESET)	$V_{OL}$	Output Low Voltage	$I_{OL} = 46 \text{ mA}$		0.4	V	
	$V_{OH}$	Output High Voltage	OPEN COLLECTOR				
	$V_{IL}$	Input Low Voltage			0.8	V	

\*Capacitive load values are approximations.



Table 2-14. iSBC 80/30 DC Characteristics (Continued)

Signals	Symbol	Parameter Description	Test Conditions	Min.	Max.	Units
INIT/ (SYSTEM RESET) (Continued)	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.5V$		-2.0	mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.7V$		90	$\mu A$
	* $C_L$	Capacitive Load			15	pF
INT0/-INT7	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.4V$		-0.4	mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.7V$		20	$\mu A$
	* $C_L$	Capacitive Load			15	pF
IORC/, IOWC/	$V_{OL}$	Output Low Voltage	$I_{OL} = 32 \text{ mA}$		0.45	V
	$V_{OH}$	Output High Voltage	$I_{OH} = -2 \text{ mA}$	2.4		V
	$I_{LH}$	Output Leakage High	$V_O = 5.25V$		100	$\mu A$
	$I_{LL}$	Output Leakage Low	$V_O = 0.45V$		-100	$\mu A$
	* $C_L$	Capacitive Load			15	pF
MRDC/, MWTC/	$V_{OL}$	Output Low Voltage	$I_{OL} = 32 \text{ mA}$		0.45	V
	$V_{OH}$	Output High Voltage	$I_{OH} = -2 \text{ mA}$	2.4		V
	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.4V$		-0.5	mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.7V$		150	$\mu A$
	* $C_L$	Capacitive Load			15	pF
XACK/	$V_{OL}$	Output Low Voltage	$I_{OL} = 32 \text{ mA}$		0.4	V
	$V_{OH}$	Output High Voltage	$I_{OH} = -5 \text{ mA}$	2.4		V
	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.4V$		-0.95	mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.4V$		60	$\mu A$
	* $C_L$	Capacitive Load			25	pF
*Capacitive load values are approximations.						

Table 2-15. iSBC 80/30 AC Characteristics (Master Mode)

Parameter	Overall		Read		Write		Description	Remarks
	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)	Min. (ns)	Max. (ns)		
t <sub>AS</sub>	50		50		50		Address setup time to command	
t <sub>AH</sub>	50		50		50		Address hold time from command	
t <sub>DS</sub>	50		50				Data setup to command	
t <sub>DHW</sub>	50				50		Data hold time from command	
t <sub>CY</sub>	358	365					CPU cycle time	
t <sub>CMDR</sub>			670				Read command width	With 1 wait state
t <sub>CMDW</sub>					390		Write command width	With 1 wait state
t <sub>CSWR</sub>	575						Read-to-write command separation	In override mode
t <sub>CSRR</sub>	465						Read-to-read command separation	In override mode
t <sub>CSWW</sub>	575						Write-to-write command separation	In override mode
t <sub>CSRW</sub>	465						Write-to-read command separation	In override mode
t <sub>XACK1</sub>			-195				Read command to XACK 1st sample point	In override mode
t <sub>XACK2</sub>					-509		Write command to XACK 1st sample point	In override mode
t <sub>SAM</sub>	350	375					Time between XACK samples	In override mode
t <sub>ACKRD</sub>			75				AACK to valid read data	When AACK is used
t <sub>ACKWT</sub>					175		AACK to write command inactive	When AACK is used
t <sub>DHR</sub>			0				Read data hold time	
t <sub>DXL</sub>			-75				Read data setup to XACK	
t <sub>XKH</sub>	0		0		0		XACK hold time	
t <sub>BW</sub>	35						Bus clock low or high interval	Supplied by system
t <sub>BS</sub>	23						BPRN to BCLK setup time	
t <sub>DBY</sub>		55					BCLK to BUSY delay	
t <sub>NOD</sub>		30					BPRN to BPRO delay	
t <sub>BCY</sub>	108	109					Bus clock period (BCLK)	From iSBC 80/30 when terminated
t <sub>BW</sub>	35	74					Bus clock low or high interval	From iSBC 80/30 when terminated
t <sub>INIT</sub>	3000						Initialization width	After all voltages have stabilized

Table 2-16. iSBC 80/30 AC Characteristics (Slave Mode)

Parameter	Minimum (ns)	Maximum (ns)	Description	Remarks
t <sub>AS</sub>	50		Address setup to command	From address to command
t <sub>DS</sub>	-200		Write data setup to command	
t <sub>OB1</sub>		1500	On-board memory cycle delay	No refresh
t <sub>ACK</sub>	480	720	Command to XACK	
t <sub>CMD</sub>	720		Command width	
t <sub>AH</sub>	0		Address hold time	
t <sub>DHW</sub>	0		Write data hold time	
t <sub>DHR</sub>	0		Read data hold time	
t <sub>XTH</sub>	0	45	Acknowledge hold time	Acknowledge turnoff delay
*t <sub>ACC</sub>		645	Access time to read data	
t <sub>IH</sub>	50		Inhibit time from command trailing edge	Blocks AACK if t <sub>IS</sub> > t <sub>IS</sub> min.
t <sub>IPW</sub>	100		Inhibit pulse width	
*t <sub>CY</sub>	680	920	Minimum cycle time	t <sub>CY</sub> = t <sub>ACK</sub> + t <sub>SEP</sub>
t <sub>OB2</sub>		1865	On board memory cycle delay	Refresh delaying SACK
t <sub>RD</sub>	535	555	Refresh delay time	
t <sub>DXL</sub>	25		Read data setup to XACK	
t <sub>SEP</sub>	200		Command separation	
t <sub>IS</sub>	-50		Inhibit setup to command	Blocks RAM cycle and t <sub>ACK</sub>

\*When an asynchronous refresh cycle occurs, t<sub>RD</sub> is added to these parameters; when on-board memory cycle occurs, t<sub>OB1</sub> is also added.

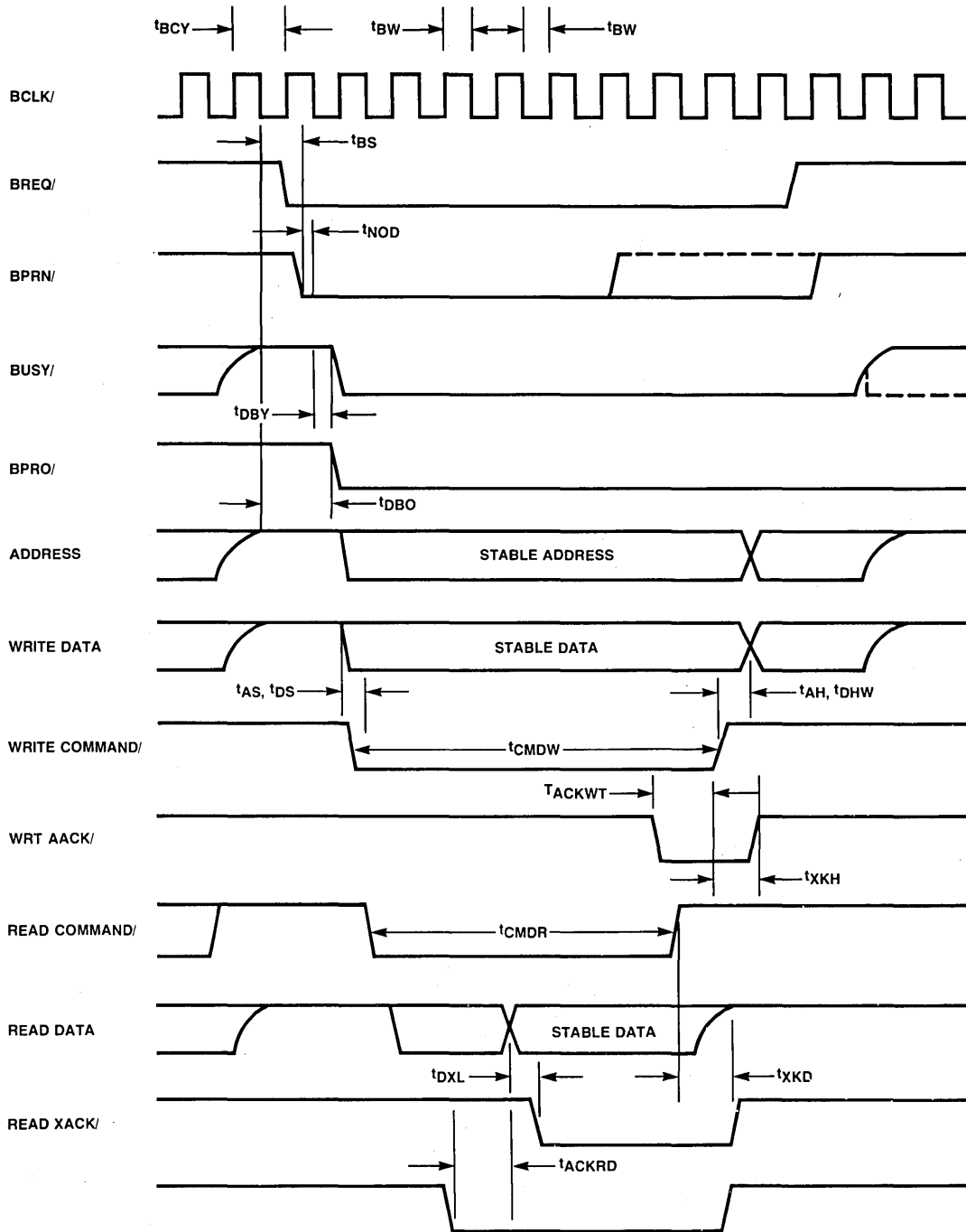


Figure 2-3. Bus Exchange Timing (Master Mode)

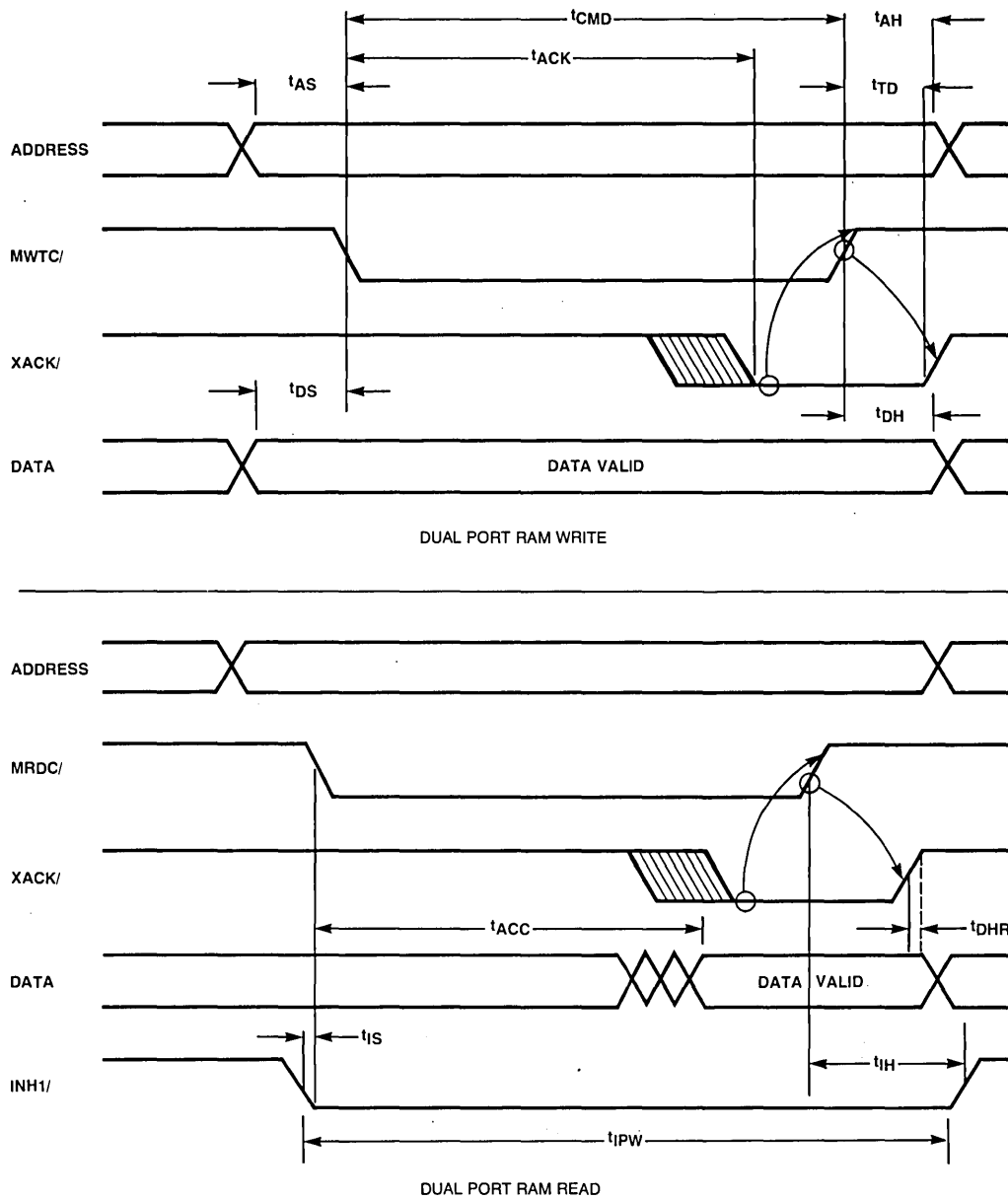


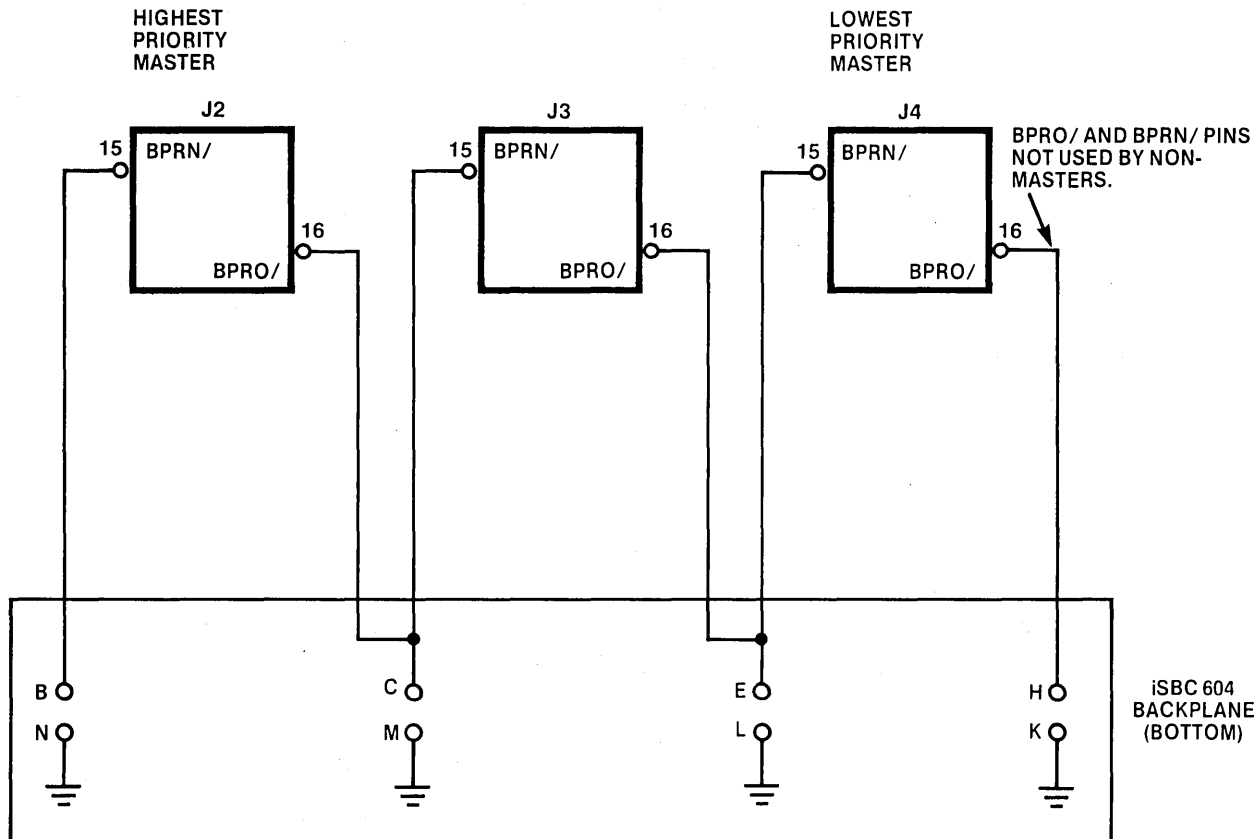
Figure 2-4. Bus Exchange Timing (Slave Mode)

**2-26. SERIAL PRIORITY RESOLUTION**

In a multiple bus master system, bus contention can be resolved in an iSBC 604 Modular Backplane and Cardcage by implementing a serial priority resolution scheme as shown in figure 2-5. Due to the propagation delay of the BPRO/ signal path, this scheme is limited to a maximum of three bus masters capable of acquiring and controlling the Multibus. In the configuration shown in figure 2-5, the bus master installed in slot J2 has the highest priority and is able to acquire control of the

Multibus at any time because its BPRN/input is always enabled (tied to ground) through jumpers B and N on the backplane. (See figure 5-3.)

If the bus master in slot J2 desires control of the Multibus, it drives its BPRO/ output high and inhibits the BPRN/ input to all lower-priority bus masters. When finished using the Multibus, the J2 bus master pulls its BPRO/ output low and gives the J3 bus master the opportunity to take control of the Multibus. If the J3 bus master does not



484-2

Figure 2-5. Serial Priority Resolution Scheme

desire to control the Multibus at this time, it pulls its BPRO/ output low and gives the lowest priority bus master in slot J4 the opportunity to assume control of the Multibus.

The serial priority scheme can be implemented in a user-designed system bus if the chaining of BPRO/ and BPRN/ signals are wired as shown in figure 5-3.

## 2-27. PARALLEL PRIORITY RESOLUTION

A parallel priority resolution scheme allows up to 16 bus masters to acquire and control the Multibus. Figure 2-6 illustrates one method of implementing such a scheme for resolving bus contention in a system containing eight bus masters installed in an iSBC 604/614. Notice that the two highest and two lowest priority bus masters are shown installed in the iSBC 604.

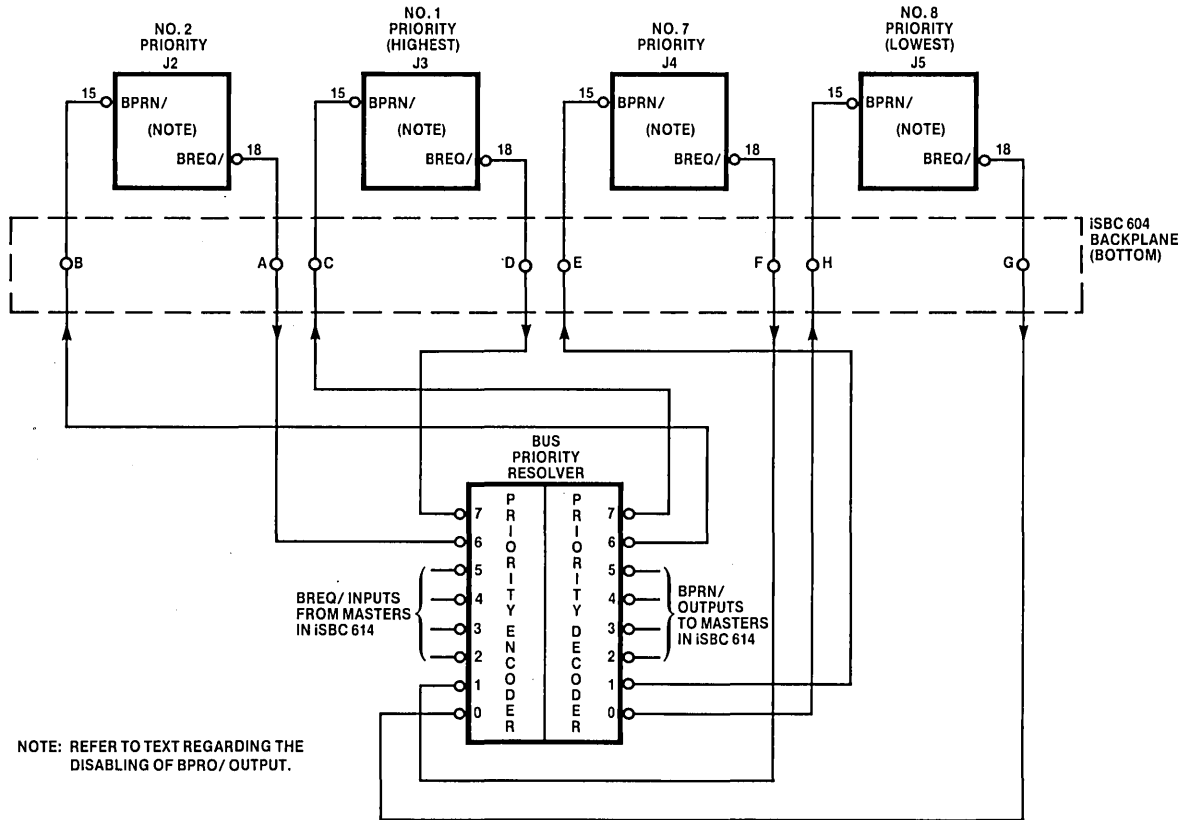
In the scheme shown in figure 2-6, the priority encoder is a Texas Instruments 74148 and the priority decoder is an Intel 8205. Input connections to the priority encoder

determine the bus priority, with input 7 having the highest priority and input 0 having the lowest priority. Here, the J3 bus master has the highest priority and the J5 bus master has the lowest priority.

**IMPORTANT:** In a parallel priority resolution scheme, the BPRO/ output must be disabled on all bus masters. On the iSBC 80/30, disable the BPRO/ output signal by removing jumper 169-170. If a similar jumper cannot be removed on the other bus masters, either clip the IC pin that supplies the BPRO/ output signal to the Multibus or cut the signal trace.

## 2-28. POWER FAIL/MEMORY PROTECT CONFIGURATION

A mating connector must be installed in the iSBC 604/604 Modular Cardage and Backplane to accommodate auxiliary connector P2. (Refer to figure 1-1.) Table 2-2 lists



484-1

Figure 2-6. Parallel Priority Resolution Scheme

some 60-pin connectors that can be used for this purpose; flat crimp, solder, and wirewrap connector types are listed. Table 2-17 correlates the signals and pin numbers on the connector.

Procure the appropriate mating connector for P2 and secure it in place as follows:

- Position holes in P2 mating connector over mounting holes that are in line with corresponding P1 mating connector.
  - From top of connector, insert two 0.5-inch #4-40 pan head screws down through connector and mounting holes.
  - Install a flat washer, lock washer, and star-type nut on each screw; then tighten the nuts.
- When the mating connector for P2 is in place, wire the power fail signals to the appropriate pins of the connector as listed in table 2-17. (The dc characteristics of the auxiliary signals are given in table 2-18.) In a typical system, these signals would be wired as follows:
- Connect auxiliary signal common and returns for +5V, -5V, and +12V backup batteries to P2 pins 1 and 2.
  - Connect +5V battery input to P2 pins 3 and 4; -5V battery input to P2 pins 7 and 8; and +12V battery input to P2 pins 11 and 12. Remove jumpers W7, W8, and W9.
  - Connect PFS/input to P2 pin 17 and MEM PROT/ input to P2 pin 20.
  - Connect PFI/ input to P2 pin 19; this signal is inverted and applied to the priority interrupt matrix. To assign the PFI/ input as the highest priority interrupt (8085A TRAP), remove jumper 137-145 and connect jumper 134-137.
  - Connect HLT/ output at P2 pin 28 to external HALT indicator, which is typically a light-emitting diode (LED) mounted on the system enclosure.
  - Connect BTMO output at P2 pin 34 to external TIME OUT indicator, which is typically a LED mounted on the system enclosure.
  - Connect AUX RESET/ input to P2 pin 38. This signal is usually supplied by a momentary closure switch mounted on the system enclosure.
  - Connect WAIT/ output at P2 pin 32 to external WAIT indicator, which is typically a LED mounted on the system enclosure.

Table 2-17. Auxiliary Connector P2 Pin Assignments

Pin*	Signal	Definition
1	GND	} Auxiliary common
2	GND	
3	+5V AUX	} Auxiliary backup battery supply
4	+5V AUX	
7	-5V AUX	
8	-5V AUX	
11	+12V AUX	
12	+12V AUX	
17	PFS/	Power Fail Status. This externally supplied signal is applied to the SID input of the 8085A microprocessor to allow the program to check the current status of the system power fail circuit. The status is checked by periodically executing a RIM instruction.
19	PFI/	Power Fail Interrupt. This externally supplied signal is applied to the priority interrupt matrix. This signal should normally be jumpered to the 8085A microprocessor TRAP input.
20	MEM PROT/	Memory Protect. This externally supplied signal prevents access to RAM during battery backup operation.
28	HLT/	Halt. This output signal indicates that the 8085A microprocessor is halted.
32	WAIT/	Wait. This output signal, which is the same as the CPU ALE signal, indicates that the 8085A microprocessor is either halted or in a wait state.
34	BTMO	Bus Time Out. This output signal indicates that the 8085A microprocessor has either halted or is hung up in a wait state (i.e., waiting for an acknowledge signal in response to the previous I/O or Memory Command).
38	AUX RESET/	Auxiliary Reset. This externally supplied signal initiates a pseudo power-up sequence; i.e., initializes the board and resets the entire system to a known internal state.

\*All odd-numbered pins (1, 3, 5 . . . 59) are on component side of the board. Pin 1 is the left-most pin when viewed from the component side of the board with the extractors at the top.

Table 2-18. Auxiliary Signal (Connector P2) DC Characteristics

Signals	Symbol	Parameter Description	Test Conditions	Min.	Max.	Units	
PFS/, MEM PROT/	$V_{IL}$	Input Low Voltage	$V_{IN} = 0.4V$ $V_{IN} = 2.4V$	2.4	0.8	V	
	$V_{IH}$	Input High Voltage				V	
	$I_{IL}$	Input Current at Low V				2	mA
	$I_{IH}$	Input Current at High V				50	$\mu$
	$C_L$	Capacitive Load				10	pF
AUX RESET/	$V_{IL}$	Input Low Voltage	$V_{IN} = 0.45V$ $V_{IN} = 5.25V$	2.6	0.8	V	
	$V_{IH}$	Input High Voltage				V	
	$I_{IL}$	Input Current at Low V				-0.25	mA
	$I_{IH}$	Input Current at High V				10	$\mu$ A
	$C_L$	Capacitive Load				10	$\mu$ F
WAIT/	$V_{OL}$	Output Low Voltage	$I_{OL} = 17\text{ mA}$ $I_{OH} = -950\ \mu\text{A}$	2.7	0.5	V	
	$V_{OH}$	Output High Voltage				V	
	$C_L$	Capacitive Load				15	pF
PFI/	$V_{IL}$	Input Low Voltage	$V_{IN} = 0.40V$ $V_{IN} = 2.7V$	2.0	0.80	V	
	$V_{IH}$	Input High Voltage				V	
	$I_{IL}$	Input Current at Low V				-0.4	mA
	$I_{IH}$	Input Current at High V				20	$\mu$ A
	$C_L$	Capacitive Load				15	pF

## 2-29. PARALLEL I/O CABLING

Parallel I/O ports E8, E9, and EA are controlled by the Intel 8255A Programmable Peripheral Interface (PPI) and interfaced via edge connector J1. Parallel I/O ports 1 and 2 are controlled by the optional Intel 8041/8741 Universal Peripheral Interface and interfaced via edge connector J2. (Refer to figure 1-1.) Pin assignments for J1 and J2 are listed in tables 2-19 and 2-20, respectively; dc characteristics of the parallel I/O signals are given in table 2-21. Table 2-2 lists some 50-pin edge connectors that can be used for interface to J1 and J2; flat crimp, solder, and wirewrap connector types are listed.

The transmission path from the I/O source to the iSBC 80/30 should be limited to 3 meters (10 feet) maximum.

The following bulk cable types (or equivalent) are recommended for interfacing with the parallel I/O ports:

- a. Cable, flat, 50-conductor, 3M 3306-50.
- b. Cable, flat, 50-conductor (with ground plane), 3M 3380-50.
- c. Cable, woven, 25-pair, 3M 3321-25.

An Intel iSBC 956 Cable Set, consisting of two cable assemblies, is recommended for parallel I/O interfacing.

**Table 2-19. Connector J1 Pin Assignments**

Pin*	Function	Pin*	Function	
1	Ground	2	Port E9 bit 7	
3		4	Port E9 bit 6	
5		6	Port E9 bit 5	
7		8	Port E9 bit 4	
9		10	Port E9 bit 3	
11		12	Port E9 bit 2	
13		14	Port E9 bit 1	
15		16	Port E9 bit 0	
17		Ground	18	Port EA bit 3
19			20	Port EA bit 2
21	22		Port EA bit 1	
23	24		Port EA bit 0	
25	26		Port EA bit 4	
27	28		Port EA bit 5	
29	30		Port EA bit 6	
31	32		Port EA bit 7	
33	Ground		34	Port E8 bit 7
35			36	Port E8 bit 6
37		38	Port E8 bit 5	
39		40	Port E8 bit 4	
41		42	Port E8 bit 3	
43		44	Port E8 bit 2	
45		46	Port E8 bit 1	
47		48	Port E8 bit 0	
49		Ground	50	EXT INTR0/

\*All odd-numbered pins (1, 3, 5, ... 49) are on component side of the board. Pin 1 is the right-most pin when viewed from the component side of the board with the extractors at the top.

Both cable assemblies consist of a 50-conductor flat cable with a 50-pin PC connector at one end. When attaching the cable to J1 or J2, be sure that the connector is oriented properly with respect to pin 1 on the edge connector. (Refer to the footnotes in tables 2-19 and 2-20.)

## 2-30. SERIAL I/O CABLING

Pin assignments and signal definitions for RS232C serial I/O interface are listed in table 2-10. An Intel iSBC 955 Cable Set is recommended for RS232C interfacing. One cable assembly consists of a 25-conductor flat cable with a 26-pin PC connector at one end and an RS232C interface connector at the other end. The second cable assembly includes an RS232C connector at one end and has spade lugs at the other end; the spade lugs are used to interface to a teletypewriter. (See Appendix B for ASR-33 TTY interface instructions.)

For OEM applications where cables will be made for the iSBC 80/30, it is important to note that the mating connector for J3 has 26 pins whereas the RS232C connector has 25 pins. Consequently, when connecting the 26-pin mating connector to 25-conductor flat cable, be sure that the

**Table 2-20. Connector J2 Pin Assignments**

Pin*	Function	Pin*	Function	
1	Ground	2	Port 2 bit 7	
3		4	Port 2 bit 6	
5		6	Port 2 bit 5	
7		8	Port 2 bit 4**	
9		10	Port 2 bit 3**	
11		12	Port 2 bit 2**	
13		14	Port 2 bit 1**	
15		16	Port 2 bit 0**	
17		Ground	18	T0 Input
19			20	T1 Input
21	22		Not Used	
23	24		Not Used	
25	26		Not Used	
27	28		Not Used	
29	30		41RS232 Out* *	
31	32		41RS232 In* *	
33	Ground		34	Port 1 bit 7
35			36	Port 1 bit 6
37		38	Port 1 bit 5	
39		40	Port 1 bit 4	
41		42	Port 1 bit 3	
43		44	Port 1 bit 2	
45		46	Port 1 bit 1	
47		48	Port 1 bit 0	
49		Ground	50	EXT INTR1/

\*All odd-numbered pins (1, 3, 5, ... 49) are on component side of the board. Pin 1 is the right-most pin when viewed from the component side of the board with the extractors at the top.

\*\*Jumpered pin or function. Refer to figure 5-2 sheet 5.



Table 2-21. Parallel I/O Signal (Connectors J1/J2) DC Characteristics

Signals	Symbol	Parameter Description	Test Conditions	Min.	Max.	Units
Port E8 Bidirectional Drivers	$V_{OL}$	Output Low Voltage	$I_{OL} = 20 \text{ mA}$		0.45	V
	$V_{OH}$	Output High Voltage	$I_{OH} = -12 \text{ mA}$	2.4		V
	$V_{IL}$	Input Low Voltage			0.95	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.45\text{V}$		-5.25	mA
	$C_L$	Capacitive Load			18	pF
8255A Driver/Receiver	$V_{OL}$	Output Low Voltage	$I_{OL} = 1.7 \text{ mA}$		0.45	V
	$V_{OH}$	Output High Voltage	$I_{OH} = -200 \mu\text{A}$	2.4		V
	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.45$		10	$\mu\text{A}$
	$I_{IH}$	Input Current at High V	$V_{IN} = 5.0$		10	$\mu\text{A}$
	$C_L$	Capacitive Load			18	pF
8041/8741A Driver/Receiver	$V_{OL}$	Output Low Voltage	$I_{OL} = 1.6 \text{ mA}$		0.45	V
	$V_{OH}$	Output High Voltage	$I_{OH} = 50 \mu\text{A}$	2.4		V
	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.8\text{V}$		0.4	$\mu\text{A}$
	$I_{IH}$	Input Current at High V	$V_{IN} = 5.25\text{V}$		10	$\mu\text{A}$
	$C_L$	Capacitive Load			18	pF
EXT INT0	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.4\text{V}$		-5.5	3 mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.7\text{V}$		20	3 $\mu\text{A}$
	$C_L$	Capacitive Load			15	pF
EXT INT1	$V_{IL}$	Input Low Voltage			0.8	V
	$V_{IH}$	Input High Voltage		2.0		V
	$I_{IL}$	Input Current at Low V	$V_{IN} = 0.4\text{V}$		-0.4	mA
	$I_{IH}$	Input Current at High V	$V_{IN} = 2.7\text{V}$		20	$\mu\text{A}$
	$C_L$	Capacitive Load			15	pF

cable makes contact with pins 1 and 2 of the mating connector and not with pin 26. Table 2-22 provides pin correspondence between connector J3 and an RS232C connector. When attaching the cable to J3, be sure that the PC connector is oriented properly with respect to pin 1 on the edge connector. (Refer to the footnote in table 2-10.)

### 2-31. BOARD INSTALLATION



**Table 2-22. Connector J3 Vs RS232C Pin Correspondence**

PC Conn. J3	RS232C Conn.	PC Conn. J3	RS232C Conn.
1	14	14	7
2	1	15	21
3	15	16	8
4	2	17	22
5	16	18	9
6	3	19	23
7	17	20	10
8	4	21	24
9	18	22	11
10	5	23	25
11	19	24	12
12	6	25	N/C
13	20	26	13

Always turn off the computer system power supply before installing or removing the iSBC 80/30 board and before installing or removing device interface cables. Failure to take these precautions can result in damage to the board.

In an iSBC 80 Single Board Computer based system, install the iSBC 80/30 in any slot that has not been wired for a dedicated function. In an Intellec System, install the iSBC 80/30 in any slot except slot 1 or 2. Make sure that auxiliary connector P2 (if used) mates with the user-installed mating connector. Attach the appropriate cable assemblies to connectors J1 through J3.



# CHAPTER 3 PROGRAMMING INFORMATION

## 3-1. INTRODUCTION

This chapter lists the on-board memory and I/O address assignments, describes the effects of a system initialize command, and provides programming information for the following programmable chips:

- a. Intel 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter) that controls the serial I/O port.
- b. Intel 8253 PIT (Programmable Interval Timer) that controls various frequency and timing functions.
- c. Intel 8255A PPI (Programmable Peripheral Interface) that controls the three parallel I/O ports.
- d. Intel 8259A PIC (Programmable Interrupt Controller) that can handle up to eight vectored priority interrupts for the on-board 8085A microprocessor (CPU).
- e. Intel 8041/8741 UPI (Universal Peripheral Interface).

This chapter also discusses the Intel 8085A Microprocessor interrupt capability. The instruction set for the 8085A is included in Appendix A; a complete description of programming with Intel's assembly language is given in the *8080/8085 Assembly Language Programming Manual*, Order No. 9800310.

This chapter does not provide assembly language programming information for the optional Intel 8041/8741 UPI (Universal Peripheral Interface). This information is available in the *UPI-41 User's Manual*, Order No. 9800504A.

## 3-2. FAILSAFE TIMER

The 8085A microprocessor (CPU) expects an acknowledge signal to be returned from the addressed I/O or memory device in response to each Read or Write Command. The iSBC 80/30 includes a Failsafe Timer that is triggered during T<sub>1</sub> of every machine cycle. If the Failsafe Timer is enabled by hardware jumper as described in table 2-4, and an acknowledge signal is not received within 10 milliseconds, the Failsafe Timer will time out and allow the CPU to exit the wait state. As described in Chapter 2, provision is made so that the Failsafe Timer output (BUS TIME OUT) can optionally be used to interrupt the CPU and/or to drive a front panel indicator.

If the Failsafe Timer is not enabled, and an acknowledge signal is not returned for any reason, the CPU will hang up

in a wait state. In this situation, the only way to free the CPU is to initialize the system as described in paragraph 3-5.

## 3-3. MEMORY ADDRESSING

The iSBC 80/30 includes 16K of dynamic RAM and two IC sockets to accommodate up to 8K of user-installed ROM/PROM. The iSBC 80/30 features a two-port RAM access arrangement in which the on-board RAM can be accessed by the on-board 8085A microprocessor (CPU) or by another bus master board via the Multibus. The ROM/PROM can be accessed only by the CPU.

The on-board RAM can be accessed by another bus master that currently has control of the Multibus. It should be noted, however, that even though another bus master may be continuously accessing the iSBC 80/30 on-board RAM, this does not lock out the CPU from accessing the on-board RAM. In this situation, Memory Commands from the CPU and the controlling bus master are interleaved. This, of course, will impose CPU wait states until the current access by the controlling bus master is completed.

Addresses for CPU access of ROM/PROM and on-board RAM are provided in table 3-1. Note that the ROM/PROM address space depends on the user's configura-

**Table 3-1. On-Board Memory Addresses  
(For CPU Access)**

On-Board Memory	Configuration	Legal Address	Illegal Address
ROM/PROM	One 1K × 8 chip Two 1K × 8 chips	0000-03FF 0000-07FF	0400-07FF —
ROM/PROM	One 2K × 8 chip Two 2K × 8 chips	0000-07FF 0000-0FFF	0800-0FFF —
ROM/PROM	One 4K × 8 chip Two 4K × 8 chips	0000-0FFF 0000-1FFF	1000-1FFF —
RAM	8K Access	2000-3FFF *4000-5FFF 6000-7FFF 8000-9FFF A000-BFFF C000-DFFF E000-FFFF	None None None None None None None
RAM	*16K Access	*4000-7FFF 8000-BFFF C000-FFFF	None None None

\*Default (factory connected) jumper; refer to paragraph 2-16. Addressing RAM outside the jumper-selected block results in an off-board request via the Multibus.

tion, and that the RAM address space depends on whether the board jumpers are configured to allow the CPU to access 8K or 16K of on-board RAM.

For Multibus access, the on-board RAM may be mapped into any 8K or 16K segment within the addressing constraints of the controlling bus master. In other words, for 16-bit Multibus addressing, the RAM may be mapped into any 8K or 16K segment of the 64K byte address space. For 20-bit Multibus addressing, the RAM may be mapped into any 8K or 16K segment of the 1-megabyte address space. Additional information is provided in paragraphs 2-17 through 2-19.

When the CPU is addressing *on-board* memory (ROM/PROM or RAM), an internal PROM or RAM Advanced Acknowledge (AACK/) is automatically generated to prevent imposing a CPU wait state. When the CPU is addressing *system* memory via the Multibus, the CPU must first gain control of the Multibus and, after the Memory Read or Memory Write Command is given, must wait for a Transfer Acknowledge (XACK/) to be received from the addressed memory device. The Failsafe Timer, if enabled, will prevent a CPU hang-up in the event of a memory device equipment failure or a bus failure.

It should be noted in table 3-1 that it is possible to configure ROM/PROM such as to create *illegal* addresses. If an illegal address is used in conjunction with a Memory Write Command to ROM/PROM, a PROM AACK/ signal is generated as though the address was legal and the CPU will continue executing the program. However, in this case, erroneous data will be returned.

### 3-4. I/O ADDRESSING

The on-board 8085A microprocessor (CPU) communicates with the programmable chips through a sequence of I/O Read and I/O Write Commands. As shown in table 3-2, each of these chips recognizes four separate hexadecimal I/O addresses that are used to control the various programmable functions. Where two hexadecimal addresses are listed for a single function, either address may be used. For example, an I/O Read Command to ED or EF will read the status of the 8251A USART.

#### NOTE

The on-board I/O functions are not accessible to another bus master via the Multibus.

### 3-5. SYSTEM INITIALIZATION

When power is initially applied to the system, an Initialize (INIT/) signal is automatically generated that clears the internal Program Counter, Instruction Register, and Interrupt Enable flip-flop and “resets” the 8251A USART, 8255A PPI, and optional 8041/8741A UPI as follows:

Table 3-2. I/O Address Assignments

I/O Address	Chip Select	Function
D8 or DA	8259A PIC	Write: ICW1, OCW2, and OCW3 Read: Status and Poll
D9 or DB		Write: ICW2 and OCW1 (Mask) Read: OCW1 (Mask)
DC	8253 PIT	Write: Counter 0 (Load Count ÷ N) Read: Counter 0
DD		Write: Counter 1 (Load Count ÷ N) Read: Counter 1
DE		Write: Counter 2 (Load Count ÷ N) Read: Counter 2
DF		Write: Control Read: None
E4 or E6	8041/8741A UPI	Write: Data (J2) Read: Data (J2)
E5 or E7		Write: Command Read: Status
E8	8255A PPI	Write: Port A (J1) Read: Port A (J1)
E9		Write: Port B (J1) Read: Port B (J1)
EA		Write: Port C (J1) Read: Port C (J1)
EB		Write: Control Read: None
EC or EE	8251A USART	Write: Data (J3) Read: Data (J3)
ED or EF		Write: Mode or Command Read: Status

- a. The 8251A USART is set to an “idle” mode, waiting for a set of Command Words to program the desired function.
- b. All three ports of the 8255A PPI are set to the input mode.
- c. The 8041/8741A UPI internal Program Counter and Status flip-flops are cleared.

The 8253 PIT and 8259A PIC are not affected by the power-up sequence.

The INIT/ signal is also gated onto the Multibus to set the remainder of the system components to a known internal state.

The INIT/ signal can also be generated by an auxiliary RESET switch. Pressing and releasing the RESET switch produces the same effect as the INIT/ signal described above.

### 3-6. 8251A USART PROGRAMMING

The USART converts parallel output data into virtually any serial output data format (including IBM Bi-Sync) for half- or full-duplex operation. The USART also converts serial input data into parallel data format.

Prior to starting transmitting or receiving data, the USART must be loaded with a set of control words. These control words, which define the complete functional operation of the USART, must immediately follow a reset (internal or external). The control words are either a Mode instruction or a Command instruction.

### 3-7. MODE INSTRUCTION FORMAT

The Mode instruction word defines the general characteristics of the USART and must follow a reset operation. Once the Mode instruction word has been written into the USART, sync characters or command instructions may be inserted. The Mode instruction word defines the following:

- a. For Sync Mode:
  - (1) Character length
  - (2) Parity enable
  - (3) Even/odd parity generation and check
  - (4) External sync detect (not supported by iSBC 80/30)
  - (5) Single or double character sync
- b. For Async Mode:
  - (1) Baud rate factor (X1, X16, or X64)
  - (2) Character length
  - (3) Parity enable
  - (4) Even/odd parity generation and check
  - (5) Number of stop bits

Instruction word and data transmission formats for synchronous and asynchronous modes are shown in figures 3-1 through 3-4.

### 3-8. SYNC CHARACTERS

Sync characters are written to the USART in the synchronous mode only. The USART can be programmed for either one or two sync characters; the format of the sync characters is at the option of the programmer.

### 3-9. COMMAND INSTRUCTION FORMAT

The Command instruction word shown in figure 3-5 controls the operation of the addressed USART. A Command

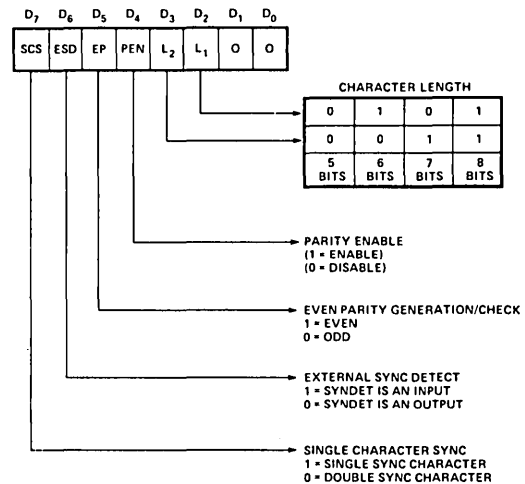


Figure 3-1. USART Synchronous Mode Instruction Word Format

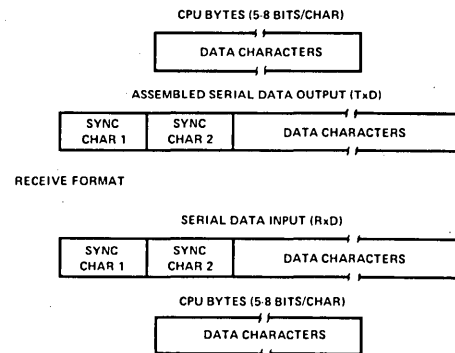


Figure 3-2. USART Synchronous Mode Transmission Format

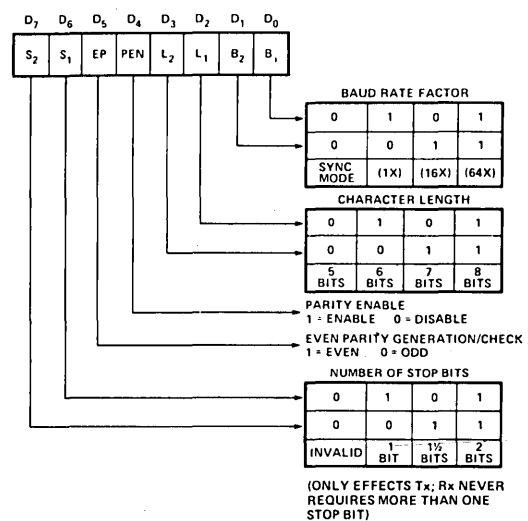


Figure 3-3. USART Asynchronous Mode Instruction Word Format

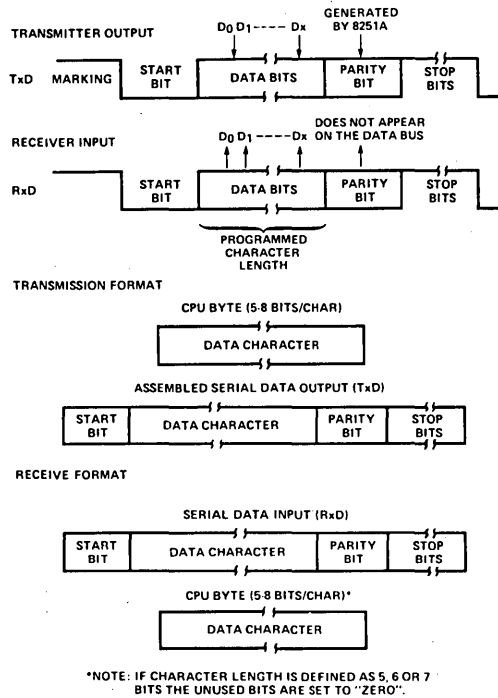


Figure 3-4. USART Asynchronous Mode Transmission Format

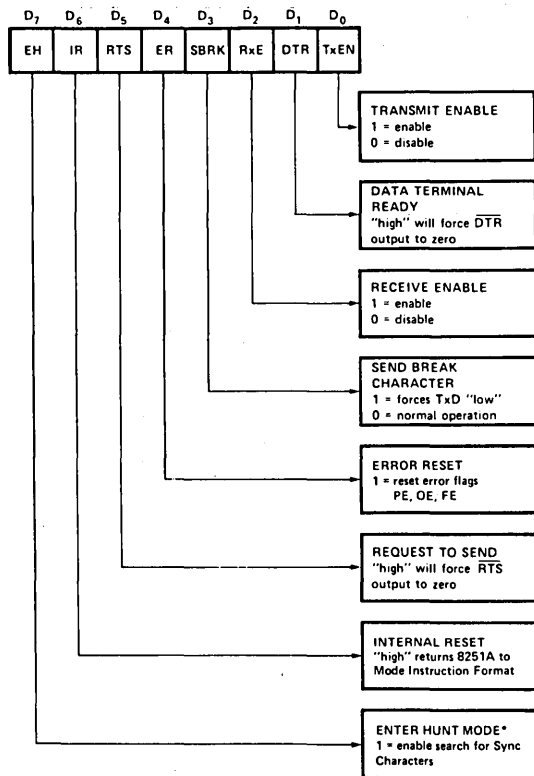


Figure 3-5. USART Command Instruction Word Format

instruction must follow the mode and/or sync words and, once the Command instruction is written, data can be transmitted or received by the USART.

It is not necessary for a Command instruction to precede all data transactions; only those transmissions that require a change in the Command instruction. An example is a change in the enable transmit bit or enable receive bit. Command instructions can be written to the USART at any time after one or more data operations.

After initialization, always read the chip status and check for the TXRDY bit prior to writing either data or command words to the USART. This ensures that any prior input is not overwritten and lost. Note that issuing a Command instruction with bit 6 (IR) set will return the USART to the Mode instruction format.

### 3-10. RESET

To change the Mode instruction word, the USART must receive a Reset command. The next word written to the USART after a Reset command is assumed to be a Mode instruction. Similarly, for sync mode, the next word after a Mode instruction is assumed to be one or more sync characters. All control words written into the USART after the Mode instruction (and/or the sync character) are assumed to be Command instructions.

### 3-11. ADDRESSING

The USART chip uses two consecutive pairs of addresses. The lower of the two addresses in each pair is used to read and write I/O data; the upper address in each pair is used to write mode and command words and to read the USART status. (Refer to table 3-2.)

### 3-12. INITIALIZATION

A typical USART initialization and I/O data sequence is presented in figure 3-6. The USART chip is initialized in four steps:

- a. Reset USART to Mode instruction format.
- b. Write Mode instruction word. One function of mode word is to specify synchronous or asynchronous operation.
- c. If synchronous mode is selected, write one or two sync characters as required.
- d. Write Command instruction word.

To avoid spurious interrupts during USART initialization, disable the USART interrupt. This can be done by either masking the appropriate interrupt request input at the 8259A PIC or by disabling the CPU interrupts by executing a DI instruction.

First, reset the USART chip by writing a Command instruction to location ED (or EF). The Command instruction must have bit 6 set (IR6 = 1); all other bits are immaterial.

**NOTE**

This reset procedure should be used only if the USART has been completely initialized, or if the initialization procedure has reached the point that the USART is ready to receive a Command word. For example, if the reset command is written when the initialization sequence calls for a sync character, then subsequent programming will be in error.

Next, write a Mode instruction word to the USART. (See figures 3-1 through 3-4.) A typical subroutine for writing both Mode and Command instructions is given in table 3-3.

If the USART is programmed for the synchronous mode, write one or two sync characters depending on the transmission format.

Finally, write a Command instruction word to the USART. Refer to figure 3-5 and table 3-3.

**3-13. OPERATION**

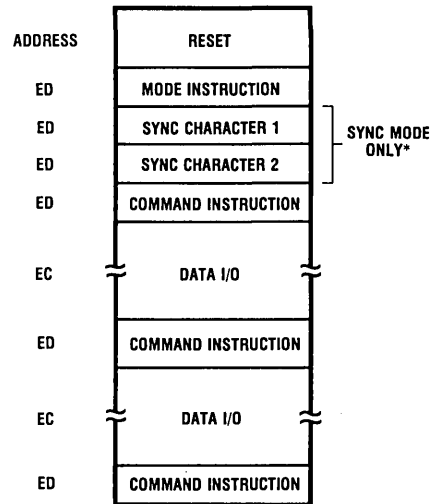
Normal operating procedures use data I/O read and write, status read, and Command instruction write operations. Programming and addressing procedures for the above are summarized in following paragraphs.

**NOTE**

After the USART has been initialized, always check the status of the TXRDY bit *prior* to writing data or writing a new command word to the USART. The TXRDY bit *must* be *true* to prevent overwriting and subsequent loss of command or data words. The TXRDY bit is

inactive until initialization has been completed; therefore, do not check TXRDY until after the command word, which concludes the initialization procedure, has been written.

Prior to any operating change, a new command word must be written with command bits changed as appropriate. (Refer to figure 3-5 and table 3-3.)



\*The second sync character is skipped if Mode instruction has programmed USART to single character internal sync mode. Both sync characters are skipped if Mode instruction has programmed USART to async mode.

**Figure 3-6. Typical USART Initialization and Data I/O Sequence**

**3-14. DATA INPUT/OUTPUT.** For data receive or transmit operations perform a read or write, respectively, to the USART. Tables 3-4 and 3-5 provide examples of typical character read and write subroutines.

During normal transmit operation, the USART generates a Transmit Ready (TXRDY) signal that indicates that the

**Table 3-3. Typical USART Mode or Command Instruction Subroutine**

```

;CMD2 OUTPUTS CONTROL WORD TO USART.
;USES-A, STAT; DESTROYS-NOTHING.

      EXTRN  STAT

CMD2:  PUSH  PSW
      CALL  STAT
      ANI   1           ;CHECK TXRDY
      JZ   LP          ;TXRDY MUST BE TRUE
      POP  PSW
      OUT  0ED         ;ENTER HERE FOR INITIALIZATION
      RET

      END
    
```

Table 3-4. Typical USART Data Character Read Subroutine

```

;RX1 READS DATA CHARACTER FROM USART.
;USES-STAT; DESTROYS-A,FLAGS.

      EXTRN  STAT

RX1:  CALL  STAT
      ANI   2           ;CHECK FOR RXRDY
      JZ    RX1        ;ENTER HERE IF RXRDY IS TRUE
      IN    0EC
      RET

      END

```

Table 3-5. Typical USART Data Character Write Subroutine

```

;TX1 WRITES DATA CHARACTER FROM REG A TO USART.
;USES-STAT; DESTROYS-A,FLAGS.

      EXTRN  STAT

TX1:  PUSH  PSW         ;SAVE DATA
TX11: CALL  STAT
      ANI   1           ;CHECK FOR TXRDY
      JZ    TX11
      POP  PSW
      OUT  0EC        ;ENTER HERE IF TXRDY IS TRUE
      RET

      END

```

USART is ready to accept a data character for transmission. TXRDY is automatically reset when the CPU loads a character into the USART.

Similarly, during normal receive operation, the USART generates a Receive Ready (RXRDY) signal that indicates that a character has been received and is ready for input to the CPU. RXRDY is automatically reset when a character is read by the CPU.

The TXRDY and RXRDY outputs of the USART are available at the priority interrupt jumper matrix. If, for instance, TXRDY and RXRDY are input to the 8259A PIC, the PIC resolves the priority and drives the INTR input high to the CPU. TXRDY and RXRDY are also available in the status word. (Refer to paragraph 3-15.)

**3-15. STATUS READ.** The CPU can determine the status of the serial I/O port by issuing an I/O Read Command to the upper address (ED or EF) of the USART chip. The format of the status word is shown in figure 3-7. A typical status read subroutine is given in table 3-6.

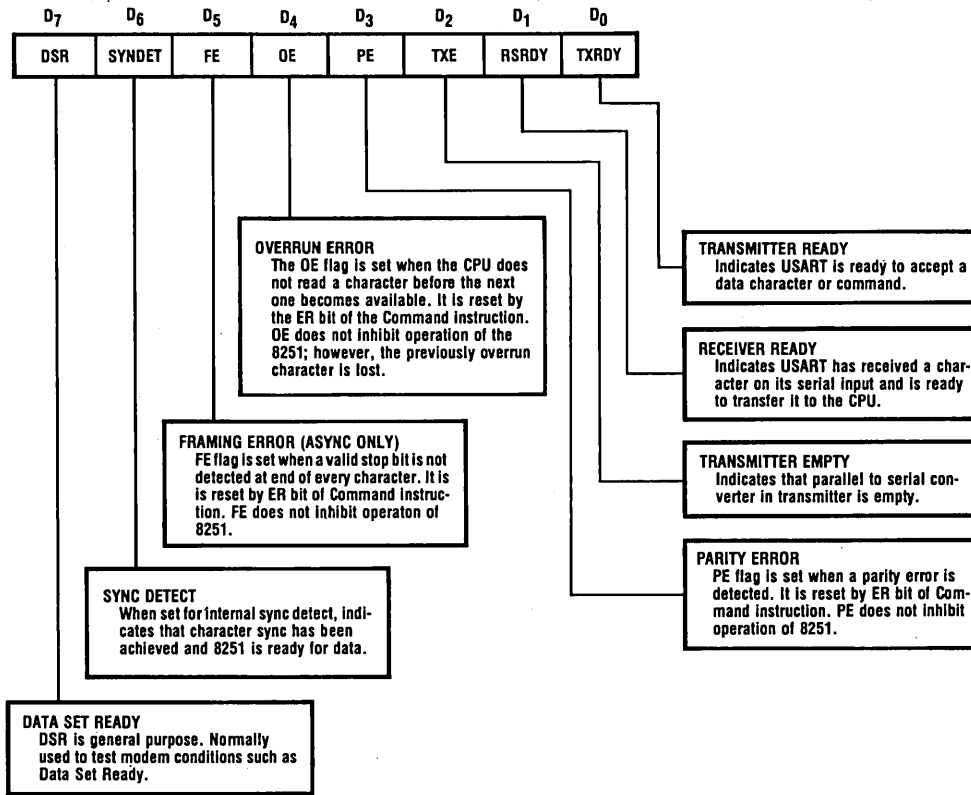
### 3-16. 8253 PIT PROGRAMMING

A 22.1184-MHz crystal oscillator supplies the basic clock frequency for the programmable chips. This clock frequency is divided by 9, 18, and 144 to produce three jumper-selectable clocks: 2.4576 MHz, 1.2288 MHz, and 153.6 kHz. These clocks are available for input to Counter 0, Counter 1, and Counter 2 of the 8253 PIT. The default (factory connected) and optional jumpers for selecting the clock inputs to the three counters are listed in table 2-4.

Default jumpers connect the output of Counter 2 to the TXC and RXC inputs of the 8251A USART. Jumpers are included so that Counters 0 and 1 can provide real-time interrupts or provide an event clock to the 8041/8741 UPI and a count out clock to 8255A PPI Port EA I/O driver.

Before programming the 8253 PIT, ascertain the input clock frequency and the output function of each of the three counters. These factors are determined and established by the user during the installation.





450-14

Figure 3-7. USART Status Read Format

Table 3-6. Typical USART Status Read Subroutine

```

;STAT READS STATUS FROM USART.
;USES-NOTHING; DESTROYS-A.

;STAT    IN    OED    ;GET STATUS
        IN    RET
        END
    
```

### 3-17. MODE CONTROL WORD AND COUNT

All three counters must be initialized separately prior to their use. The initialization for each counter consists of two steps:

- A mode control word (figure 3-8) is written to the control register for each individual counter.
- A down-count number is loaded into each counter; number is in one or two 8-bit bytes as determined by mode control word.

The mode control word (figure 3-8) does the following:

- Selects counter to be loaded.
- Selects counter operating mode.
- Selects one of the following four counter read/load functions:
  - Counter latch (for stable read operation).
  - Read or load most-significant byte only.
  - Read or load least-significant byte only.
  - Read or load least-significant byte first, then most-significant byte.
- Sets counter for either binary or BCD count.

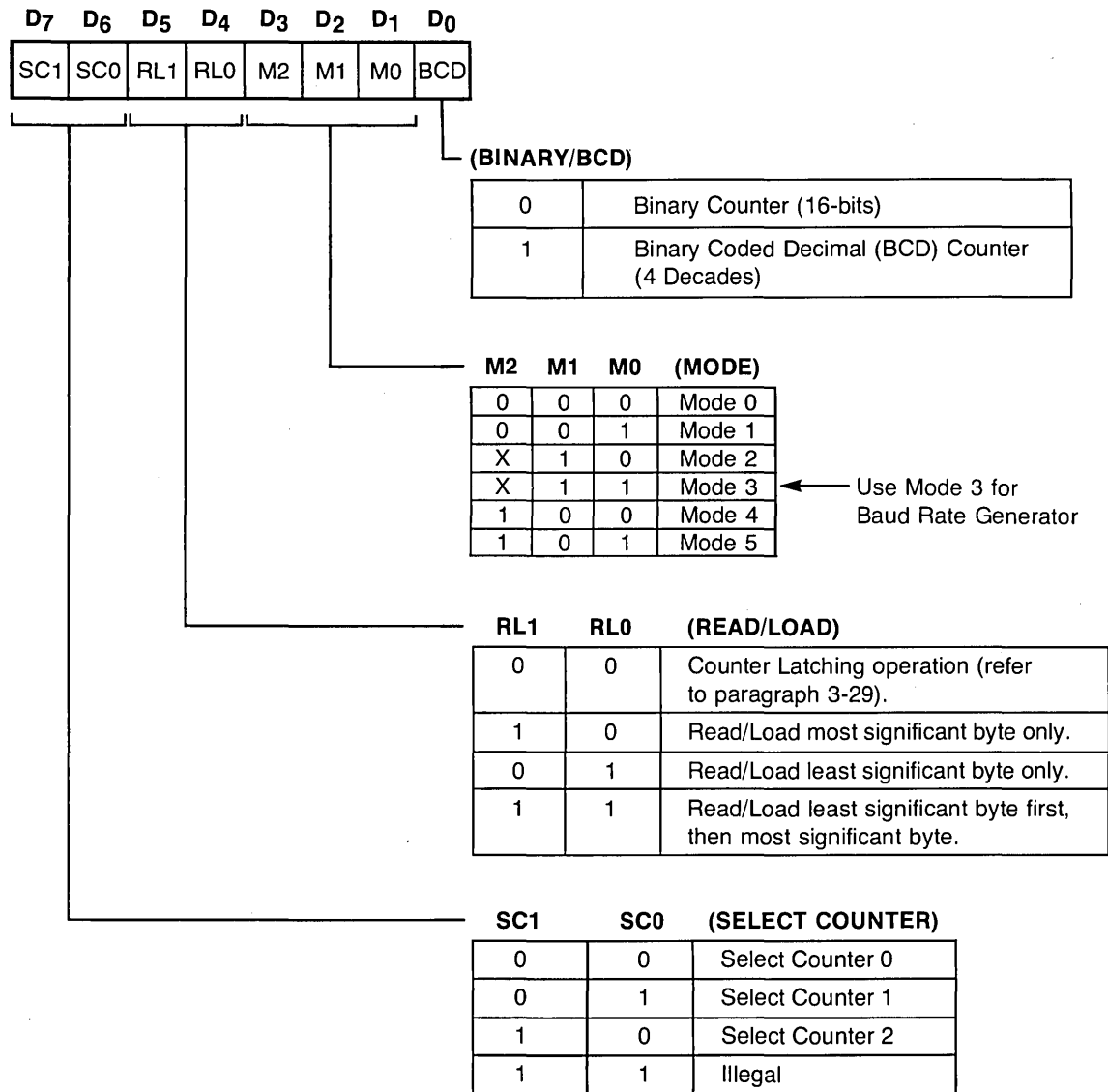


Figure 3-8. PIT Mode Control Word Format

611-7

The mode control word and the count register bytes for any given counter must be entered in the following sequence:

- a. Mode control word.
- b. Least-significant count register byte.
- c. Most-significant count register byte.

As long as the above procedure is followed for each counter, the chip can be programmed in any convenient sequence. For example, mode control words first can be loaded into each of three counters, followed by the least-significant byte, etc. Figure 3-9 shows the two programming sequences described above.

Since all counters in the PIT chip are downcounters, the value loaded in the count registers is decremented. Loading all zeroes into a count register results in a maximum count of  $2^{16}$  for binary numbers or  $10^4$  for BCD numbers.

When a selected count register is to be loaded, it *must* be loaded with the number of bytes programmed in the mode control word. One or two bytes can be loaded, depending on the appropriate down count. These two bytes can be programmed at any time following the mode control word, as long as the correct number of bytes is loaded in order.

The count mode selected in the control word controls the counter output. As shown in figure 3-8, the PIT chip can operate in any of six modes:

**PROGRAMMING FORMAT**

Step		
1		Mode Control Word Counter n
2	LSB	Count Register Byte Counter n
3	MSB	Count Register Byte Counter n

**ALTERNATE PROGRAMMING FORMAT**

Step		
1		Mode Control Word Counter 0
2		Mode Control Word Counter 1
3		Mode Control Word Counter 2
4	LSB	Counter Register Byte Counter 1
5	MSB	Count Register Byte Counter 1
6	LSB	Count Register Byte Counter 2
7	MSB	Count Register Byte Counter 2
8	LSB	Count Register Byte Counter 0
9	MSB	Count Register Byte Counter 0

450-18

**Figure 3-9. PIT Programming Sequence Examples**

- a. Mode 0: Interrupt on terminal count. In this mode, Counters 1 and 2 can be used for auxiliary functions such as generating real-time interrupt intervals. After the count value is loaded into the count register, the counter output goes low and remains low until the terminal count is reached. The output then goes high until either the count register or the mode control register is reloaded.
- b. Mode 1: Programmable one-shot. In this mode, the output of Counter 1 and/or Counter 2 will go low on the count following the rising edge of the GATE input from Port EA. The output will go high on the terminal count. If a new count value is loaded while the output is low, it will not affect the duration of the one-shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse. The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.
- c. Mode 2: Rate generator. In this mode, the output of Counter 1 and/or Counter 2 will be low for one period of the clock input. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses, the present period will not be affected but the subsequent period will reflect the new value. The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Thus, the gate input can be used to synchronize the counter. When Mode 2 is set, the output will remain high until after the count register is loaded; thus, the count can be synchronized by software.
- d. Mode 3: Square wave generator. Mode 3, which is the primary operating mode for Counter 2, is used for generating Baud rate clock signals. In this mode, the counter output remains high until one-half of the count value in the count register has been decremented (for even numbers). The output then goes low for the other half of the count. If the value in the count register is odd, the counter output is high for  $(N + 1)/2$  counts, and low for  $(N - 1)/2$  counts.
- e. Mode 4: Software triggered strobe. After this mode is set, the output will be high. When the count is loaded, the counter begins counting. On terminal count, the output will go low for one input clock period and then go high again. If the count register is reloaded between output pulses, the present count will not be affected, but the subsequent period will reflect the new value. The count will be inhibited while the gate input is low. Reloading the count register will restart the counting for the new value.
- f. Mode 5: Hardware triggered strobe. The counter will start counting on the rising edge of the gate input and the output will go low for one clock period when

the terminal count is reached. The counter is re-triggerable; the output will not go low until the full count after the rising edge of the gate input.

Table 3-7 provides a summary of the counter operation versus the gate inputs. The gate inputs to Counters 0 and 2 are tied high by default jumpers; these gates may optionally be controlled by Port EA. The gate input to Counter 2 is floating high and not optionally controlled.

**Table 3-7. PIT Counter Operation Vs. Gate Inputs**

Modes	Signal Status	Low Or Going Low	Rising	High
0		Disables counting	—	Enables counting
1		—	1) Initiates counting 2) Resets output after next clock	—
2		1) Disables counting 2) Sets output high immediately	Initiates counting	Enables counting
3		1) Disables counting 2) Sets output high immediately	Initiates counting	Enables counting
4		Disables counting	—	Enables counting
5		—	Initiates counting	—

### 3-18. ADDRESSING

As listed in table 3-2, the PIT uses four consecutive I/O addresses: DC through DF. Addresses DC, DD, and DE, respectively, are used in loading and reading the count in Counters 0, 1, and 2. Address DF is used in writing the mode control word to the desired counter.

### 3-19. INITIALIZATION

To initialize the PIT chips, perform the following:

- Write mode control word for Counter 0 to DF. Note that *all* mode control words are written to DF, since mode control word must specify which counter is being programmed. (Refer to figure 3-8.) Table 3-8 provides a sample subroutine for writing mode control words to all three counters.
- Assuming mode control word has selected a 2-byte load, load least-significant byte of count into Counter 0 at DC. (Count value to be loaded is described in paragraphs 3-22 through 3-24). Table 3-9 provides a sample subroutine for loading 2-byte count value.
- Load most-significant byte of count into Counter 0 at DC.

#### NOTE

Be sure to enter the downcount in two bytes if the counter was programmed for a two-byte entry in the mode control word. Similarly, enter the downcount value in BCD if the counter was so programmed.

- Repeat steps a, b, and c for Counters 1 and 2.

**Table 3-8. Typical PIT Control Word Subroutine**

```

;INTTMR INITIALIZES COUNTERS 0, 1, 2.
;COUNTERS 0 AND 1 ARE INITIALIZED AS INTERRUPT TIMERS.
;COUNTER 2 IS INITIALIZED AS BAUD RATE GENERATOR.
;ALL THREE COUNTERS ARE SET UP FOR 16-BIT OPERATION.
;USES-NOTHING; DESTROYS-A.

INTTMR:   MVI   A,30H           ;MODE CONTROL WORD FOR COUNTER 0
          OUT   0DF
          MVI   A,70H           ;MODE CONTROL WORD FOR COUNTER 1
          OUT   0DF
          MVI   A, B6H         ;MODE CONTROL WORD FOR COUNTER 2
          OUT   0DF
          RET

          END
    
```

Table 3-9. Typical PIT Count Value Load Subroutine

	;LOAD0 LOADS COUNTER 0 FROM D&E, D IS MSB, E IS LSB.		
	;USES-D,E; DESTROYS-A.		
LOAD0:	MOV	A,E	;GET LSB
	OUT	ODC	
	MOV	A,D	;GET MSB
	OUT	ODC	
	RET		
	END		

### 3-20. OPERATION

The following paragraphs describe operating procedures for a counter read, clock frequency divider/ratio selection, and interrupt timer count selection.

**3-21. COUNTER READ.** There are two methods that can be used to read the contents of a particular counter. The first method involves a simple read of the desired counter. The only requirement with this method is that, in order to ensure a stable count reading, the desired counter must be *inhibited* by controlling its gate input. Only Counter 0 and Counter 1 can be read using this method because the gate input to Counter 2 is not controllable.

The second method allows the counter to be read "on-the-fly." The recommended procedure is to use a mode control word to latch the contents of the count register; this ensures that the count reading is accurate and stable. The latched value of the count can then be read.

#### NOTE

If a counter is read during count, it is mandatory to complete the read procedure; that is, if two bytes were programmed to the counter, then two bytes *must* be read before any other operations are performed with that counter.

To read the count of a particular counter, proceed as follows (a typical counter read subroutine is given in table 3-10):

- Write counter register latch control word (figure 3-10) to DF. Control word specifies desired counter and selects counter latching operation.
- Perform a read operation of desired counter; refer to table 3-2 for counter addresses.

#### NOTE

Be sure to read one or two bytes, whichever was specified in the initialization mode control word. For two bytes, read in the order specified.

**3-22. CLOCK FREQUENCY/DIVIDE RATIO SELECTION.** Table 2-4 lists the default and optional timer input frequencies to Counters 0 through 2. The timer input frequencies are divided by the counters to generate the 8041/8741 Event Clock (Counter 0), Count Out (Counter 1), and the 8251A Baud Rate Clock (Counter 2).

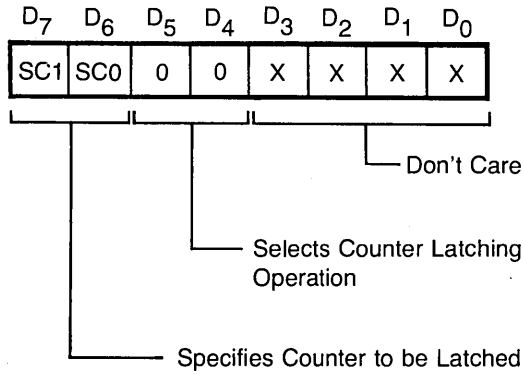
Each counter must be programmed with a downcount number, or count value N. When count value N is loaded into a counter, it becomes the clock divisor. To derive N for either synchronous or asynchronous RS232C operation, use the procedures described in following paragraphs.

**3-23. Synchronous Mode.** In the synchronous mode, the TXC and/or RXC rates equal the Baud rate. Therefore, the count value is determined by

$$N = C/B$$

Table 3-10. Typical PIT Counter Read Subroutine

	;READ1 READS COUNTER 1 ON-THE-FLY INTO D&E. MSB IN D, LSB IN E.		
	;USES-NOTHING; DESTROYS-A,D,E.		
READ1:	MVI	A,40H	;MODE WORD FOR LATCHING COUNTER 1 VALUE
	OUT	ODF	
	IN	ODD	;LSB OF COUNTER
	MOV	E,A	
	IN	ODD	;MSB OF COUNTER
	MOV	D,A	
	RET		
	END		



**Figure 3-10. PIT Counter Register Latch Control Word Format**

450-19A

where N is the count value,  
 B is the desired Baud rate, and  
 C is 1.23 MHz, the input clock frequency.

Thus, for a 4800 Baud rate, the required count value (N) is:

$$N = \frac{1.23 \times 10^6}{4800} = \underline{\underline{256}}$$

If the binary equivalent of count value N = 256 is loaded into Counter 2, then the output frequency is 4800 Hz, which is the desired clock rate for synchronous mode operation.

**3-24. ASYNCHRONOUS MODE.** In the asynchronous mode, the TXC and/or RXC rates equal the Baud rate times one of the following multipliers: X1, X16, or X64. Therefore, the count value is determined by:

$$N = C/BM$$

where N is the count value,  
 B is the desired Baud rate,  
 M is the Baud rate multiplier (1, 16, or 64), and  
 C is 1.23 MHz, the input clock frequency.

Thus, for a 4800 Baud rate, the required count value (N) is

$$N = \frac{1.23 \times 10^6}{4800 \times 16} = \underline{\underline{16}}$$

If the binary equivalent of count value N = 16 is loaded into Counter 2, then the output frequency is 4800 × 16 Hz, which is the desired clock rate for asynchronous mode operation. Count values (N) versus rate multiplier (M) for each Baud rate are listed in table 3-11.

**NOTE**

During initialization, be sure to load the count value (N) into the appropriate counter and the Baud rate multiplier (M) into the 8251A USART.

**Table 3-11. PIT Count Value Vs Rate Multiplier for Each Baud Rate**

Baud Rate (B)	*Count Value (N) For		
	M = 1	M = 16	M = 64
75	16384	1024	256
110	11171	698	175
150	8192	512	128
300	4096	256	64
600	2048	128	32
1200	1024	64	16
2400	512	32	8
4800	256	16	4
9600	128	8	2
19200	64	4	
38400	32	2	
76800	16		

\*Count Values (N) assume clock is 1.23 MHz. Double Count Values (N) for 2.46 MHz clock. Count Values (N) and Rate Multipliers (M) are in decimal.

**3-25. RATE GENERATOR/INTERVAL TIMER.**

Table 3-12 shows the maximum and minimum rate generator frequencies and timer intervals for Counters 0 and 1 when these counters, respectively, have 1.23-MHz and 153.6-KHz clock inputs. The table also provides the maximum and minimum generator frequencies and time intervals and may be obtained by connecting Counters 0 and 1 in series.

**3-26. INTERRUPT TIMER.** To program an interval timer for an interruption terminal count, program the appropriate timer for the correct operating mode (Mode 0) in the control word. Then load the count value (N), which is derived by

$$N = TC$$

where

N is the count value for Counter 0,  
 T is the desired interrupt time interval in seconds, and  
 C is the internal clock frequency (Hz).

Table 3-13 shows the count value (N) required for several time intervals (T) that can be generated for Counters 0 and 1.

**Table 3-12. PIT Rate Generator Frequencies and Timer Intervals**

	Single Timer <sup>1</sup> (Counter 0)		Single Timer <sup>2</sup> (Counter 1)		Dual Timer <sup>3</sup> (0 and 1 in Series)	
	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
Rate Generator (Frequency)	18.75 Hz	614.4 kHz	2.344 Hz	76.8 kHz	0.00029 Hz	307.2 kHz
Real-Time Interrupt (Interval)	1.63 $\mu$ sec	53.3 msec	13 $\mu$ sec	426.67 msec	3.26 $\mu$ sec	58.25 minutes

NOTES:  
 1. Assuming a 1.23-MHz clock input  
 2. Assuming a 153.6-kHz clock input.  
 3. Assuming Counter 0 has 1.23-MHz clock input.

### 3-27. 8255A PPI PROGRAMMING

The three parallel I/O ports interfaced to connector J1 are controlled by an Intel 8255A Programmable Peripheral Interface. Port A includes bidirectional data buffers and Ports B and C include IC sockets for installation of either input terminators or output drivers depending on the user's application.

Default jumpers set the Port A bidirectional data buffers to the input mode. Optional jumpers allow the bidirectional data buffers to be set to the output mode or to allow any one of the eight Port C bits to selectively set the Port A bidirectional data buffers to the input or output mode.

Table 2-11 lists the various operating modes for the three PPI parallel I/O ports. Note that Port A (E8) can be operated in Modes 0, 1, or 2; Port B (E9) and Port C (EA) can be operated in Mode 0 or 1.

### 3-28. CONTROL WORD FORMAT

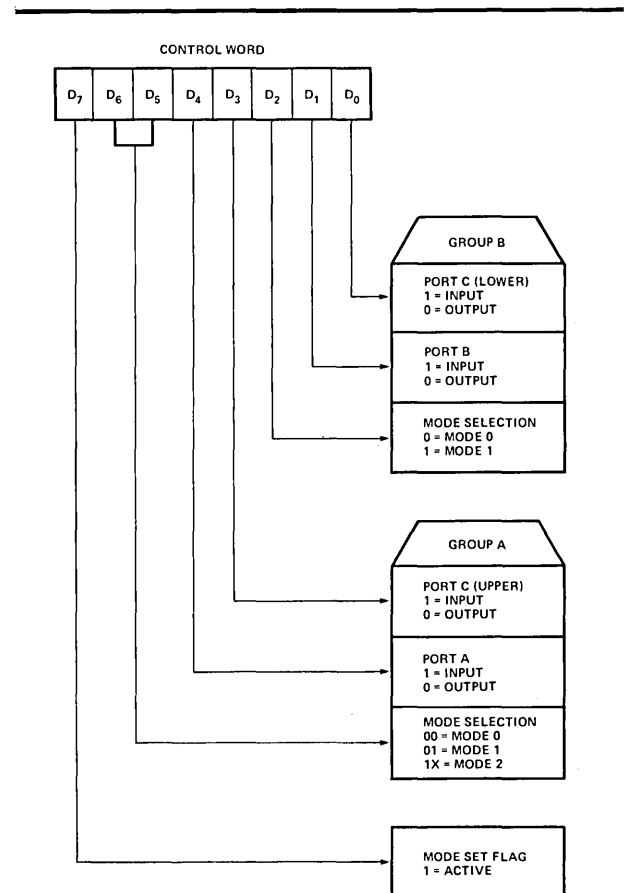
The control word format shown in figure 3-11 is used to initialize the PPI to define the operating mode of the three ports. Note that the ports are separated into two groups. Group A (control word bits 3 through 6) defines the

operating mode for Port A (E8) and the upper four bits of Port C (EA). Group B (control word bits 0 through 2) defines the operating mode for Port B (E9) and the lower four bits of Port C (EA). Bit 7 of the control word controls the mode set flag.

**Table 3-13. PIT Timer Intervals Vs Timer Counts**

T	N*
10 $\mu$ sec	12
100 $\mu$ sec	123
1 msec	1229
10 msec	12288
50 msec	61440

\*Count Values (N) assume clock is 1.23 MHz. Double Count Values (N) for 2.46 MHz clock. Count Values (N) are in decimal.



**Figure 3-11. PPI Control Word Format**

**Table 3-14. Typical PPI Initialization Subroutine**

```

;INT PAR INITIALIZES PARALLEL PORTS.
;USES-NOTHING; DESTROYS-A.
INT PAR:  MVI      A,92H      ;MODE WORD TO PPI PORT A&B IN,C OUT
          OUT      0EBH
          RET
          END

```

**3-29. ADDRESSING**

The PPI uses four consecutive addresses (E8 through EB) for data transfer, obtaining the status of Port C (EA), and for port control. (Refer to table 3-2.)

**3-30. INITIALIZATION**

To initialize the PPI, write a control word to E8. Refer to figure 3-11 and table 3-14 and assume that the control word is 92 (hexadecimal). This initializes the PLI as follows:

- a. Mode Set Flag active
- b. Port A (E8) set to Mode 0 Input
- c. Port C (EA) upper set to Mode 0 Output
- d. Port B (E9) set to Mode 0 Input
- e. Port C (EA) lower set to Mode 0 Output

**3-31. OPERATION**

After the PPI has been initialized, the operation is simply performing a read or a write to the appropriate port.

**3-32. READ OPERATION.** A typical read subroutine for Port A is given in table 3-15.

**3-33. WRITE OPERATION.** A typical write subroutine for Port C is given in table 3-16. As shown in figure 3-12, and of the Port C bits can be selectively set or cleared by writing a control word to EB.

**3-34. 8259A PIC PROGRAMMING**

As described in paragraph 2-20, the PIC monitors the interrupt requests from eight separate sources. When one or more of the interrupt requests are active (true), the PIC determines the following:

- a. Which input signal has the highest priority.
- b. Whether the input signal has a higher priority than the interrupt presently being serviced by the main processor. If so, the interrupt being serviced is interrupted; if not, the input signal is held for later output.
- c. Whether the interrupt input bit is masked.

Thus, the basic functions of the PIC are (1) resolve the priority of interrupt requests and (2) issue a single interrupt request to the 8085A microprocessor (CPU) based on that priority. The output of the PIC is applied directly to the INTR input of the CPU. (Refer to paragraph 3-50.)

**3-35. INTERRUPT PRIORITY MODES**

The PIC has two modes for resolving the priority of interrupt inputs: (1) fully nested mode and (2) rotating mode. The rotating mode has two variations: auto-rotating and specific rotating.

**3-36. FULLY NESTED MODE.** In this mode the PIC input signals are assigned priority from 0 through 7. The PIC operates in this mode unless specifically programmed otherwise. Interrupt IR0 has the highest priority and IR7 has the lowest priority. When an interrupt is acknowledged, the highest priority request is available to the CPU. Lower priority interrupts are inhibited; higher priority

**Table 3-15. Typical PPI Port Read Subroutine**

```

;AREAD READS A BYTE FROM PORT A INTO REG A.
;USES-A; DESTROYS-A.
AREAD:   IN      0E8H
          RET
          END

```



**Table 3-16. Typical PPI Port Write Subroutine**

```

;COUT OUTPUTS A BYTE FROM REG A TO PORT C.
;USES-A; DESTROYS-NOTHING.

COUT:  OUT      0EAH
        RET

        END
    
```

ity interrupts will be able to generate an interrupt that will be acknowledged if the CPU has enabled its own interrupt input through software. An End-Of-Interrupt (EOI) command from the CPU is required to reset the PIC for the next interrupt.

**3-37. AUTO-ROTATING MODE.** In this mode the interrupt priority rotates. Once an interrupt on a given input is serviced, that interrupt assumes the lowest priority. Thus, if there are a number of simultaneous interrupts, the priority will rotate among the interrupts in numerical order. For example, if interrupts IR4 and IR6 request service simultaneously, IR4 will receive the highest priority. After service, the priority level rotates so that IR4 has the lowest priority and IR5 assumes the highest priority. In the worst case, seven other interrupts are serviced before IR4 again has the highest priority. Of course, if IR4 is the only request, it is serviced promptly. In the Auto-Rotating Mode, priority shifts when the PIC chip receives an End-of-Interrupt (EOI) command.

**3-38. SPECIFIC ROTATING MODE.** In this mode the software can change interrupt priority by specifying the bottom priority, which automatically sets the highest priority. For example, if IR5 is assigned the bottom priority, IR6 assumes the highest priority. In the specific rotat-

ing mode, the priority can be rotated by writing a Specific Rotate at EOI (SEOI) command to the PIC chip. This command contains the BCD code of the interrupt being serviced; that interrupt is reset as the bottom priority. In addition, the bottom priority interrupt can be fixed at any time by writing a command word to the PIC chip.

**3-39. INTERRUPT MASK**

One or more of the eight interrupt request inputs can be individually masked during the PIC initialization or at any subsequent time. If an interrupt is masked while it is being serviced, lower priority interrupts are inhibited. There are two ways to enable the lower priority interrupts:

- a. Write an End-of-Interrupt (EOI) command.
- b. Set the Special Mask Mode.

The Special Mask Mode is useful when one or more interrupts are masked. If for any reason an input is masked while it is being serviced, the lower priority interrupts are disabled. However, it is possible to enable the lower priority interrupt with the Special Mask Mode. In this mode, the lower priority lines are enabled until the Special Mask Mode is reset. Higher priorities are not affected.

**3-40. STATUS READ**

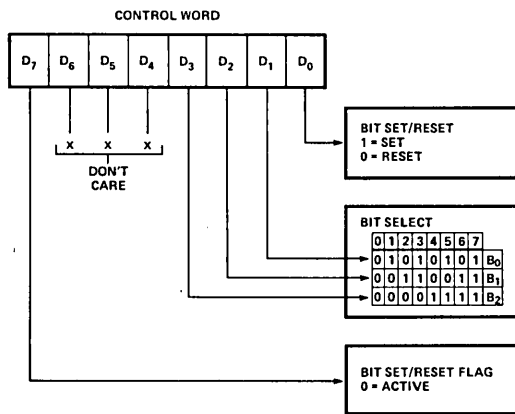
Interrupt request inputs is handled by the following two internal PIC registers:

- a. Interrupt Request Register (IRR), which stores all interrupt levels that are requesting service.
- b. In-Service Register (ISR), which stores all interrupt levels that are being serviced.

Either register can be read by writing a suitable command word and then performing a read operation.

**3-41. INITIALIZATION COMMAND WORDS**

The eight devices serviced by the PIC have eight addresses equally spaced in memory that can be programmed at intervals of four or eight bytes. Interrupt service routines for these eight devices thus occupy a 32- or 64-byte block, respectively, of memory. The address format for device interrupt service routine is shown in figure 3-13.



**Figure 3-12. PPI Port C Bit Set/Reset Control Word Format**

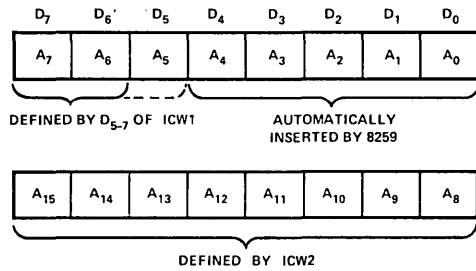


Figure 3-13. PIC Device Interrupt Addresses

Bits 0-4 are automatically inserted by the PIC whereas bits 6-15 are programmed by Initialization Command Words ICW1 and ICW2. If the address interval is eight bytes, bit 5 is automatically inserted by the PIC; if the address interval is four bytes, bit 5 is programmed in ICW1. Thus, the 32-byte or 64-byte block of addresses reserved for interrupt service routines can be located anywhere in the available memory space. Table 3-17 shows the address format inserted by the PIC for each device.

Initialization of the PIC consists of writing two or three 8-bit Initialization Command words as shown in figure 3-14. Since there are no slave PIC's, the initialization for the one PIC consists of writing two Initialization Command Words as follows:

- a. The first Initialization Command Word (ICW1) consists of the following:
  - (1) Bits 5-7 specify the most-significant bits of the lower address byte of the interrupt service routine.
  - (2) Bit 3 establishes whether the interrupts are requested by a positive-true or requested by a low-to-high transition input. This applies to all input requests handled by the PIC. In other words, if bit 3-1, a low-to-high transition is required to request an interrupt on any of the eight levels handled by the PIC.
  - (3) Bit 2 specifies a 4-byte or 8-byte address interval.

- (4) Bit 1 specifies whether or not there are slave (cascaded) PIC's. Since there are no slave PIC's, set bit 1 = 1.
- (5) Bits 0 and 4 identify the word as ICW1.
- b. The second word (ICW2) specifies the upper byte (bits 8-15) of the interrupt service routine.

**3-42. OPERATION COMMAND WORDS**

After being initialized, the PIC can be programmed at any time for various interrupt modes. The Operation Command Word (OCW) formats are shown in figure 3-15 and discussed in paragraph 3-45.

**3-43. ADDRESSING**

The PIC uses two consecutive addresses for writing to and reading internal registers. Address functions pertinent to programming are identified in table 3-2.

**3-44. INITIALIZATION**

To initialize the PIC, proceed as follows (table 3-17 provides a typical initialization subroutine);

- a. Disable system interrupts by executing a DI (Disable Interrupts) instruction.
- b. Write ICW1 to D8.
- c. Write ICW2 to D9.
- d. Enable system interrupts by executing an EI (Enable Interrupts) instruction.

**NOTE**

The PIC operates in the fully nested mode after the initialization sequence without requiring any Operation Control Word (OCW).

**3-45. OPERATION**

After initialization, the PIC chips can be programmed at any time for the following operations:

Table 3-17. PIC Device Address Insertion

Device	Lower Routine Address Byte															
	Interval = 4								Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
IR7	A7	A6	A5	1	1	1	0	0	A7	A6	1	1	1	0	0	0
IR6	A7	A6	A5	1	1	0	0	0	A7	A6	1	1	0	0	0	0
IR5	A7	A6	A5	1	0	1	0	0	A7	A6	1	0	1	0	0	0
IR4	A7	A6	A5	1	0	0	0	0	A7	A6	1	0	0	0	0	0
IR3	A7	A6	A5	0	1	1	0	0	A7	A6	0	1	1	0	0	0
IR2	A7	A6	A5	0	1	0	0	0	A7	A6	0	1	0	0	0	0
IR1	A7	A6	A5	0	0	1	0	0	A7	A6	0	0	1	0	0	0
IR0	A7	A6	A5	0	0	0	0	0	A7	A6	0	0	0	0	0	0

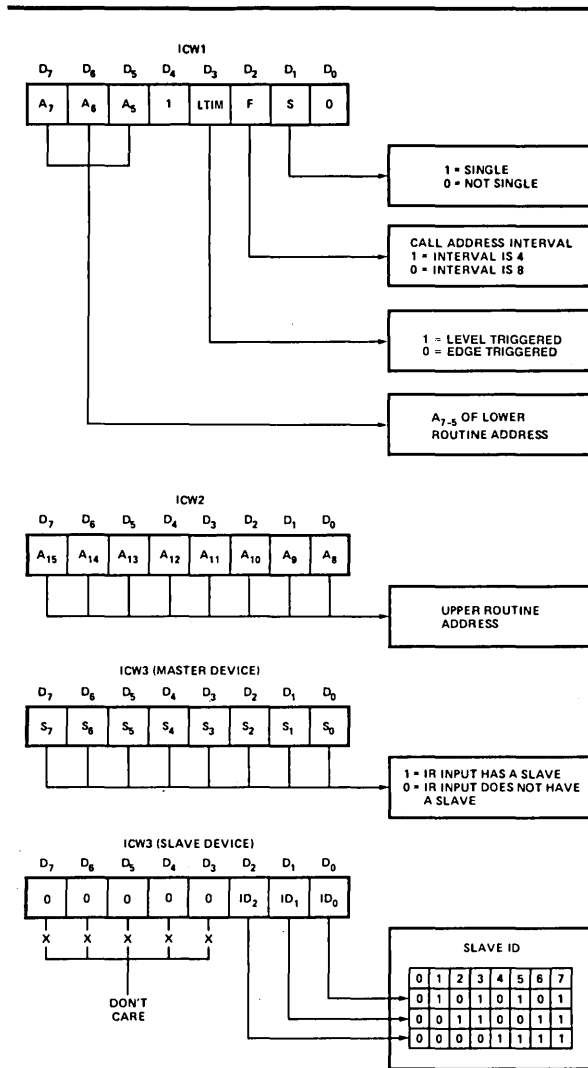


Figure 3-14. PIC Initialization Command Word Formats

611-8

- a. Auto-rotating priority.
- b. Specific rotating priority.
- c. Status read of Interrupt Request Register (IRR).
- d. Status read of In-Service Register (ISR).
- e. Interrupt mask bits set, reset, or read.
- f. Special mask mode set or reset.

Table 3-19 lists details of the above operations. Note that an End-Of-Interrupt (EOI) or a Special End-Of-Interrupt (SEOI) command is required at the end of each interrupt service routine to reset the ISR. The EOI command is used in the fully nested and auto-rotating priority modes and the SEOI command, which specifies the bit to be reset, is used in the specific rotating priority mode. Tables 3-20 through 3-24 provide typical subroutines for the following:

- a. Read IRR (table 3-20).
- b. Read ISR (table 3-21).
- c. Set mask register (table 3-22).
- d. Read mask register (table 3-23).
- e. Issue EOI command (table 3-24).

### 3-46. 8041/8741A UPI PROGRAMMING

Table 3-2 lists the addresses used for writing data and commands and for reading data and status words. Assembly language programming information for the Intel 8041/8741A is given in the *Intel UPI-41 User's Manual*, Order No. 9800504, and *Control With UPI-41*, Application Note AP-27.

Figure 3-16 illustrates the internal configuration of the UPI data bus buffer and status register. The CPU accesses the UPI data bus buffer register via I/O Read and Write Commands to E4 or E6. The CPU accesses the UPI status via an I/O Read Command to E5 or E7.

Table 3-18. Typical PIC Initialization Subroutine

```

;INT59 INITIALIZES THE PIC. A 64-BYTE ADDRESS BLOCK BEGINNING WITH
;4000H IS SET UP FOR INTERRUPT SERVICE ROUTINES.
;PIC MASK IS SET, DISABLING ALL PIC INTERRUPTS.
;USES-SMASK; DESTROYS-A.

                                EXTRN    SMASK
INT59:    CALL    SETD
           MVI    A,12H
           OUT    0D8H           ;ICW1 TO PIC
           MVI    A,40H
           OUT    0D9H           ;ICW2 TO PIC
           MVI    0FFH
           CALL   SMASK
           RET

                                END
    
```

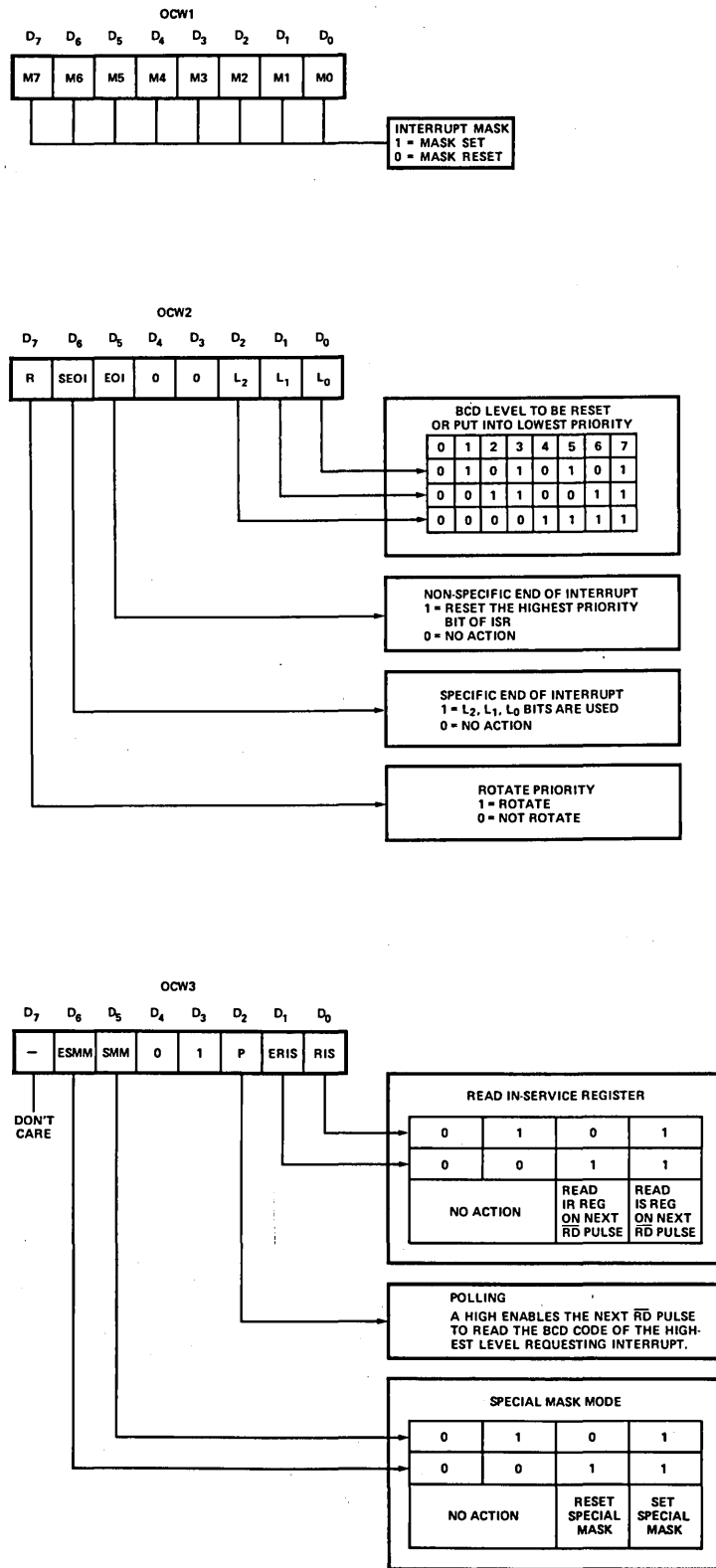


Figure 3-15. PIC Operation Control Word Formats

**Table 3-19. PIC Operation Procedures**

Operation	Procedure																																																
<p>Auto-Rotating Priority Mode</p>	<p>To set: In OCW2, write a Rotate Priority at EOI command (A0H) to D8.</p> <p>Terminate interrupt and rotate priority: In OCW2, write EOI command (20H) to D8.</p>																																																
<p>Specific Rotating Priority Mode</p>	<p>To set: In OCW2, write a Rotate Priority at SEOI command in the following format to D8:</p> <table border="1" data-bbox="899 552 1406 646"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> <tr> <td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>L2</td><td>L1</td><td>L0</td> </tr> </table> <p>BCD of IR line to be reset and/or put into lowest priority.</p> <p>To terminate interrupt and rotate priority: In OCW2, write an SEOI command in the following format to D8.</p> <table border="1" data-bbox="899 764 1406 879"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>L2</td><td>L1</td><td>L0</td> </tr> </table> <p>BCD of ISR flip-flop to be reset.</p> <p>To rotate priority without EOI: In OCW2, write a command word in the following format to D8.</p> <table border="1" data-bbox="899 997 1406 1092"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> <tr> <td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>L2</td><td>L1</td><td>L0</td> </tr> </table> <p>BCD of bottom priority IR line.</p>	D7	D6	D5	D4	D3	D2	D1	D0	1	1	1	0	0	L2	L1	L0	D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	0	0	L2	L1	L0	D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	0	0	L2	L1	L0
D7	D6	D5	D4	D3	D2	D1	D0																																										
1	1	1	0	0	L2	L1	L0																																										
D7	D6	D5	D4	D3	D2	D1	D0																																										
0	1	1	0	0	L2	L1	L0																																										
D7	D6	D5	D4	D3	D2	D1	D0																																										
1	1	0	0	0	L2	L1	L0																																										
<p>Interrupt Request Register (IRR) Status</p>	<p>The IRR stores a "1" in the associated bit for each IR input line that is requesting an interrupt. To read the IRR (refer to footnote):</p> <p>(1) Write 0BH to D8. (2) Read D8. Status format is as follows:</p> <table border="1" data-bbox="899 1310 1406 1356"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> </table> <p>IR Line:      7    6    5    4    3    2    1    0</p>	D7	D6	D5	D4	D3	D2	D1	D0																																								
D7	D6	D5	D4	D3	D2	D1	D0																																										
<p>In-Service Register (ISR) Status</p>	<p>The ISR stores a "1" in the associated bit for priority inputs that are being serviced. The ISR is updated when an EOI command is issued. To read the ISR (refer to footnote):</p> <p>(1) Write 0AH to D8. (2) Read D8. Status format is as follows:</p> <table border="1" data-bbox="899 1591 1406 1638"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> </table> <p>IR Line:      7    6    5    4    3    2    1    0</p> <p>Be sure to reset ISR bit at end-of-interrupt when in the following modes: Auto-Rotating (both types) and Special Mask. To reset ISR in OCW2, write:</p> <table border="1" data-bbox="899 1759 1406 1854"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> <tr> <td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>L2</td><td>L1</td><td>L0</td> </tr> </table> <p>BCD identifies bit to be reset.</p>	D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	0	0	L2	L1	L0																								
D7	D6	D5	D4	D3	D2	D1	D0																																										
D7	D6	D5	D4	D3	D2	D1	D0																																										
0	1	1	0	0	L2	L1	L0																																										

**Table 3-19. PIC Operation Procedures (Continued)**

Operation	Procedure								
Interrupt Mask Register	<p>To set mask bits in OCW1, write the following mask byte to D9:</p> <table border="1" data-bbox="816 359 1325 407"> <tr> <td>D7</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td> </tr> </table> <p>IR Bit Mask:           M7 M6 M5 M4 M3 M2 M1 M0                      1 = Mask Set, 0 = Mask Reset</p> <p>To read mask bits, read D9.</p>	D7	D6	D5	D4	D3	D2	D1	D0
D7	D6	D5	D4	D3	D2	D1	D0		
Special Mask Mode	<p>The Special Mask Mode enables desired bits that have been previously masked; lower priority bits are also enabled.</p> <p>To set, write 68H to D8.</p> <p>To reset, write 48H to D8.</p>								
<p>NOTE:                      If previous operation was addressed to same register, it is not necessary to rewrite the OCW.</p>									

**Table 3-20. Typical PIC Interrupt Request Register Read Subroutine**

```

;RR0 READS PIC INTERRUPT REQUEST REGISTER.
;USES-NOTHING; DESTROYS-A.

RR0:   MVI     A,0AH      ;OCW3 RR INSTRUCTION TO PIC
        OUT    0D8H
        IN     0D8H
        RET
        END
    
```

**Table 3-21. Typical PIC In-Service Register Read Subroutine**

```

;RIS0 READS PIC IN-SERVICE REGISTER.
;USES-NOTHING; DESTROYS-A.

RIS0:  MVI     A,08H      ;OCW3 RIS INSTRUCTION TO PIC
        OUT    0D8H
        IN     0D8H
        RET
        END
    
```

**Table 3-22. Typical PIC Set Mask Register Subroutine**

```

;SMASK STORES A REG INTO PIC MASK REGISTER.
;A ONE MASKS OUT AN INTERRUPT, A ZERO ENABLES IT.
;USES-A; DESTROYS-NOTHING.

SMASK: OUT     0D9H
        RET
        END
    
```

**Table 3-23. Typical PIC Mask Register Read Subroutine**

```

;RMASK READS PIC MASK REGISTER.
;USES-NOTHING; DESTROYS-A.

RMASK:  IN      0D9H
        RET
        END
    
```

**Table 3-24. Typical PIC End-of-Interrupt Command Subroutine**

```

;EOI ISSUES END-OF-INTERRUPT TO PIC.
;USES-NOTHING; DESTROYS-A.

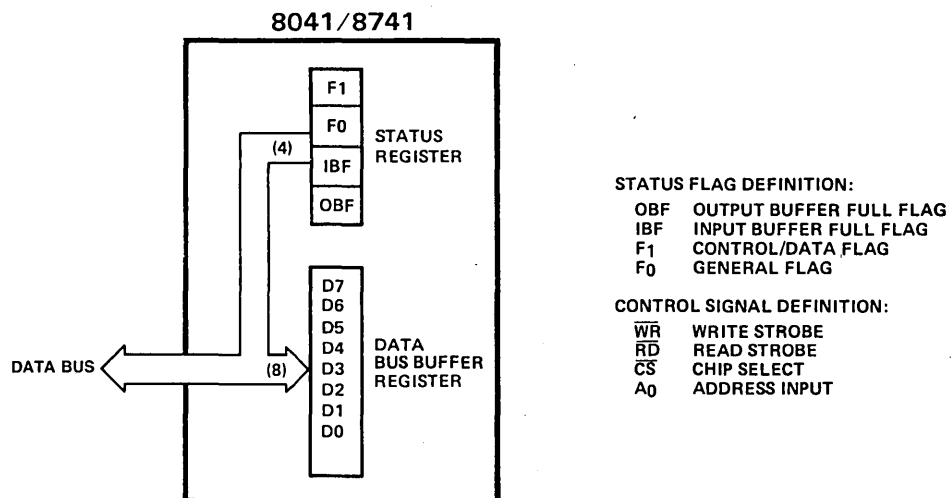
EOI:    MVI      A,20H      ;NON-SPECIFIC EOI
        OUT      0D8H
        RET
        END
    
```

**NOTE**

The data bus buffer configuration does not allow the CPU to read back data it has previously written into the data bus buffer register. Also, the UPI cannot input data from the data bus buffer register that it previously output. The CPU can only read data that the UPI has output and the UPI can only input data that the CPU has written.

The 4-bit status register indicates the status of the data bus buffer register and implements the handshaking protocol required for two-way data transfer. The four bits comprising the status register are defined as follows:

- a. Bit 0: Output Buffer Full. The OBF flag is automatically set when the UPI loads the data bus buffer register and cleared when the CPU reads the data bus buffer register.



**Figure 3-16. UPI Data Bus Buffer and Status Registers**

**Table 3-25. Typical UPI Data Byte Read Subroutine**

```

;IN41 READS DATA BYTE FROM UPI.
;USES-NOTHING; DESTROYS-A.

IN41:  IN    0E5H    ;INPUT STATUS
        ANI   1      ;CHECK OBF FLAG
        JZ    IN41   ;WAIT IF EMPTY
        IN    0E4H
        RET

        END
    
```

**Table 3-26. Typical UPI Data Byte Write Subroutine**

```

;OUT 41 WRITES DATA BYTE TO UPI.
;USES-NOTHING; DESTROYS-A.

OUT41:  IN    0E5H    ;INPUT STATUS
        ANI   2      ;CHECK IBF FLAG
        JZ    OUT41  ;WAIT IF NOT READY
        OUT   0E4H
        RET

        END
    
```

- b. Bit 1: Input Buffer Full. The IBF flag is set when the CPU writes a character to the data bus buffer register and cleared when the UPI inputs the contents of the data bus buffer register to its accumulator.
- c. Bit 2: F0. This general purpose flag, which can be cleared or toggled under UPI software control, is used to transfer UPI status information to the CPU.
- d. Bit 3: F1 (Control/Data). This flag is set to the condition of the A0 input line when the CPU writes a character to the data bus buffer register. This flag can be cleared or toggled under UPI software control.

tion independent of the state of the interrupt enable or interrupt mask. The priority and vector location of each of the restart interrupts are given in table 3-27.

**Table 3-27. 8085A CPU Restart Interrupt Vectors**

Interrupt	Vector Location	Priority
TRAP	24	Highest
RST 7.5	3C	2nd
RST 6.5	34	3rd
RST 5.5	2C	4th

Tables 3-25 and 3-26, respectively, provides typical routines for inputting and outputting a data byte from and to the UPI.

### 3-47. INTERRUPT HANDLING

The iSBC 80/30 includes five interrupt inputs: TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR. The three “restart” interrupts (7.5, 6.5, and 5.5) are maskable; the TRAP is also a “restart” interrupt but is nonmaskable.

The RST 7.5, RST 6.5, and RST 5.5 interrupts cause the internal execution of an RST instruction if the interrupts are enabled and if the interrupt mask is not set by a previously executed SIM instruction. The nonmaskable TRAP interrupt causes the execution of an RST instruc-

### 3-48. TRAP INPUT

There are special considerations that must be understood when the TRAP interrupt is used. The fact that the TRAP interrupt is nonmaskable can present problems in at least two areas.

Interrupt driven systems often contain parameters that must be modified only within critical regions. A critical region can be roughly defined as a section of code that once begun must complete execution before it or another critical region that corresponds to the same system parameter(s) can be executed. A TRAP interrupt handler cannot safely alter such parameters either directly or indirectly by causing the execution of procedures or tasks that may alter such parameters.



If the hardware generates a TRAP interrupt on power up or power fail, the system must be able to process the TRAP interrupt before it is completely initialized. It should also take into account that an interrupt routine that runs with interrupts disabled can still be interrupted by a TRAP.

Because of these considerations, it is recommended that the TRAP interrupt only be used for system startup and/or catastrophic error handling.

It should be noted that TRAP does not destroy a previously established interrupt enable status. Executing the first RIM instruction following a TRAP interrupt yields the interrupt enable status as it was before the TRAP occurred. Following this first *mandatory* RIM instruction, subsequently executed RIM instructions provide current interrupt enable status.

### 3-49. RST 7.5, 6.5, 5.5 INPUTS

These interrupts can be individually masked by a SIM instruction and can thus be prevented from interrupting the CPU. The priorities shown in table 3-27 does not take into account the priority of a routine that was started by a higher priority interrupt. An RST 5.5 interrupt can interrupt an RST 7.5 routine if the interrupts are *re-enabled* before the end of the RST 7.5 routine.

The RST 6.5 and 5.5 interrupts are *high level sensitive* and, in order to be recognized, must remain high. The

RST 7.5 interrupt is *rising edge sensitive* and only a pulse is required to set the interrupt request; this request will be remembered until the request is serviced or reset by a SIM instruction.

The TRAP interrupt, which is both edge and level sensitive, must have a leading (positive-going) edge and then remain high until serviced.

### 3-50. INTR INPUT

The INTR input from the 8259A PIC has the lowest priority and is sampled only during the last clock cycle of a given instruction. When INTR is active, the CPU Program Counter (PC) is inhibited and three bytes, timed by INTA/ pulses, are transferred from the PIC to the CPU:

- a. Byte 1 = CALL instruction (11001101)
- b. Byte 2 = Lower byte of routine address
- c. Byte 3 = Upper byte of routine address.

The 16-bit routine address must be on a 32-byte boundard or 64-byte boundary as determined by the Initialization Command Words previously written to the PIC. (Refer to paragraph 3-41.)

The INTR is enabled or disabled by software. It is disabled by "reset" and immediately after an interrupt is accepted.



## 4-1. INTRODUCTION

This chapter provides a functional description and a circuit analysis of the iSBC 80/30 Single Board Computer. Figures 4-1 and 4-2, located at the end of this chapter, are simplified foldout block diagrams that illustrate the functional interface between the 8085A microprocessor (CPU) and the on-board facilities and between the CPU and the system facilities via the Multibus. Also shown in figure 4-2 is the dual port control logic that allows the iSBC 80/30 to function in a master/slave relationship with the Multibus to allow another bus master to access the on-board RAM.

## 4-2. FUNCTIONAL DESCRIPTION

A brief description of the functional blocks comprising the iSBC 80/30 is given in following paragraphs. An operational circuit analysis is given beginning with paragraph 4-14.

## 4-3. CLOCK CIRCUITS

The clock circuit composed of A12, A13, and A21 is stabilized by a 22.1184-MHz crystal. This circuit, which provides the various clock frequencies required for on-board activities, generates a power-up Reset signal to initialize the system to a known internal state; a Reset signal can also be initiated by a signal supplied via auxiliary connector P2.

The clock circuit composed of A55 and A57 is stabilized by an 18.432-MHz crystal. This frequency is divided by two to provide the 9.22-MHz Bus Clock (BCLK/) and Constant Clock (CCLK/) to the Multibus. (The BCLK/ signal is also used by Bus Controller A67.) Removable jumpers are provided to allow this on-board clock circuit to be disabled if some other source supplies BCLK/ and CCLK/ to the Multibus.

## 4-4. CENTRAL PROCESSOR UNIT

The 8085A Microprocessor (CPU), which is the heart of the single board computer, performs system processing functions and generates the address and control signals required to access memory and I/O devices. The AD0-AD7 pins are used to multiplex the 8-bit input/output data and the lower eight bits of the address. During the first part of a machine cycle, for example, the lower eight bits of address are strobed into Latch A23 by the Address Latch Enable (ALE) signal; the outputs of A23 are combined with the upper eight bits of the address to form the 16-bit address bus. During the remainder of the machine cycle, AD0-AD7 pins of the CPU are used for data input/output.

## 4-5. INTERVAL TIMER

The 8253 Programmable Interval Timer (PIT) includes three independently controlled counters that provide optional (jumper selectable) timing inputs to the on-board I/O devices and the CPU interrupts. The clock frequency of 2.46 MHz, 1.23 MHz, or 153.6 kHz, which is derived from the clock circuit composed of A12, A13, and A21, provides the basic timing input.

Counter 2 provides timing for the serial I/O port (8251A USART). This counter, in conjunction with the USART, can provide programmable Baud rates from 110 to 9600. Counter 0 can be used as a timing input to the optional 8041/8741 UPI which, in turn, can connect this signal to its own internal timer input. Counter 0, if not employed by an optional UPI, can be used as an interval timer to generate a CPU interrupt. Counter 1, which is the system interval timer and can also generate a CPU interrupt, has a range of 1.6 microseconds to 853.3 milliseconds. If longer times are required, Counters 0 and 1 can be cascaded to provide a single timer with a maximum delay of more than 50 hours.

## 4-6. SERIAL I/O

The 8251A Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides RS232C compatibility and is configured as a data terminal. Synchronous or asynchronous mode, character size, parity bits, stop bits, and Baud rates are all programmable. Data, clocks, and control lines to and from connector J3 are buffered with drivers and receivers.

A second serial I/O channel may be derived from programming the optional 8041/8741 UPI to simulate the USART. This second channel is limited in speed and the number of RS232C interface lines (one driver and one receiver) but is sufficient to communicate with a simple CRT interface.

## 4-7. PARALLEL I/O

The 8255A Programmable Peripheral Interface provides 24 programmable I/O lines. A bidirectional bus driver is included to interface eight of the I/O lines to connector J1. Two IC sockets are provided so that, depending on the application, TTL drivers or I/O terminators may be installed to complete the interface to connector J1. The 24 lines are grouped into three ports of eight lines each; these ports can be programmed to be simple I/O ports or strobed I/O ports with handshaking. One port can be programmed as a bidirectional port with control lines. The iSBC 80/30 includes various optional features such as RS232C interface lines, timer gates, and interrupts that can be controlled by the parallel I/O lines.

#### 4-8. UNIVERSAL PERIPHERAL INTERFACE

The optional 8041/8741A Universal Peripheral Interface (UPI) is a complete single chip microcomputer that interfaces directly to the 8085A CPU through the I/O structure. The UPI can interface devices such as a printer controller or a keyboard scanner while relying on the CPU only for data transfer.

#### 4-9. INTERRUPT CONTROL

The interrupt section provides 12 priority interrupts. Eight interrupts connect to the inputs of the 8259 Programmable Interrupt Controller (PIC); the remaining four interrupts connect directly to the 8085A CPU. All interrupts, except TRAP, are maskable; the TRAP interrupt is intended to handle a power fail (PFI) signal input via auxiliary connector P2.

There are 18 jumper-selectable interrupt sources: PPI (2), UPI (1), USART (2), PIT (2), external via J1 and J2 (2), power fail via P2 (1), and Multibus (8).

#### 4-10. ROM/EPROM CONFIGURATION

IC sockets A25 and A37 are provided for user installation of ROM/EPROM chips. Jumpers are provided to accommodate either 1K, 2K, or 4K chips. The on-board ROM/EPROM address space begins at 0000. This space is normally reserved for ROM/EPROM use only; however, ROM/EPROM may be disabled through an optional jumper connect on the 8255A PPI output port.

#### 4-11. RAM CONFIGURATION

The iSBC 80/30 includes 16K of dynamic RAM implemented with eight Intel 2117 chips and an Intel 8202 Dynamic RAM Controller. Dual-port control logic interfaces this RAM with the Multibus so that the iSBC 80/30 can perform as a slave RAM device when not acting as a bus master. The slave RAM decode logic allows extended Multibus addressing of up to 20 address lines. This allows bus masters with 20-bit addressing capability to partition the iSBC 80/30 into any 8K or 16K segment in a 1-megabyte address space. The 8085A CPU, however, has only a 16-bit address capability and the 16K RAM must reside in a 64K address space. Hardware jumpers allow the 8085A CPU or the system to access either 8K or 16K of on-board RAM.

#### 4-12. BUS INTERFACE

The interface to the Multibus includes an Intel Bus Controller, bidirectional address and data bus drivers, and the bus interrupt driver/receivers. The bus controller allows the iSBC 80/30 to operate as a bus master in a serial or parallel priority arrangement with other bus masters in the system in which the 8085A CPU can request the Multibus only when a bus resource is needed.

#### 4-13. DUAL PORT CONTROL

The dual-port control logic allows the iSBC 80/30 to function as a slave RAM device when not acting as a bus master. The dual-port control logic enables or disables the internal data and address buses, depending on whether the access is from the 8085A CPU or from the Multibus. For RAM access, the 8085A CPU has priority over a Multibus request.

An extended addressing facility is provided to allow the 16K RAM to reside within a 1-megabyte address space when accessed from the Multibus. This is useful when one or more bus masters have a 20-bit addressing capability. The 8085A CPU, however, can only access this memory in the lowest 64K address space.

#### 4-14. CIRCUIT ANALYSIS

The schematic diagram for the iSBC 80/30 is given in figure 5-2. The schematic diagram consists of nine sheets, each of which includes grid coordinates. Signals that transverse from one sheet to another are assigned grid coordinates at both the signal source and signal destination. For example, the grid coordinates 2ZB1 locate a signal source (or signal destination as the case may be) on sheet 2 Zone B1.

Both active-high and active-low signals are used. A signal mnemonic that ends with a virgule (e.g., DAT7/) denotes that the signal is active low ( $\leq 0.4V$ ). Conversely, a signal mnemonic without a virgule (e.g., ALE) denotes that the signal is active high ( $\geq 2.0V$ ).

Figures 4-1 and 4-2 at the end of this chapter are simplified block diagrams of the input/output, interrupt, and memory sections. These block diagrams will be helpful in understanding both the addressing scheme and the internal bus structure of the board.

#### 4-15. INITIALIZATION

When power is applied in a start-up sequence, the contents of the 8085A CPU program counter, instruction register, and interrupt enable flip-flop are subject to random factors and cannot be predicted. For this reason, a power-up sequence is used to set the CPU, bus controller, and I/O ports to a known internal state.

When power is initially applied to the iSBC 80/30, capacitor C7 (7ZA7) begins to charge through resistor R2. The charge developed across C7 is sensed by a Schmitt trigger, which is internal to Clock Generator A13. The Schmitt trigger converts the slow transition appearing at pin 2 into a clean, fast-rising synchronized RESET output signal at pin 1. The RESET signal is inverted by A29-11

(2ZC7) to produce the Initialize signal INIT/. The INIT/ signal clears the CPU program counter, instruction register, and interrupt enable flip-flop; resets the parallel I/O ports to the input mode; resets the serial I/O port to the "idle" mode; clears the UPI internal program counter and status flags; resets the bus controller; and, via the Multibus, sets the system to a known internal state.

The initialization described above can be performed at any time by inputting an AUX RESET/ signal via connector P2.

#### 4-16. CLOCK CIRCUITS

The time base for Bus Clock signal BCLK/ and Constant Clock signal CCLK/ is provided by Clock Generator A57 (5ZA6) and crystal Y2. The 18.43-MHz output of A57 is divided by A55 and driven onto the Multibus through jumpers 165-166 and 167-168. The BCLK/ signal is also used as the clock input to Bus Controller A67.

The time base for all other functions on the board is provided by Clock Generator A13 and crystal Y1. The crystal frequency of 22.12-MHz appearing at the OSC output of A13 is buffered and applied to the dual port control logic (9ZC8) and to RAM Controller A66 (3Z5D). The 22.12-MHz OSC output is also divided by A44 and A31 to develop the 5.53-MHz clock for 8085 CPU A36 (2ZD6) and for optional UPI A20 (5Z4D).

The crystal frequency is divided by nine internally by A13 to produce a 2.46-MHz clock at its  $\Phi$ 2 TTL output. This signal clocks Divider A12 and supplies a selectable clock signal to PIT A21. Divider A12 divides the clock input by two and by 16 to produce 1.23-MHz and 153.6-kHz selectable clocks for PIT A21 and the parallel I/O port.

#### 4-17. 8085A CPU TIMING

The 8085A CPU internally divides the 5.53-MHz clock input by two to develop the timing requirements for the various time-dependent functions. These functions are described in following paragraphs.

**4-18. INSTRUCTION TIMING.** The execution of any program consists of read and write operations, where each operation transfers one byte of data between the CPU and memory or between the CPU and an I/O device. Although the CPU can vary the address, data, type, and sequence of operations, it is capable of performing only a basic read or write operation. With the exception of a few control lines, such as Address Latch Enable (ALE), these read and write operations are the only communication necessary between the CPU and the other components to execute any instruction.

An *instruction cycle* is the time required to fetch and execute an instruction. During the fetch phase, the

selected instruction (consisting of up to three bytes) is read from memory and stored in the operating registers of the CPU. During the execution phase, the instruction is decoded by the CPU and translated into specific processing activities.

Each instruction cycle consists of up to five machine cycles. A *machine cycle* is required each time the CPU accesses memory or an I/O device. The fetch phase requires one machine cycle for each byte to be fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instructions from memory; other instructions, however, require an additional machine cycle(s) to write or read data to or from memory or I/O devices.

Every instruction cycle has at least one reference to memory during which time the instruction is fetched. An instruction cycle must always have a fetch, even if the execution of that instruction requires no reference to memory. The first machine cycle in every instruction cycle is therefore a fetch, and beyond that there are no specific rules. For instance, the IN (input) and OUT (output) instructions each require three machine cycles: fetch (to obtain the instruction), memory read (to obtain the I/O address of the device), and an input or output machine cycle (to complete the transfer).

Each machine cycle consists of a minimum of three and a maximum of six states designated  $T_1$  through  $T_6$ . A *state* is the smallest unit of processing activity and is defined as the interval between two successive falling edges of the CPU clock. Each state (or CPU clock cycle) has a duration of 350 nanoseconds (derived by dividing the 5.53-MHz frequency by two).

Every machine cycle normally consists of three T-states with the exception of an opcode fetch, which consists of either four or six T-states. The actual number of states required to execute any instruction depends on the instruction being executed, the particular machine cycle within the instruction cycle, and the number of *wait* states inserted into the machine cycle. The wait state is initiated when the READY input to the CPU is pulsed low.

There are no wait states imposed when the CPU is addressing on-board ROM/PROM or on-board I/O. There will be wait states imposed, however, in the following operations:

- a. When addressing on-board RAM and a refresh cycle in progress.
- b. When addressing on-board RAM and a slave memory read or write is in progress. (One wait state is always imposed when performing a write to on-board RAM.)
- c. When addressing system RAM, PROM, or I/O devices.

Figure 4-3 is presented to show the relationship between an instruction cycle, machine cycle, and T-state. This example shows the execution of a Store Accumulator Direct (STA) instruction involving on-board memory. Notice that for this instruction the opcode fetch (machine cycle  $M_1$ ) requires four T-states and the remaining three cycles each require three T-states.

The opcode fetch is the only machine cycle that requires more than three T-states. This is because the CPU must interpret the requirements of the opcode fetched during  $T_1$  through  $T_3$  before it can decide what must be done in the remaining T-state(s).

There are seven types of machine cycles, each of which can be differentiated by the states of three CPU status lines ( $IO/\overline{M}$ ,  $S_0$ , and  $S_1$ ) and three CPU control lines ( $\overline{RD}$ ,  $\overline{WR}$ , and  $\overline{INTA}$ ). Table 4-1 lists the states of the CPU status and control lines during each of the seven machine cycles and during a CPU halt.

Table 4-1. CPU Status and Control Lines

Machine Cycle	Status			Control		
	$IO/\overline{M}$	$S_0$	$S_1$	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch	0	1	1	0	1	1
Memory Read	0	0	1	0	1	1
Memory Write	0	1	0	1	0	1
I/O Read	1	0	1	0	1	1
I/O Write	1	1	0	1	0	1
INTR Acknowledge	1	1	1	1	1	0
Bus Idle	X	X	X	1	1	1
Halt	TS	0	0	TS	TS	1

**4-19. OPCODE FETCH TIMING.** Figure 4-4 shows the timing relationship of a typical opcode fetch machine cycle. At the beginning of  $T_1$  of every machine cycle, the CPU performs the following:

- Pulls  $IO/\overline{M}$  low to signify that the machine cycle is a memory reference operation.
- Drives status lines  $S_0$  and  $S_1$  high to identify the machine cycle as an opcode fetch.
- Places high-order bits (PCH) of program counter onto address lines  $A_8$ - $A_{15}$ . These address bits will remain true until at least  $T_4$ .
- Places low-order bits (PCL) of program counter onto address/data lines  $AD_0$ - $AD_7$ . These address bits will remain true for only one clock cycle, after which  $AD_0$ - $AD_7$  go to their high-impedance state as indicated by the dashed line in figure 4-3.
- Activates the Address Latch Enable (ALE) signal.

At the beginning of  $T_2$ , the CPU pulls the  $\overline{RD}/$  line low to enable the addressed memory device. The device will then drive the  $AD_0$ - $AD_7$  lines. After a period of time, as determined by the access time of the addressed memory device, valid data (the DCX instruction in this example) will be present on the  $D_0$ - $D_7$  lines. During  $T_3$  the CPU loads the data on the  $D_0$ - $D_7$  lines into its instruction register and drives  $\overline{RD}/$  high, disabling the addressed memory device. During  $T_4$  the CPU decodes the opcode and decides whether or not to enter  $T_5$  on the next clock cycle or start a new machine cycle and enter  $T_5$  and then  $T_6$  before beginning a new machine cycle.

Figure 4-5 is identical to figure 4-4 with one exception, which is the use of the READY input to the CPU. As shown in figure 4-5, the CPU examines the state of the

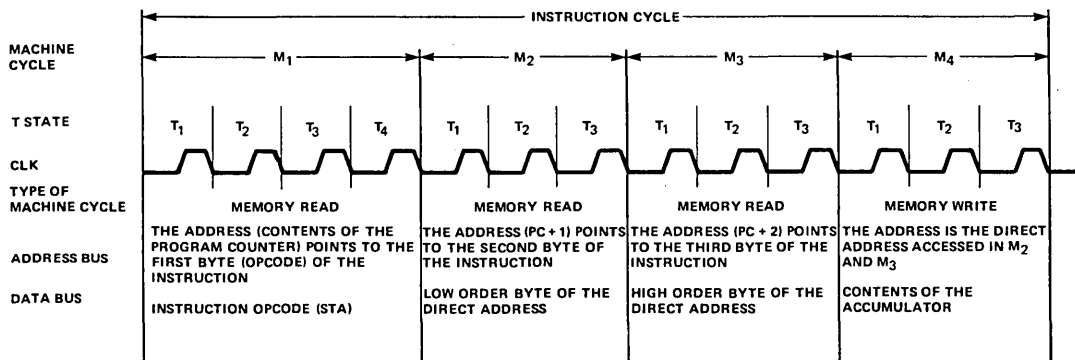


Figure 4-3. Typical CPU Instruction Cycle

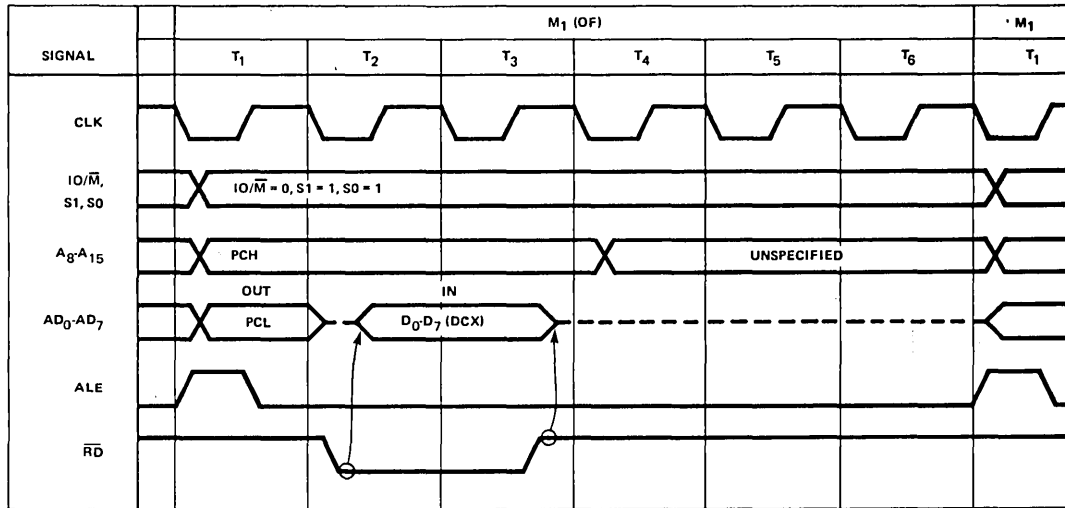


Figure 4-4. Opcode Fetch Machine Cycle

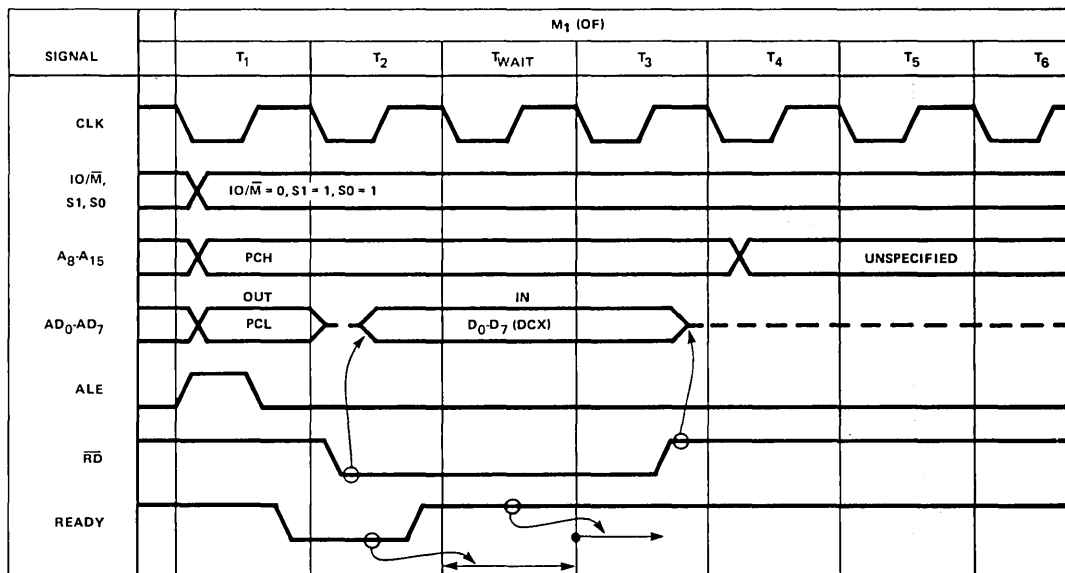


Figure 4-5. Opcode Fetch Machine Cycle (With Wait)

READY input during  $T_2$ . If the READY input is high, the CPU will proceed to  $T_3$  as shown in figure 4-4. If the READY input is low, however, the CPU will enter the  $T_{wait}$  state and stay there until READY goes high. When READY goes high, the CPU will exit the  $T_{wait}$  state and enter  $T_3$ . The external effect of using the READY input is to preserve the exact state of the CPU signals at the end of  $T_3$  for an integral number of clock periods before finishing the machine cycle. This 'stretching' of the system timing,

in effect, increases the allowable access time for memory or I/O devices. By inserting  $T_{wait}$  states, the CPU can accommodate slower memory or slower I/O devices. It should be noted, however, that access to the on-board ROM/PROM and I/O ports does not impose a  $T_{wait}$  state. However, as mentioned previously,  $T_{wait}$  states are imposed in certain instances when accessing on-board RAM;  $T_{wait}$  states are always imposed when accessing system memory or I/O devices via the Multibus.

**4-20. MEMORY READ TIMING.** Figure 4-6 shows the timing of two successive memory read machine cycles, the first without a  $T_{wait}$  state and the second with one  $T_{wait}$  state. Disregarding the states of the S0 and S1 lines, the timing during  $T_1$  through  $T_3$  is identical with the opcode fetch machine cycle shown in figure 4-4. The major difference between the opcode fetch and memory read cycles is that an opcode fetch machine cycle requires four or six T-states whereas the memory read machine cycle requires only three T-states. One minor difference between the two cycles is that the memory address used for the opcode fetch cycle is always the contents of the program counter (PC), which points to the current instruction; the address used for a memory read cycle can be one of several origins. Also, the data read from memory is placed into the appropriate register instead of the instruction register. Note that a  $T_{wait}$  state is not imposed during a read of on-board ROM/PROM.

**4-21. I/O READ TIMING.** Figure 4-6 also illustrates the timing of two successive I/O read machine cycles, the first without a  $T_{wait}$  state and the second with one  $T_{wait}$  state. With the exception of the  $IO/\overline{M}$  status signal, the timing of a memory read cycle and an I/O read cycle is identical. For an I/O read,  $IO/\overline{M}$  is driven high to identify that the current machine cycle is referencing an I/O port. One other minor exception is that the address used for an

I/O read cycle is derived from the second byte of an IN instruction; this address is duplicated onto both the A8-A15 and AD0-AD7 lines. The data read from the I/O port is always placed in the accumulator specified by the IN instruction. Note that a  $T_{wait}$  is not imposed during the access of on-board I/O devices;  $T_{wait}$  states are imposed during the access of system I/O devices via the Multibus.

**4-22. MEMORY WRITE TIMING.** Figure 4-7 shows the timing of two successive memory write machine cycles, the first without a  $T_{wait}$  state. Again, disregarding the states of the S0 and S1 lines, the timing during  $T_1$  is identical to the timing of an opcode fetch, memory read, and I/O read cycles. The difference occurs, however, at the end of  $T_1$ . For instance, in a memory read cycle the AD0-AD7 lines are disabled (high impedance) at the beginning of  $T_2$  in anticipation of the returned data. In a memory write cycle, the AD0-AD7 lines are not disabled and the data to be written into memory is placed on these lines at the beginning of  $T_2$ . The Write (WR/) line is driven low at this time to enable the addressed memory device. During  $T_2$  the READY input is checked to determine if a  $T_{wait}$  state is required. If the READY input is low,  $T_{wait}$  states are inserted until READY goes high. During  $T_3$ , the WR/ line is driven high to disable the addressed memory device and terminate the memory write operation. Note that the contents on the address and data lines do not change until the next  $T_1$  state.

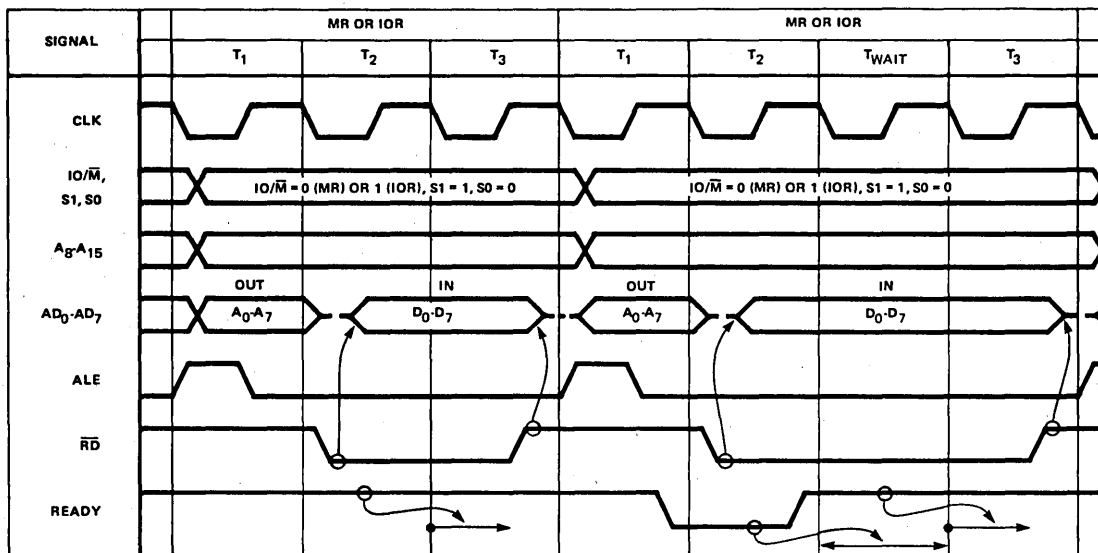


Figure 4-6. Memory Read (or I/O Read) Machine Cycles

**4-23. I/O WRITE TIMING.** Figure 4-7 also illustrates the timing of two successive I/O write machine cycles, the first without a  $T_{wait}$  state and the second with one  $T_{wait}$  state. With the exception of the  $IO/\overline{M}$  status signal, the timing of a memory write cycle and an I/O cycle are identical.

**4-24. INTERRUPT ACKNOWLEDGE TIMING.** Figure 2-8 shows the CPU timing in response to the INTR input being driven high by PIC A30 (assuming the CPU interrupt enable flip-flop has been set by a previously executed Enable Interrupt instruction). The status of the TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR inputs are sampled during CLK of the T-state immediately preceding  $T_1$  of  $M_1$ . If INTR was the only valid interrupt, the CPU clears its interrupt enable flip-flop and enters the Interrupt Acknowledge (INA) machine cycle. With two exceptions, the INA machine cycle is identical with the Opcode Fetch (OF) machine cycle. The first exception is that  $IO/\overline{M} = 1$ , which signifies that the opcode fetched will be from an I/O device. The second exception is that  $\overline{INTA}$  is asserted instead of  $\overline{RD}$ . Although the contents of CPU program counter is sent out on the address lines, the address lines are ignored.

When  $\overline{INTA}$  is asserted, the PIC provides a CALL instruction which causes the CPU to push the contents of the

program counter onto the stack before jumping to a new location. After receiving the CALL opcode, the CPU perform a second INA machine cycle ( $M_2$ ) to access the second byte of the CALL instruction from the PIC. The timing of  $M_2$  is identical with  $M_1$  except that  $M_2$  has three T-states.  $M_2$  is followed by  $M_3$  to access the third byte of the CALL instruction. When all three bytes have been received, the CPU executes the instruction. The CPU inhibits the incrementing of the program counter during the three INA cycles so that the correct program counter value can be pushed onto the stack during  $M_4$  and  $M_5$ .

During  $M_4$  and  $M_5$ , the CPU performs Memory Write (MW) machine cycles to write (push) the contents of the program counter onto the top of the stack. The CPU then places the two bytes accessed during  $M_2$  and  $M_3$  into the upper and lower bytes of the program counter. This has the same effect as jumping the execution of the program to the location specified by the CALL instruction.

After the interrupt service routine is executed, the CPU pops the stack and loads it into the program counter, and resumes system operation at the point of the interrupt. (It is the programmer's responsibility to ensure that the interrupt enable flip-flop is set before returning from the service routine.)

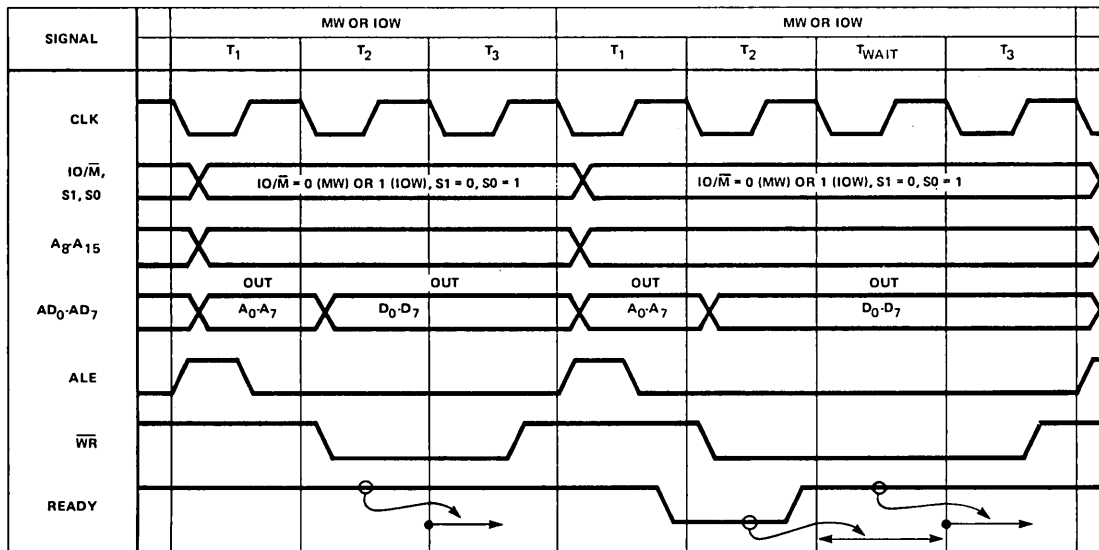
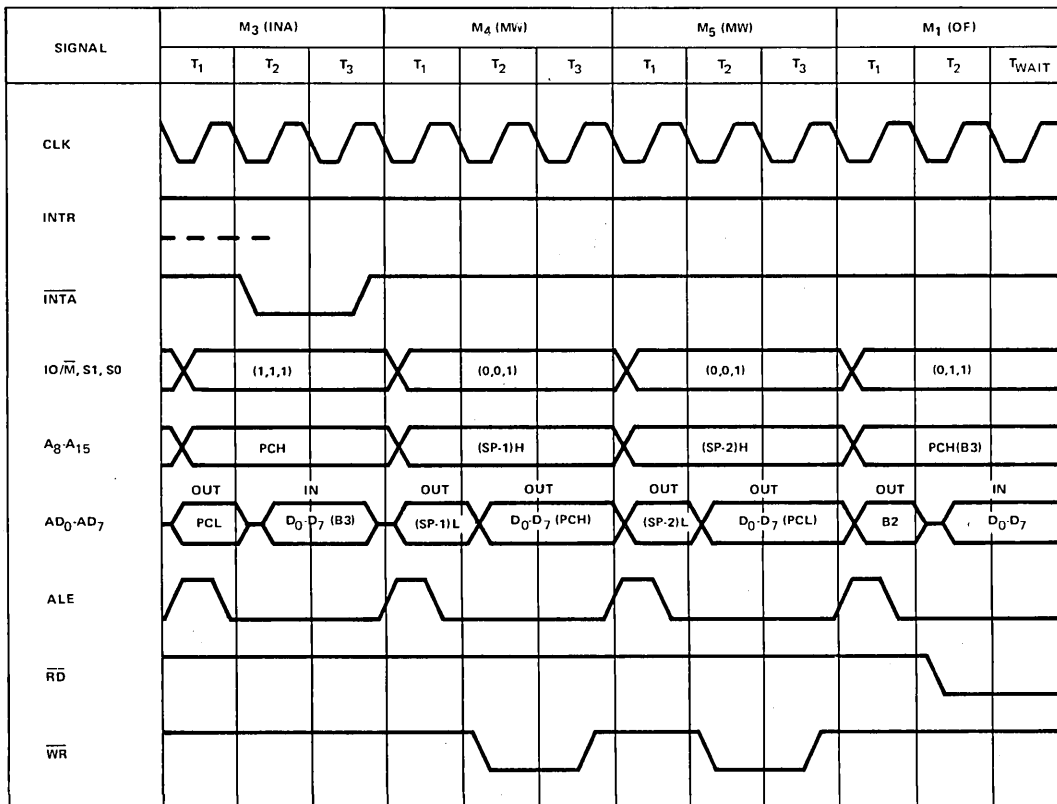
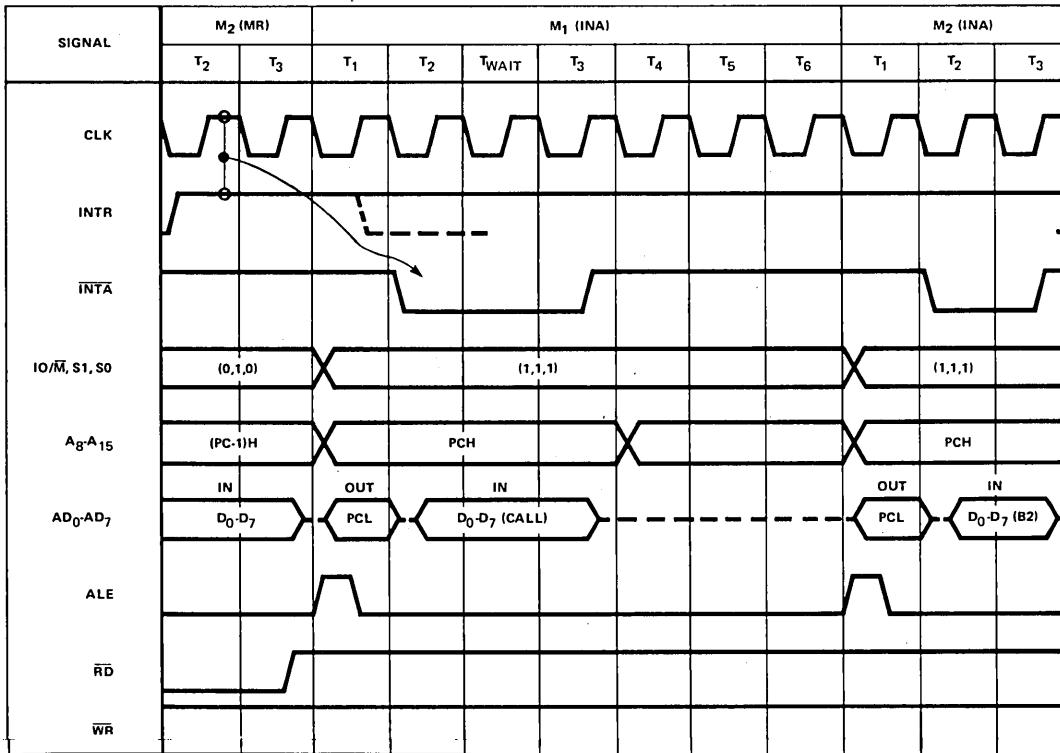


Figure 4-7. Memory Write (or I/O Write) Machine Cycles





NOTE: DISREGARD T<sub>WAIT</sub> STATES; NO T<sub>WAIT</sub> STATES IMPOSED ON 80/30 APPLICATION OF INTA.

Figure 4-8. Interrupt Acknowledge Machine Cycles

#### 4-25. ADDRESS BUS

The address bus is shown in weighted lines in figures 4-1 and 4-2. The lower eight bits (AD0-AD7) of the memory address or I/O address (depending on whether a memory reference machine cycle or an I/O reference machine cycle is in progress) are output by the CPU during the first clock cycle ( $T_1$ ). The CPU AD0-AD7 pins become the source to or input from the data bus during the second and third cycles ( $T_2$  and  $T_3$ ). The trailing edge of the Address Latch Enable (ALE) signal issued by the CPU during  $T_1$  strobes these eight address bits into Latch A23 (2ZD3). The lower eight address bits (AB0-AB7) from A23 are placed on the iSBC 80/30 address bus together with the high-order address bits (AB8-AB15). These address bits are decoded by the following:

- a. AB2-AB7 to I/O Address Decoder A50 (6ZA7).
- b. ABA-ABF to PROM Address Decode Logic (sheet 3).
- c. AB0-ABB to PROM A25/A37 (3ZA3).
- d. ABD-ABF to On-Board RAM Decoder A17 (3ZD7).

#### 4-26. BUS TIME OUT

Bus Time Out one-shot A38 (1ZC7) is retrigged by the leading edge of the ALE signal, which is asserted by the CPU during  $T_1$  of every machine cycle. If the CPU is halted, or if the CPU is hung up in a wait state for approximately 10 milliseconds, A38 times out and asserts the BTMO signal on pin 34 of the auxiliary bus (P2). If jumper 115-116 is installed, the BUS TIME OUT/ signal (A38 pin 6) drives the CPU READY line high through gates A39 and A28 and allows the CPU to exit the wait state.

The BUS TIME OUT/ signal is also applied to interrupt jumper matrix post 122 (7ZC3). If jumpered to the CPU RST 7.5 input and A38 is retrigged following a CPU hang-up state, the rising edge of BUS TIME OUT/ will cause an interrupt to report this event.

#### 4-27. DATA BUS

The CPU AD0-AD7 pins become the source or destination of the data bus DB0-DB7 during  $T_2$  and  $T_3$  clock cycles. Data can be sourced to or input from the following:

- a. Data Buffer A24 (4ZC6).
- b. Data Buffer A52 (9ZD6).

#### 4-28. READ/WRITE SIGNAL GENERATION

The COMMAND/ signal, which is used in conjunction with the various on-board read/write operations, is generated by A45-3 (2ZB4) when the CPU RD/ or WR/ signal is true. The following paragraphs describe how the various I/O and memory control signals are generated.

**4-29. I/O CONTROL SIGNALS.** The I/O control signals (2ZC2) are derived simply by gating the status of the CPU IO/M, RD/, and WR/ pins. For example, the I/O ADR signal is the buffered  $IO/\overline{M}$  pins status; the I/O WRT/ signal is the logical AND of the  $IO/\overline{M}$  and WR/ pin status.

**4-30. MEMORY CONTROL SIGNALS.** The MEM ADR, MEMRD/, and MEMWR/ signals (2ZB2, 2ZC2) are also derived simply by gating the status of the CPU  $IO/\overline{M}$ , RD/, and WR/ pins. For example, the MEM ADR signal is the buffered and inverted  $IO/\overline{M}$  pin status; the MEMWR/ signal is the logical AND of the WR/ and inverted  $IO/\overline{M}$  pin status.

The ADV MEM RD/ and ADV MEM WRT/ are derived by multiplexing the CPU S0 and S1 pin status. Refer to table 4-1. During all memory references, the CPU  $IO/\overline{M}$  status pin is low and activates the gate input to 8:4 Multiplexer A35. The select pin of A35 is controlled by the S0 pin status; when  $S0 = 0$ , the "A" inputs are selected and, when  $S0 = 1$ , the "B" inputs are selected.

For a memory read operation,  $S0 = 0$  and  $S1 = 1$  and a logic 1 appears on A35 pin 12. The trailing edge of the CPU ALE signal clocks Quad "D" Flip-Flop A34 and A34 pin 14 asserts the ADV MEM RD/ signal.

For a memory write operation,  $S0 = 1$  and  $S1 = 0$  and a logic 1 appears on A35 pin 7. The trailing edge of ALE clocks A34 and A34 pin 11 asserts the ADV MEM WRT/ signal.

For a memory fetch operation,  $S0 = 1$  and  $S1 = 1$  and, since the S1 status is applied to both the 4A and 4B inputs, the ADV MEM RD/ signal is generated as described above for the memory read operation.

When the read, write, or fetch operation is complete, the CPU RD/ or WR/ pin goes false and Flip-Flop A44 (2ZB3) is clocked and set. The output at A44-6 clears A34; the next rising edge of ALE clears A44.

#### 4-31. DUAL PORT CONTROL LOGIC

The Dual Port Control logic (figure 5-2 sheet 9) allows the on-board RAM facilities to be shared by the on-board CPU and another bus master via the Multibus. When not

acting as a bus master or when not accessing its on-board RAM, the iSBC 80/30 can act as a "slave" RAM device in a multiple bus master system. When accessing its on-board RAM, the on-board CPU has priority over any attempt to access the on-board RAM via the Multibus. In this situation, the bus access is held off until the CPU has

completed its particular read or write operation. When a bus access is in progress, the Dual Port Control Logic enters the "slave" mode and any subsequent CPU request will be held off until the slave mode is terminated. Figures 4-9 and 4-10 are timing diagrams for the Dual Port Control logic.

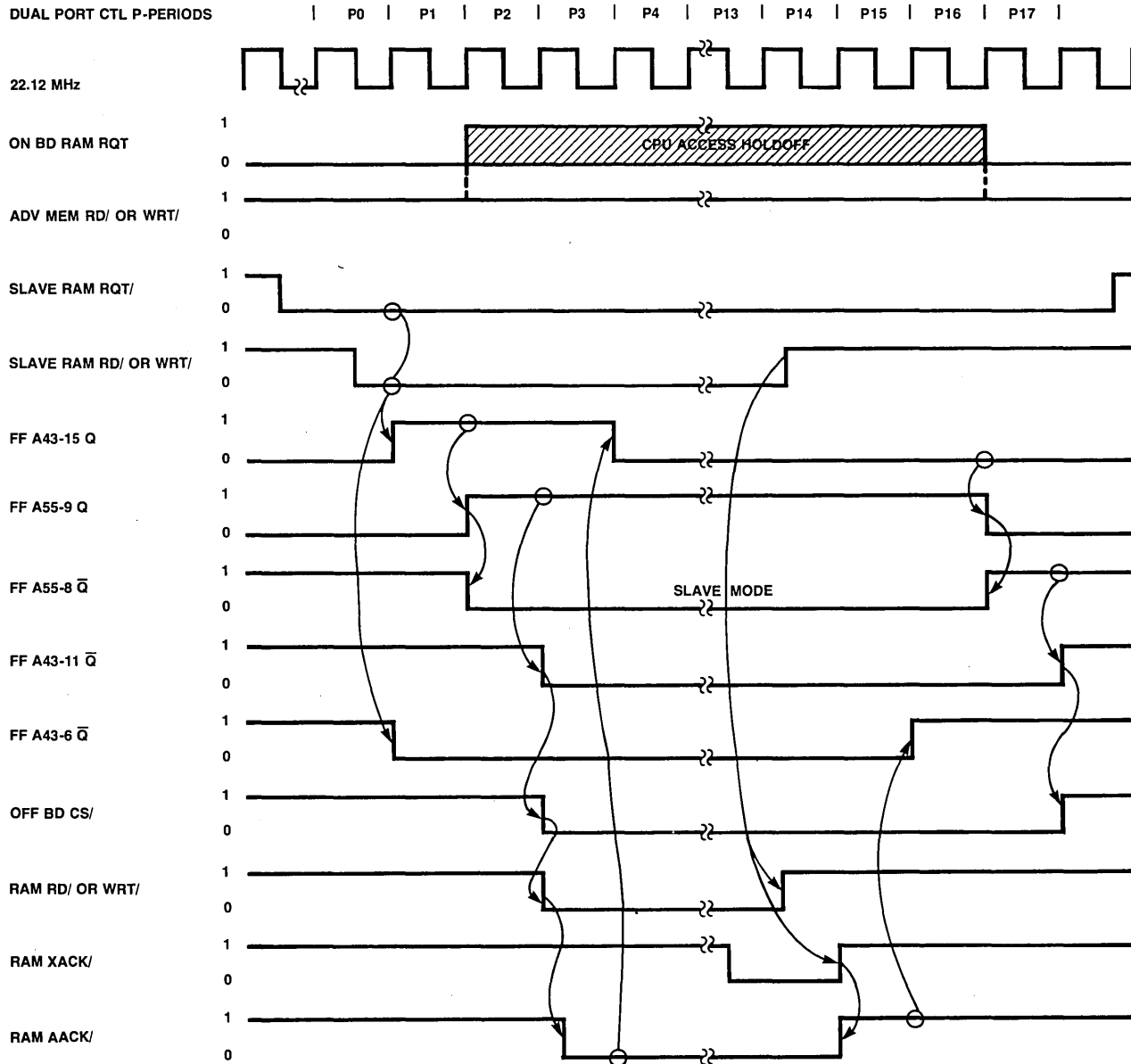
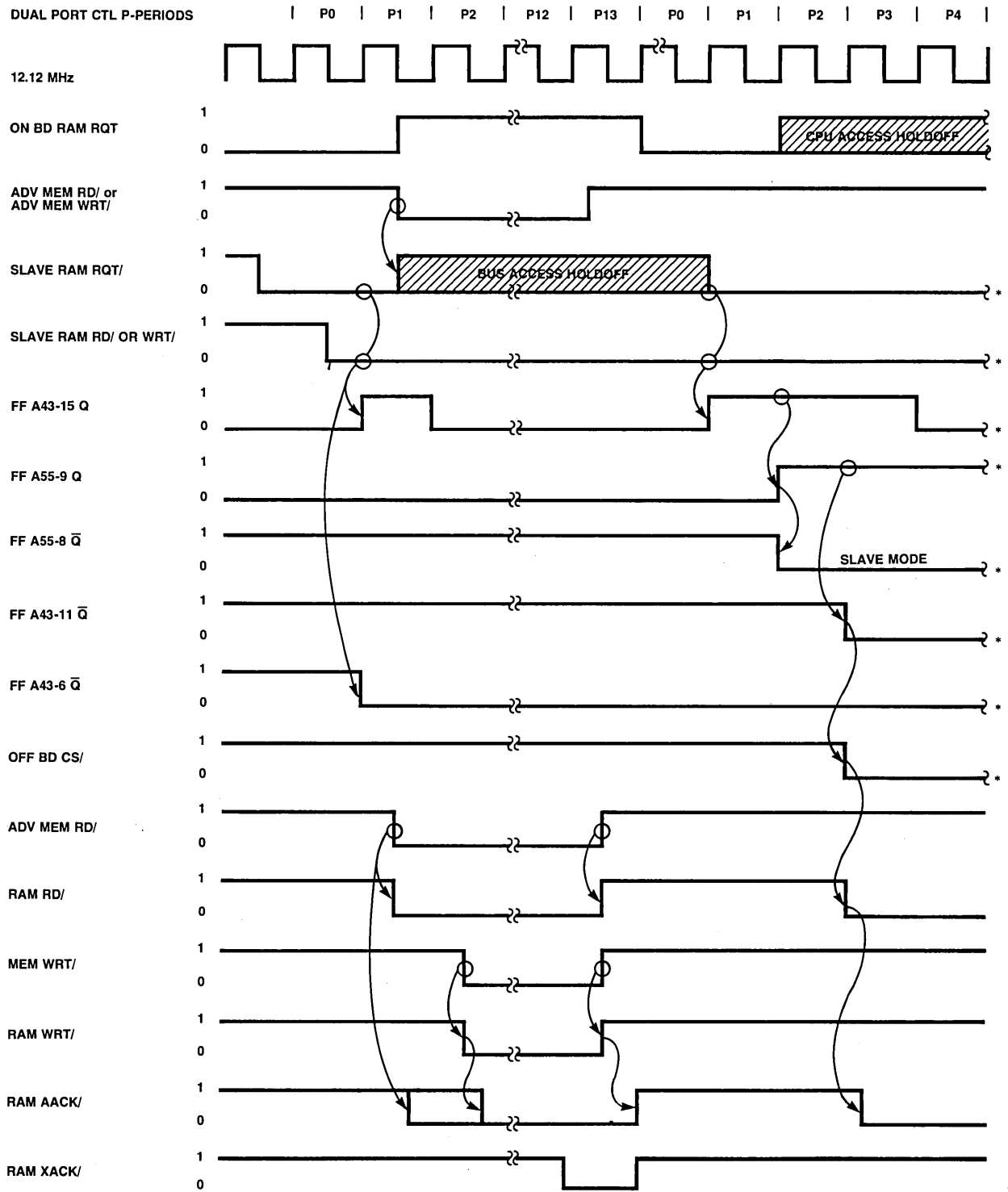


Figure 4-9. Dual Port Control Bus Access Timing With CPU Lockout



\*FOR REMAINDER OF SLAVE ACCESS TIMING, SEE FIGURE 4-9 BEGINNING WITH PERIOD P13.

Figure 4-10. Dual Port CPU Access Timing With Bus Lockout

**4-32. BUS ACCESS TIMING.** Figure 4-9 illustrates Dual Port Control timing for RAM access via the Multibus. (P-periods P0 through P17 are used only for descriptive references.) When SLAVE RAM RQT/ and SLAVE RAM RD/ or SLAVE RAM WRT/ are true, A43-15 goes high and A43-6 goes low on the next rising edge of the clock at the end of P0. (The purpose of A43-15 is to synchronize the asynchronous command and the purpose of A43-6 is to prevent a subsequent CPU request from aborting a bus access after A55-9 goes high at the end of P1).

At the end of P1, A55-9 goes high and A55-8 goes low; A55-8 asserts the SLAVE MODE signal. The outputs of A55-8 and A43-6 are ANDed to hold A55-9 in the preset (high) state. At the end of P2, A43-11 goes low and asserts the OFF BD CS/ signal, which drives the PCS/ input low to RAM Controller A66. The output of A43-11 also enables pin 1 of A60-7, A60-3, A60-5, and A60-9, to gate the RAM RD/ or RAM WRT/ signal to A66. Approximately one-half to two clock cycles later, A66 asserts the RAM AACK/ signal and, at the end of P3, A43-15 goes low.

The RAM Controller asserts RAM XACK/ during P13 and SLAVE XACK/ in driven onto the Multibus. The bus master then first raises the SLAVE RAM RD/ or SLAVE RAM WRT/ signal and then the SLAVE RAM RQT/ signal. When SLAVE RAM RD/ or WRT/ goes false, the RAM Controller raises RAM AACK/ and RAM XACK/. At the end of P15, A43-6 is clocked high. At the end of P16, A55-9 goes low and A59-8 goes high. At the end of P17, A43-11 goes high and terminates the OFF BD CS/ signal.

The foregoing discussion pertains only to the operation of the Dual Port Control for bus access of on-board RAM. The actual addressing and the transfer of data are discussed in paragraph 4-42.

**4-33. CPU ACCESS TIMING.** Figure 4-10 illustrates the Dual Port Control timing for RAM access by the on-board CPU. (P-periods P0 through P13 are used only for descriptive purposes.) To demonstrate that the CPU has priority in the access of on-board RAM, figure 4-10 shows that the SLAVE RAM RQT/ and a SLAVE RAM RD/ or SLAVE RAM WRT/ are active when the CPU access is initiated. The timing has progressed through P0 during which time A43-15 has been clocked high and A43-6 has been clocked low.

When the ON BD RAM RQT and ADV MEM RD/ are asserted, the ADV MEM RD/ signal is driven through A60-15 to assert the RAM RD/ signal. The RAM Controller then asserts RAM AACK/, which is driven through A62-8 to produce RAM QAACK/. At the end of P1, A43-15 is clocked low to effectively abort the slave access timing cycle. (For a write cycle, the MEM WR/ signal is used instead of ADV MEM WRT/.

During P12 the RAM controller asserts RAM XACK/ (not used by the CPU) and, during P13, the ADV MEM RD/ and ON BD RAM go false to complete the CPU access. On the following rising edge of the clock, the SLAVE RAM RQT/ and SLAVE RAM RD/ or SLAVE RAM WRT/ signals, which have been held off during the CPU access, set A43-15 once again to initiate the slave access timing cycle.

The foregoing discussion pertains only to the operation of the Dual Port Control for CPU access of on-board RAM. The actual addressing and the transfer of data are discussed in paragraph 4-41.

#### 4-34. MULTIBUS INTERFACE

The Multibus interface consists of Bus Controller A67 (8ZD5), bidirectional Address Bus Driver A72/A73 (8ZA2), bidirectional Data Bus Driver A74/A75 (8ZB2), and the Slave RAM Address Decode Logic (figure 5-2 sheets 3 and 5).

The Bus Controller allows the iSBC 80/30 to assume the role of a bus master and includes bus arbitration, timing, and read/write command logic. The falling edge of BCLK/ provides the timing reference, and bus arbitration begins when the Qualified Command (QCMD/) signal is asserted and the address decoders have determined that the associated Read or Write Command is not intended for on-board I/O or memory. The QCMD/ signal (1ZD8), which is used only for Multibus requests, is derived directly from the CPU Read (RD/) signal or derived by delaying the CPU Write (WT/) signal half a clock cycle. (Delaying WT/ ensures the adequate setup of the address and data to the Bus Drivers before the Write Command (IORC/ or MWTC/) is asserted.

The QCMD/ signal activates the Transfer Start Request (XSTR) input to the Bus Controller, which drives BREQ/ low and BPRO/ high. The BREQ/ output from each bus master in the system is used by the Multibus when the bus priority is resolved by a parallel priority scheme as described in paragraph 2-27. The BPRO/ output is used by the Multibus when the bus priority is resolved by a serial priority scheme as described in paragraph 2-26.

The iSBC 80/30 gains control of the Multibus when the BPRN/ input to the Bus Controller is driven low. On the next falling edge of BCLK/, the Bus Controller drives BUSY/ and ADEN/ low. The BUSY/ output indicates that the bus is in use and that the current bus master in control will not relinquish control until it raises its BUSY/ signal. The ADEN/ output, which can be thought of as a "master bus control" signal, enables Address Bus Driver A72/A73 and Data Bus Driver A74/A75. Since the board is not in the slave mode, the QSLAVE RQT/ signal is false (high) and the Address Bus Driver transmit function is selected. The steering logic composed of A39-6,

A59-6, A42-6, and A41-8 (8ZB4) decides which function (transmit or receive) to select for the Data Bus Driver. This decision is based on the mode (master or slave) that the board is in and whether the command is a read or a write. For the master mode, a Write Command selects the transmit function (DIEN is driven high); a Read Command selects the receive function (DIEN is driven low).

The BUS Controller examines its IORR, IOWR, MRDR, and MWTR inputs and drives the appropriate command line low on the Multibus. After the command is acknowledged (signified by the addressed device driving AACK/ or XACK/ low), the CPU terminates the appropriate command and tristates its address lines. This deselects the Address Bus Drivers and Data Bus Drivers and causes QCMD/ to go false (high). The Bus Controller relinquishes control of the Multibus by driving BREQ/ high and BPRO/ low and then raising BUSY/.

It should be noted that, after gaining control of the Multibus, the iSBC 80/30 can invoke an override condition to prevent losing control at a critical time. (For instance, it may be desired to execute several consecutive commands without having to contend for the bus after each command is executed.) Bus override is invoked by executing a SIM instruction with accumulator bits 6 and 7 = 1, which causes the CPU SOD line to drive the Bus Controller OVRD input high.

#### 4-35. I/O OPERATION

The following paragraphs describe on-board and system I/O operations. The actual functions performed by specific read and write commands to on-board I/O devices are described in Chapter 3.

**4-36. ON-BOARD I/O OPERATION.** Address bits AB2-AB7 are applied to I/O Address Decoder A50 and associated input chip enable gates (6Z7A). Address bits AB4-AB7 are decoded by A48-12, A65-6, and A65-3 to provide E1, E2, and E3 enable inputs to A50; when enabled, A50 decodes address bits AB2-AB4 to provide chip select inputs to the appropriate I/O device. Address bits AB2-AB7 are decoded as follows:

Bits	Addresses	Chip Select Signal
7 6 5 4 3 2		
1 1 0 1 1 0	D8-DB	8259CS/
1 1 0 1 1 1	DC-DF	8253CS/
1 1 1 0 0 1	E4-E7	8741CS/
1 1 1 0 1 0	E8-EB	8255CS/
1 1 1 0 1 1	EC-EF	8251CS/

When any one of the five outputs of A50 goes low, the I/O ADR signal from the CPU (buffered IO/M) is driven through A46-11 to develop I/O AACK/. The I/O AACK/ signal drives the CPU READY line high and, together

with the COMMAND/ signal (decoded WR or RD), enables Data Buffer A24 (4ZC6). The function of A24 is selected by the CPU S1 output; S1 = 1 for read operations and S1 = 0 for write operations. For example, data is transferred from the DIO0-DIO7 lines to the DB0-DB7 lines when S1 = 1.

After the I/O device has been selected by address bits AB2-AB7, specific functions for the chip are selected by address bits AB0-AB1. (Refer to table 3-2.)

**4-37. SYSTEM I/O OPERATION.** Address bits AB2-AB7 are decoded by I/O Address Decoder A50 as described in paragraph 4-31. If the address is not for an on-board I/O device, I/O AACK/ remains false (high) and, together with QCMD/, activates the Bus Controller XSTR input. The false SLAVE MODE/ signal from A55-9 (9ZC4) in the Dual Port Control logic enables Address Buffer A53/A54 and, together with the false I/O AACK/ signal, enables Data Buffer A52.

When the Bus Controller gains control of the Multibus as described in paragraph 4-29, it drives ADEN/ low to select Address Bus Driver A72/A73 and Data Bus Driver A74/A75. Since the board is not in the slave mode, the QSLAVE RQT/ signal is false (high) and the Address Bus Driver transmit function is selected. The transmit or receive function of Data Buffer is selected by the CPU S1 line. If the operation is a read, S1 = 1 and the receive function is enabled; for a write, S1 = 0 and the transmit function is enabled. The steering logic composed of A39-6, A58-6, A42-6, and A41-8 (8ZB4) decides which function to select for the Data Bus Driver. Since the board is in the master mode for off-board operations, a write operation selects the transmit function and a read operation selects the receive function.

After gaining control of the Multibus the Bus Controller proceeds with the control tasks for the remainder of the I/O transfer. Refer to paragraph 4-29 for a description of how the Bus Controller terminates bus control.

#### 4-38. ROM/EPROM OPERATION

The two ROM/EPROM chips are installed by the user in IC sockets A25/A37 (3ZA3). Memory addresses 0000-1FFF are reserved exclusively for ROM/EPROM; the actual occupied memory space depends on the ROM/EPROM chips as follows:

Chip Size	Chip Addresses In	
	A25	A3
1K × 8	0000-03FF	0400-07FF
2K × 8	0000-07FF	0800-0FFF
4K × 8	0000-0FFF	1000-1FFF

Address bits AB0-ABB are applied directly to the inputs of A25-A37 and address bits ABA-ABF are applied to the

ROM/EPROM chip select decoders. When address bits ABA-ABF are true for the address ranges specified above, PROM AACK/ signal is true; when the MEMRD/ signal is also asserted during a read operation, the PROM ENABLE/ and PROM AACK/ signals are true. PROM ENABLE/ turns on Data Buffer A24 (4ZC6), whose transmit function is enabled by CPU S1 = 1, and PROM AACK/ drives the CPU READY line high via A28-6 (1DZ5). The decoded output of A49-8 selects the ROM/EPROM chip in A25; the decoded output of A49-6 selects the ROM/EPROM chip in A37. When the chip is enabled, the contents of the location specified by AB0-ABB are transferred to the CPU via Data Buffer A24.

#### 4-39. RAM OPERATION

As described in paragraph 4-31, the Dual Port Control logic allows the on-board RAM facilities to be shared by the on-board CPU and by another bus master via the Multibus. The following paragraphs describe briefly the RAM Controller and the overall operation of how the RAM is addressed for read/write operation.

**4-40. RAM CONTROLLER.** All address and control inputs to the on-board RAM is supplied by RAM Controller A66 (3ZD5). The RAM Controller provides a 64-cycle RAS/CAS refresh timing cycle to dynamic RAM chips A77-A84. Default jumper 110-111 holds the REFRESH RQT signal false and the RAM Controller operates in the automatic refresh mode. In the automatic refresh mode, a read or write request can be delayed if a refresh cycle is in progress. Option jumper position 110-106 allows the REFRESH RQT signal to be controlled by the CPU READY line and thereby implement the invisible refresh mode. In the invisible refresh mode, the RAM Controller works around memory accesses by refreshing RAM during the CPU instruction decode clock cycle which follows each instruction fetch. Refresh occurs during T<sub>4</sub> but, if the access is greater than the refresh time, the RAM Controller will automatically refresh RAM. Thus, the RAM will not generate CPU wait states but it will consume more power.

The RAM Controller, when enabled with a low input to its PCS/ pin, multiplexes the address to the RAM chips. Low-order address bits A0-A6 are presented at the RAM input pins and RAS/ is driven low at the beginning of the first 8202 clock cycle after RAM RD/ or RAM WRT/ is active. High-order address bits A7-A13 are presented at the RAM input pins and CAS/ is driven low during the second clock cycle after the RAS/ address hold time is satisfied.

The RAM Controller then examines its RD/ and WRT/ inputs. If RD/ is low, the RAM Controller drives its WE/ output high to provide a Read signal to RAM; if WRT/ is low, the RAM Controller drives its WE/ output high to provide a write signal to RAM. If RAM RD/ is asserted,

Memory Data Buffer A76 is enabled and RAM data is latched into A76 when RAM XACK/ is subsequently driven low.

When the memory cycle begins, the RAM Controller drives its SACK/ output low; SACK/ is used as the RAM AACK/ signal. When the cycle is complete (i.e., data is valid), drives its XACK/ output low. The SACK/ and XACK/ outputs go high when the RD/ or WRT/ input goes high.

#### 4-41. ON BOARD READ/WRITE OPERATION.

When MEM ADR goes true, Address Decoder A17 (3ZD7) decodes address bits ABD-ABF. The decoded output from A17 goes low and (1) asserts ON BD RAM RQT/ to the Dual Port Control logic and (2) drives the RAM Controller PCS/ input low. The RAM Controller then multiplexes the address to RAM and, depending on which input command (RAMRD/ or RAMWRT/) is true, drives the WE/ output pin high or low. (The WE/ pin is driven high for a read and driven low during a write.) The SACK/ signal, which is asserted at the beginning of the memory cycle, generates the RAM AACK/ signal. When RAM AACK/ goes true, the Dual Port Control logic generates the RAM QAACK/ signal, which is qualified by being in the master mode. The XACK/ signal, which is asserted when the data is valid, generates the RAM XACK/ signal.

When RAM QAACK/ goes true, the READY input is driven high to the CPU to prevent a wait state. By the time the CPU timing has progressed to T<sub>3</sub> of the memory cycle, the RAM Controller has already asserted XACK/ and the data is valid; i.e., the data has been written into RAM from the data bus or has been read from RAM onto the data bus. The XACK/ signal, however, is not used by the CPU during on-board RAM access.

#### 4-42. BUS READ/WRITE OPERATION.

When another bus master has control of the multibus, that bus master can address the iSBC 80/30 as a slave RAM device. Assuming that the bus master has a 20-bit addressing capability, ADR0/-ADRF/ and ADR10/-ADR13/ are placed on the Multibus and then either MWTC/ or MRDC/ is asserted. Address bits ADR10/-ADR13/ are decoded by A69 (5ZC6) and ADRD/-ADRF/ are decoded by A68 (5ZB5). (A description of 16-bit and 20-bit address assignment for bus access of RAM is given in paragraphs 2-17 through 2-19.) The decoded output of A69 provides an enable input to A68. The decoded output of A68 drives the SLAVE RAM RQT/ signal low to the Dual Port Control logic.

Assuming no CPU access of RAM is in progress, the SLAVE MODE/ signal is driven low and address bits ADR0/-ADRF/ are transferred through Address Buffers

A72/A73 to the RAM Controller. When the Dual Port Control logic asserts the OFF BD CS/ signal, the RAM Controller PCS/ input is enabled and, depending on which command has been asserted (RAMRD/ or RAMWRT/), drives the WE/ input high to RAM.

The RAM Controller generates SACK/ and XACK/ as previously described. For a bus access, the SACK/ signal develops SLAVE AACK/ and XACK/ develops SLAVE XACK/. The controlling bus master then drives the

MRDC/ or MWTC/ command high and deactivates address drivers A72/A73.

#### **4-43. INTERRUPT OPERATION**

All interrupts except INTR are connected to the CPU by jumper connections. (Refer to paragraph 2-20.) Since interrupt handling is handled by the internal CPU timing described in paragraph 4-24, no further explanation is considered necessary.



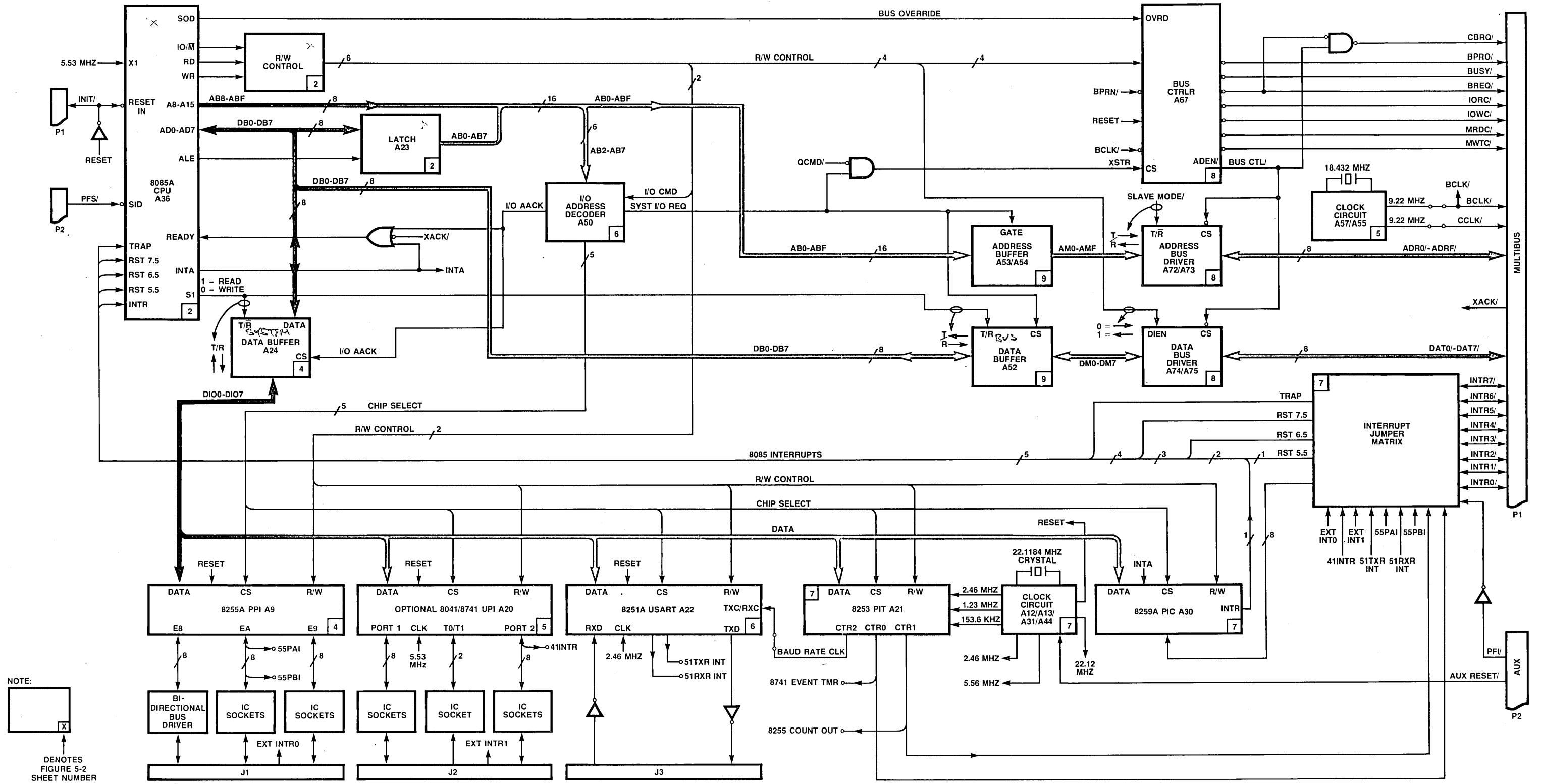


Figure 4-1. iSBC 80/30 Input/Output and Interrupt Block Diagram

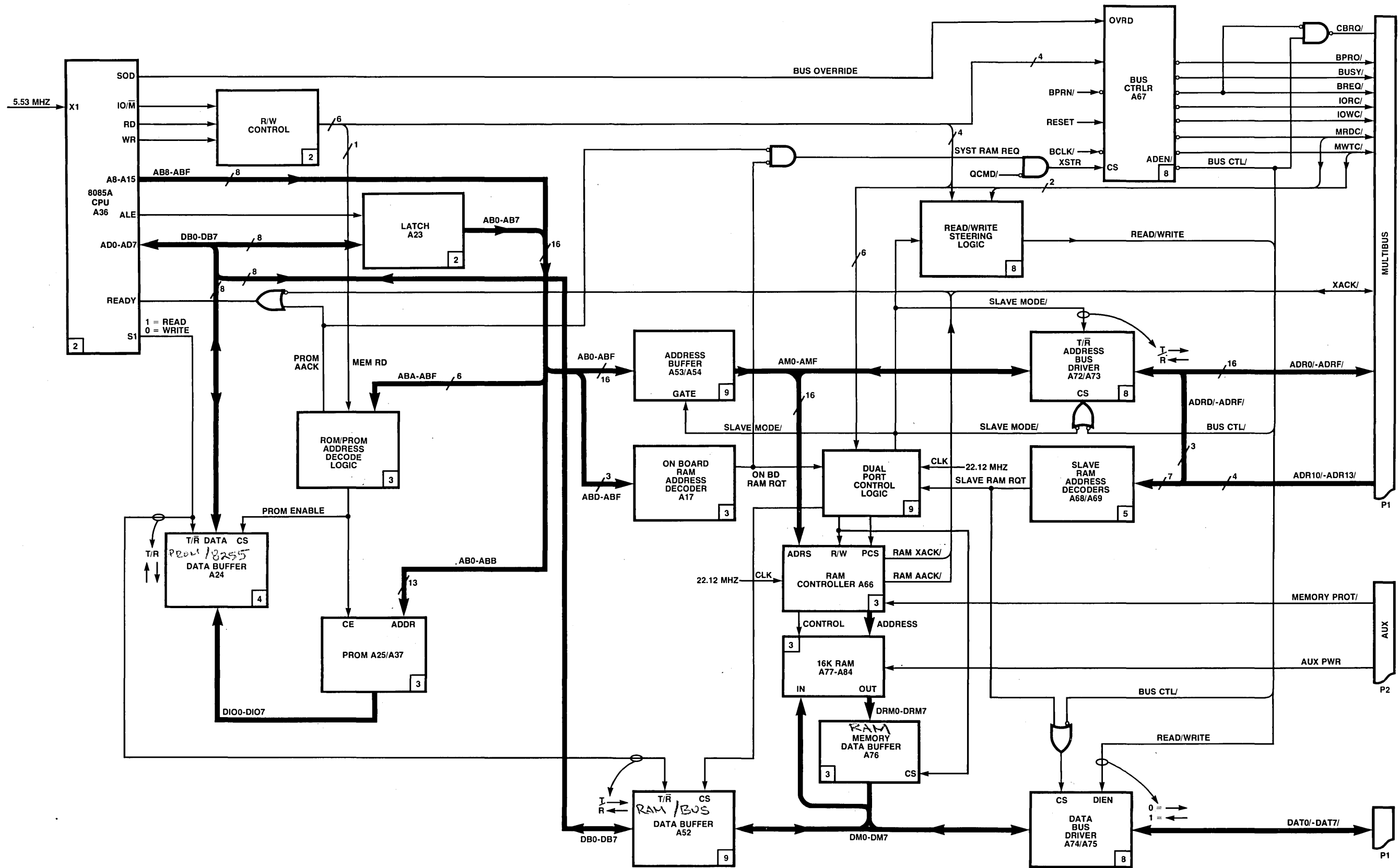


Figure 4-2. iSBC 80/30 PROM and Dual Port RAM Block Diagram



# CHAPTER 5 SERVICE INFORMATION

## 5-1. INTRODUCTION

This chapter provides a list of replaceable parts, service diagrams, and service and repair assistance instructions for the iSBC 80/30 Single Board Computer.

## 5-2. REPLACEABLE PARTS

Table 5-1 provides a list of replaceable parts for the iSBC 80/30. Table 5-2 identifies and locates the manufacturers specified in the MFR CODE column in table 5-1. Intel parts that are available on the open market are listed in the MFR CODE column as "COML"; every effort should be made to procure these parts from a local (commercial) distributor.

## 5-3. SERVICE DIAGRAMS

The iSBC 80/30 parts location diagram and schematic diagram are provided in figures 5-1 and 5-2, respectively. On the schematic diagram, a signal mnemonic that ends with a slash (e.g., IOWC/) is active low. Conversely, a signal mnemonic without a slash (e.g., BTMO) is active high.

## 5-4. SERVICE AND REPAIR ASSISTANCE

United States customers can obtain service and repair assistance from Intel by contacting the MCD Technical Support Center in Santa Clara, California, at one of the following numbers:

Telephone:

From Alaska or Hawaii call —

(408) 987-8080

From locations within California call toll free —

(800) 672-3507

From all other U.S. locations call toll free —

(800) 538-8014

TWX: 910-338-0026

TELEX: 34-6372

Always contact the MCD Technical Support Center before returning a product to Intel for service or repair. You will be given a "Repair Authorization Number", shipping instructions, and other important information which will help Intel provide you with fast, efficient service. If the product is being returned because of damage sustained during shipment from Intel, or if the product is out of warranty, a purchase order is necessary in order for the MCD Technical Support Center to initiate the repair.

In preparing the product for shipment to the MCD Technical Support Center, use the original factory packaging material, if available. If the original packaging is not available, wrap the product in a cushioning material such as Air Cap TH-240 (or equivalent) manufactured by the Sealed Air Corporation, Hawthorne, N.J., and enclose in a heavy-duty corrugated shipping carton. Seal the carton securely, mark it "FRAGILE" to ensure careful handling, and ship it to the address specified by MCD Technical Support Center personnel.

### NOTE

Customers outside of the United States should contact their sales source (Intel Sales Office or Authorized Intel Distributor) for directions on obtaining service or repair assistance.

Table 5-1. Replaceable Parts

Reference Designation	Description	Mfr. Part No.	Mfr. Code	Qty.
A1,2,74,75	IC, 8226, Bidirectional Data Buffer	8266	COML	4
A3-11	IC, 7400, Quad 2-Input Positive NAND	SN7400	TI	9
A12	IC, 74163, Synchronous 4-Bit Counter	SN74163	TI	1
A13,57	IC, 8224, System Clock Generator	8224	COML	2
A14	IC, 1488, Quad Line Driver	LM1488	NAT	1
A15,16	IC, 1489, Quad Line Driver	LM1489	NAT	2
A17,50	IC, 74LS138, 3-to-8 Decoder//Multiplexer	SN74LS138	TI	2
A18	IC, 74LS11, Triple 3-Input Positive NAND	SN74LS11	TI	1
A19	IC, 8255A, Programmable Peripheral Interface	8255A	COML	1
A21	IC, 8253, Programmable Interval Timer	8253	COML	1
A22	IC, 8251A, Serial I/O Interface (USART)	8251A	COML	1
A23	IC, 74LS373, Octal D-Type Latches	SN74LS373	TI	1
A24,52,72,73	IC, DP8304, Bidirectional Data Buffer	DP8304	COML	4
A26	IC, 74S03, Quad 2-Input Positive NAND	SN74S03	TI	1
A27,32	IC, 74LS32, Quad 2-Input OR	SN74LS32	TI	2

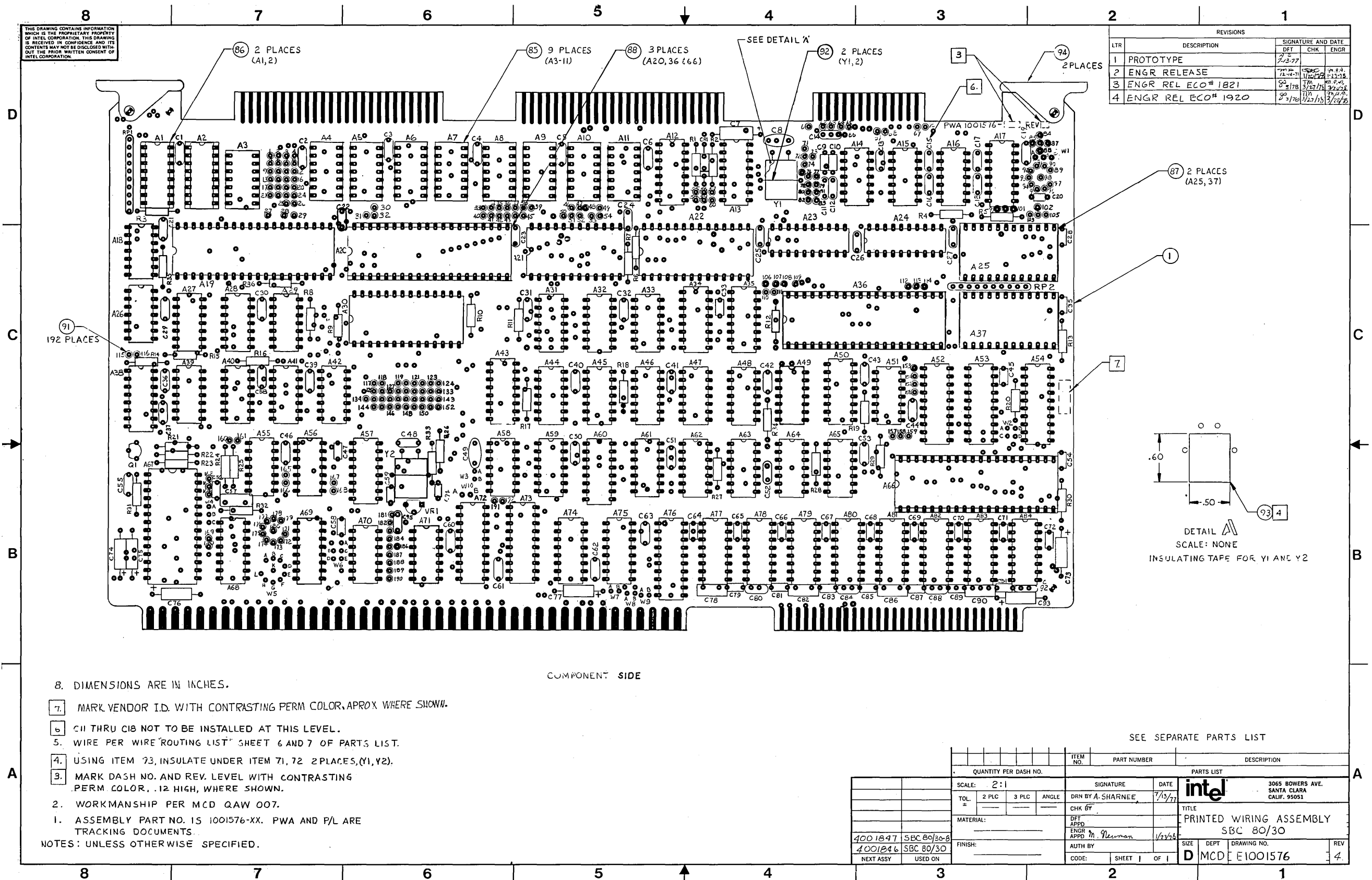
Table 5-1. Replaceable Parts (Continued)

Reference Designation	Description	Mfr. Part No.	Mfr. Code	Qty.
A28	IC, 74S20, Dual Positive NAND	SN74S20	TI	1
A29	IC, 7438, Quad 2-Input Positive NAND	SN7438	TI	1
A30	IC, 8259A, Programmable Interrupt Controller	8259A	COML	1
A31	IC, 74LS74, Dual D-Type Flip-Flops	SN74LS74	TI	1
A33,40,51,63	IC, 74S04, Hex Inverters	SN74S04	TI	4
A34,43	IC, 74S175, Quad D-Type Flip-Flops	SN74S175	TI	2
A35	IC, 74LS157, Data Selector/Multiplexer	SN74LS157	TI	1
A36	IC, 8085A Central Processor Unit	8085A	COML	1
A38	IC, 9602, One-Shot Multivibrator	9602	NAT	1
A39,45	IC, 74LS08, Quad 2-Input Positive AND	SN74LS08	TI	2
A41	IC, 74S08, Quad 2-Input Positive AND	SN74S08	TI	1
A42,62	IC, 74S32, Quad 2-Input OR	SN74S32	TI	2
A44,55	IC, 74S74, Dual D-Type Flip-Flops	SN74S74	TI	2
A46,58,61,64	IC, 74S00, Quad 2-Input Positive NAND	SN74S00	TI	4
A47	IC, 74S11, Triple 3-Input Positive NAND	SN74S11	TI	1
A48	IC, 74LS27, Triple 3-Input Positive NOR	SN74LS27	TI	1
A49	IC, 74S10, Triple 3-Input Positive NAND	SN74S10	TI	1
A53,54	IC, 74LS240, Octal Buffer/Line Driver	SN74LS240	TI	2
A56	IC, 74S140, Dual 4-Input Positive NAND	SN74S140	TI	1
A59	IC, 74LS02, Quad 2-Input Positive NOR	SN74LS02	TI	1
A60	IC, 8097, Three-State Data Buffer	8097	COML	1
A65	IC, 74LS00, Quad 2-Input Positive NAND	SN74LS00	TI	1
A66	IC, 8202, RAM Controller	8202	COML	1
A67	IC, Bus Controller	OBD	INTEL	1
A68,69	IC, 3205, 1-of-8 Binary Decoder	3205	COML	2
A70,71	IC, 74LS04, Hex Inverters	SN74LS04	TI	2
A76	IC, 74S373, Octal D-Type Latches	SN74S373	TI	1
A77-84	IC, 2117, Random Access Memory	2117	COML	8
C1-6,19,21,29,30,31,32 36-44,46,48,50-53, 55,60-63	Capacitor, Ceramic, 0.01 $\mu$ F, 25V, +80%, -20%	OBD	COML	32
C7	Capacitor, tant, 10 $\mu$ F, 20V, 10%	OBD	COML	1
C8,49	Capacitor, mono, 10pF, 500V, 5%	OBD	COML	1
C9,10,22-28,33,35,45, 47,54,56,58,59,64-72, 79,81,83,85,87,89, 91,94	Capacitor, mono, 0.1 $\mu$ F, 50V, +80%, -20%	OBD	COML	35
C57	Capacitor, mono, 220pF, 500V, 5%	OBD	COML	1
C73-77,93	Capacitor, tant, 22 $\mu$ F, 15V, 10%	OBD	COML	6
C78,82,86,90	Capacitor, mono, 1.0 $\mu$ F, 50V, 10%	OBD	COML	4
C80,84,88,92	Capacitor, mono, 0.01 $\mu$ F, 50V, +80%, -20%	OBD	COML	4
C95	Capacitor, mono, 33 $\mu$ F, 50V	OBD	COML	1
CR1	Diode, 1N4002	1N4002	TI	1
Q1	Transistor, 2N3904	2N3904	TI	1
R1,3-5,10,11,14,17-20, 23,25-29,32,33	Resistor, comp, 1K ohm, 1/4W 5%	OBD	COML	19
R2	Resistor, comp, 100K ohm, 1/4W 5%	OBD	COML	1
R6-8,15,16,30,34,35	Resistor, comp, 10K ohm, 1/4W 5%	OBD	COML	8
R9,12,13	Resistor, comp, 620K ohm, 1/4W 5%	OBD	COML	3
R13	Resistor, comp, 5.1K ohm, 1/4W 5%	OBD	COML	1
R21	Resistor, comp, 330K ohm, 1/4W 5%	OBD	COML	1
R22	Resistor, comp, 33K ohm, 1/4W 5%	OBD	COML	1
R24	Resistor, comp, 270K ohm, 1/4W 5%	OBD	COML	1
R31	Resistor, comp, 2.2K ohm, 1/4W 5%	OBD	COML	1
RP1,2	Resistor, package, 1K ohm, 10-pin	OBD	COML	2
XA1,2	Socket, IC, 16-pin	C-93-16-02	TI	2
XA3-11	Socket, IC, 14-pin	C-93-14-02	TI	9
XA20,36,66	Socket, IC, 40-pin	540-A367D	AUG	1
XA25,37	Socket, IC, 24-pin	C-93-24-02	TI	2
Y1	Crystal, 22.1184-MHz	HW3	CTS	1
Y2	Crystal, 18.432-MHz	MP184	CTS	1
	Post, Wire Wrap	85931-6	AMP	192
	Extractor, Card	S-203	SCA	2

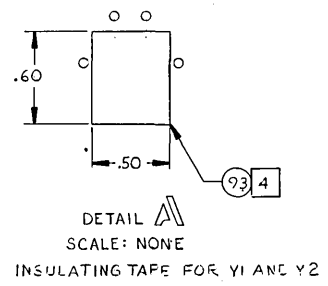
**Table 5-2. List of Manufacturers' Codes**

<b>MFR. CODE</b>	<b>MANUFACTURER</b>	<b>ADDRESS</b>	<b>MFR. CODE</b>	<b>MANUFACTURER</b>	<b>ADDRESS</b>
AMP	AMP, Inc.	Harrisburg, PA	NAT	National Semiconductor	Santa Clara, CA
AUG	Augat, Inc.	Attleboro, MA	SCA	Scanbe, Inc.	El Monte, CA
CTS	CTS Corp.	Elkhart, IN	TI	Texas Instruments	Dallas, TX
			OBD	Order by Description; available from any commercial source	





REVISIONS			
LTR	DESCRIPTION	SIGNATURE AND DATE	
		DFT	ENGR
1	PROTOTYPE		
2	ENGR RELEASE		
3	ENGR REL ECO# 1821		
4	ENGR REL ECO# 1920		



8. DIMENSIONS ARE IN INCHES.
7. MARK VENDOR ID WITH CONTRASTING PERM COLOR, APPROX WHERE SHOWN.
6. C11 THRU C18 NOT TO BE INSTALLED AT THIS LEVEL.
5. WIRE PER WIRE "ROUTING LIST" SHEET 6 AND 7 OF PARTS LIST.
4. USING ITEM 73, INSULATE UNDER ITEM 71, 72 2 PLACES, (Y1, Y2).
3. MARK DASH NO. AND REV. LEVEL WITH CONTRASTING PERM COLOR, .12 HIGH, WHERE SHOWN.
2. WORKMANSHIP PER MCD QAW 007.
1. ASSEMBLY PART NO. IS 1001576-XX. PWA AND P/L ARE TRACKING DOCUMENTS.
- NOTES: UNLESS OTHERWISE SPECIFIED.

SEE SEPARATE PARTS LIST

ITEM NO.	PART NUMBER	DESCRIPTION
4001847	SBC 80/30-8	
4001846	SBC 80/30	

SCALE:	2:1	SIGNATURE	DATE
TOL.	2 PLC 3 PLC ANGLE	DRN BY A. SHARNEE	7/13/77
MATERIAL:		CHK BY	
FINISH:		DFT APPD	
NEXT ASSY	USED ON	ENGR APPD M. Neuman	1/23/76
CODE:	SHEET 1 OF 1	AUTH BY	

SIZE	DEPT	DRAWING NO.	REV
D	MCD	E1001576	4

Figure 5-1. iSBC 80/30 Parts Location Diagram

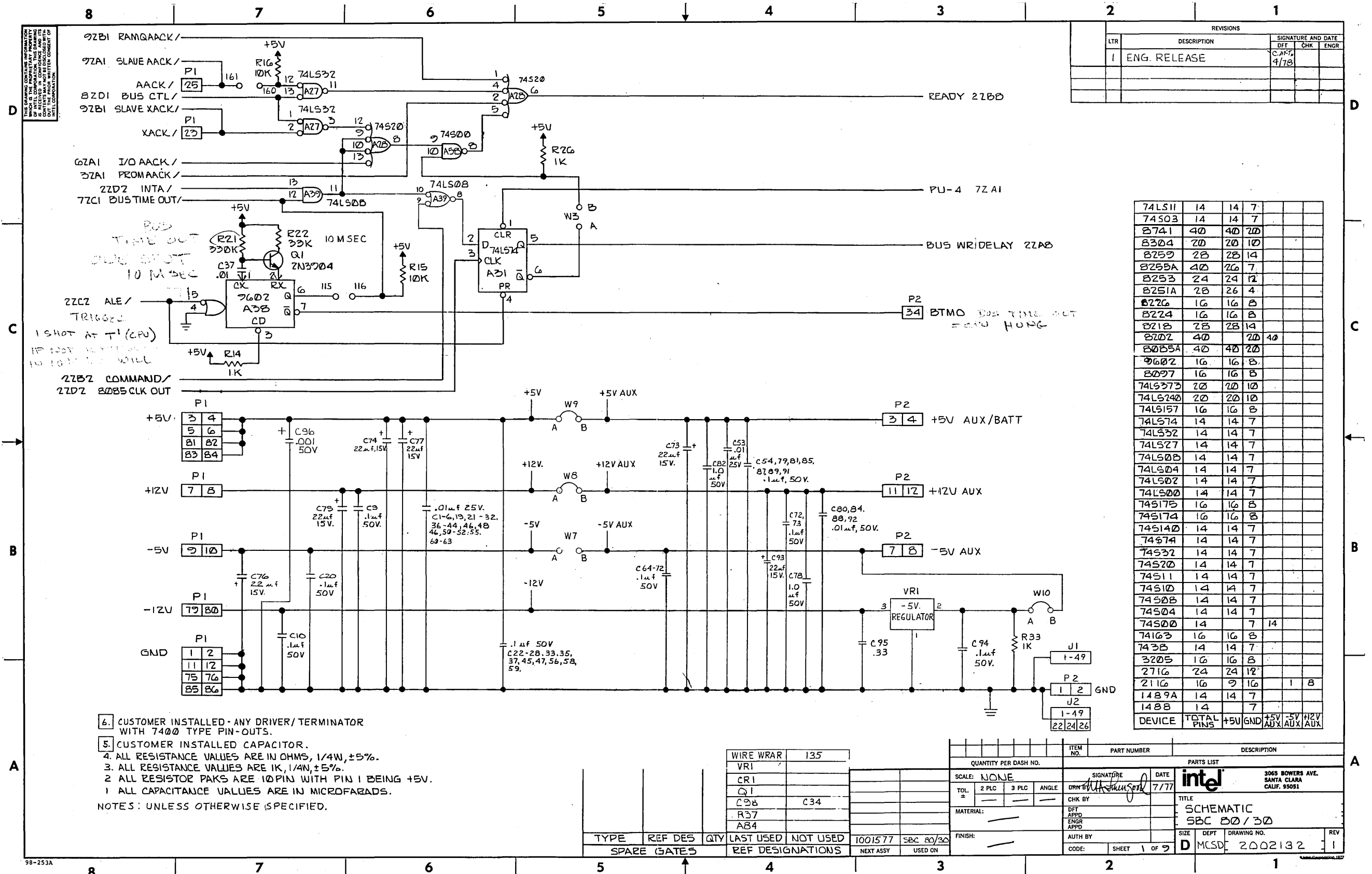


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 1 of 9)

POWER





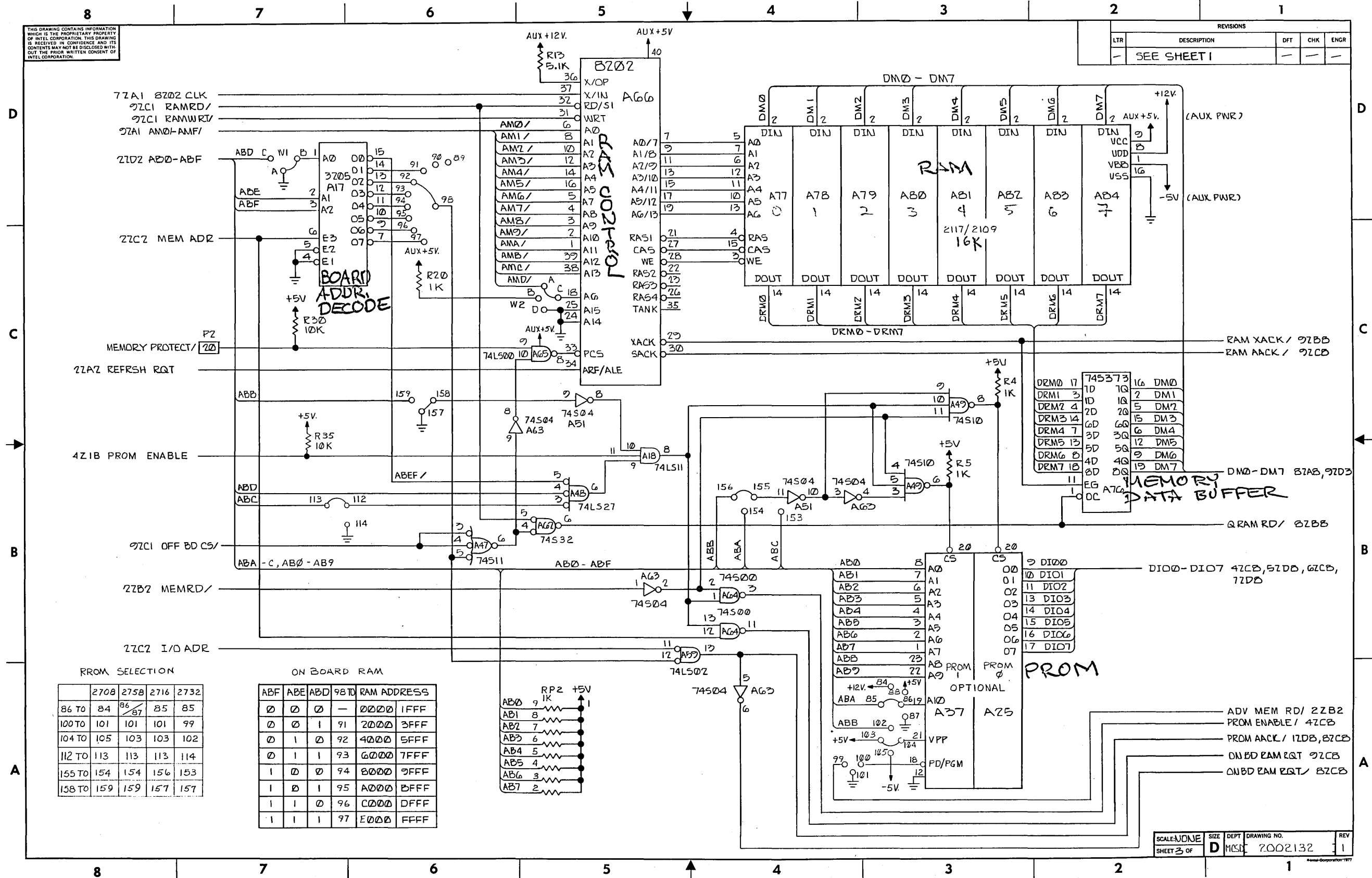


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 3 of 9)

MEMORY

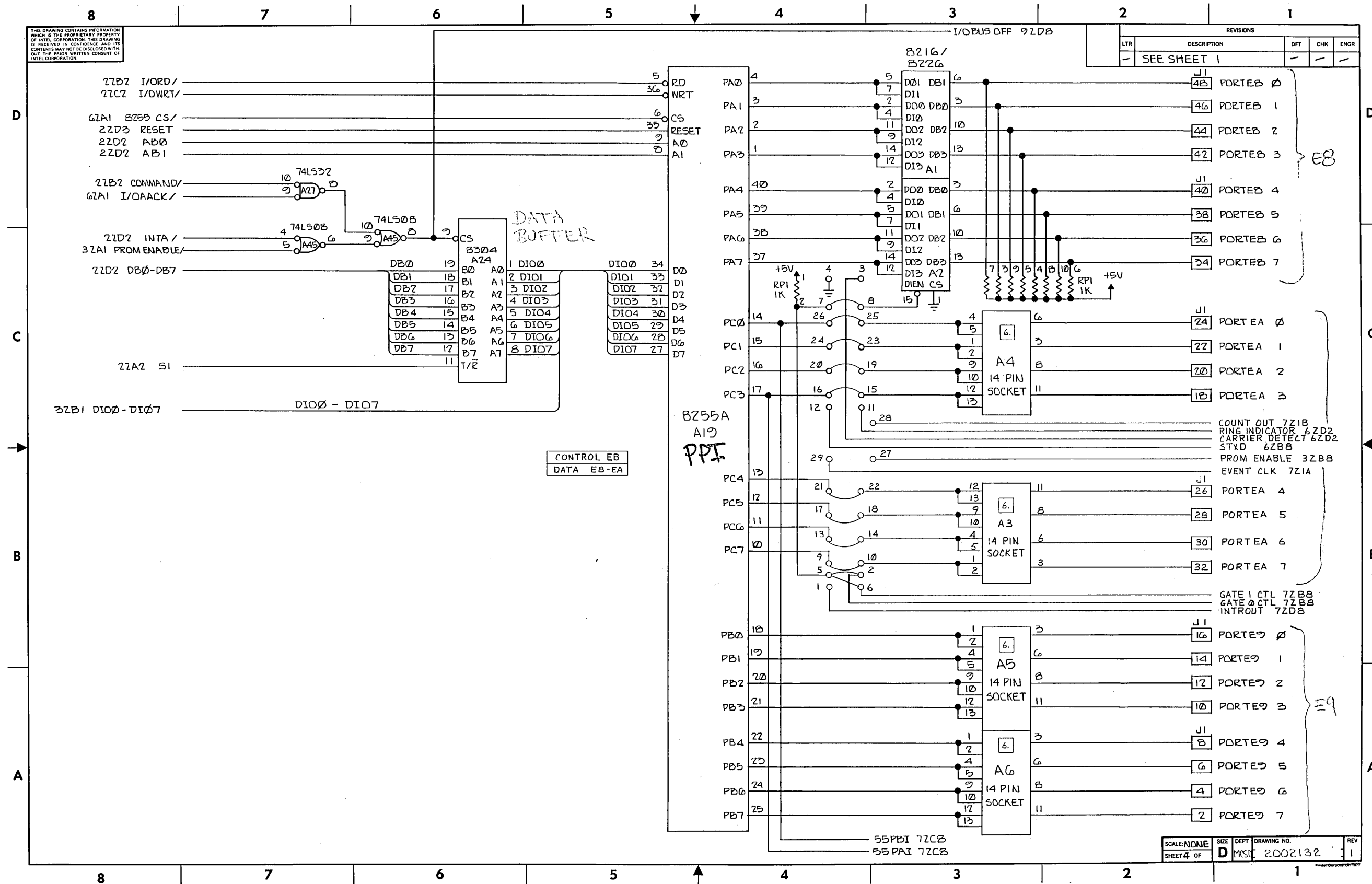


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 4 of 9)

PPI

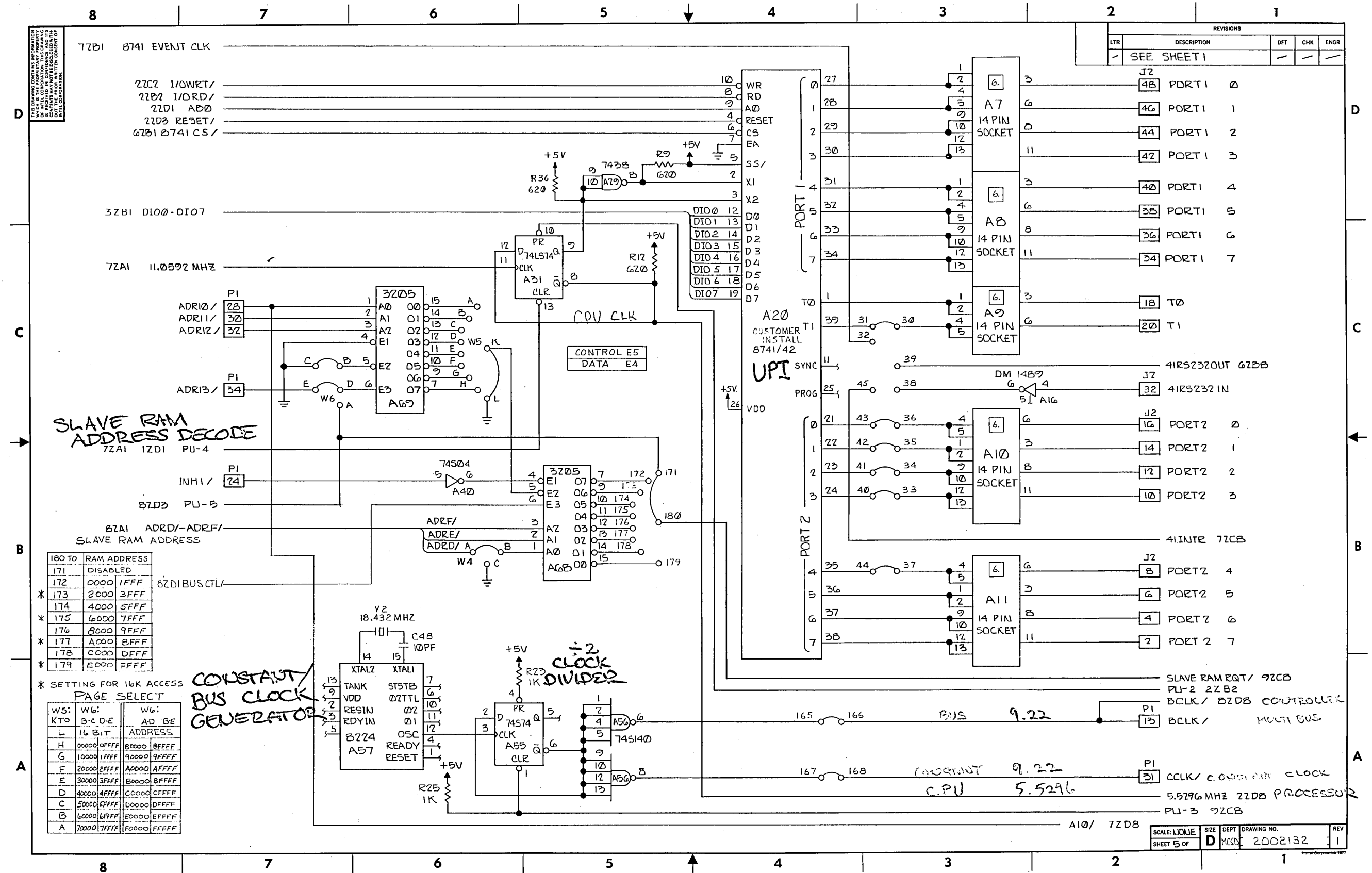


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 5 of 9)

UPT  
CPU/UPT BUS & CONSTANT CLOCKS  
SLAVE RAM ADDR. DECODES-15/5-16

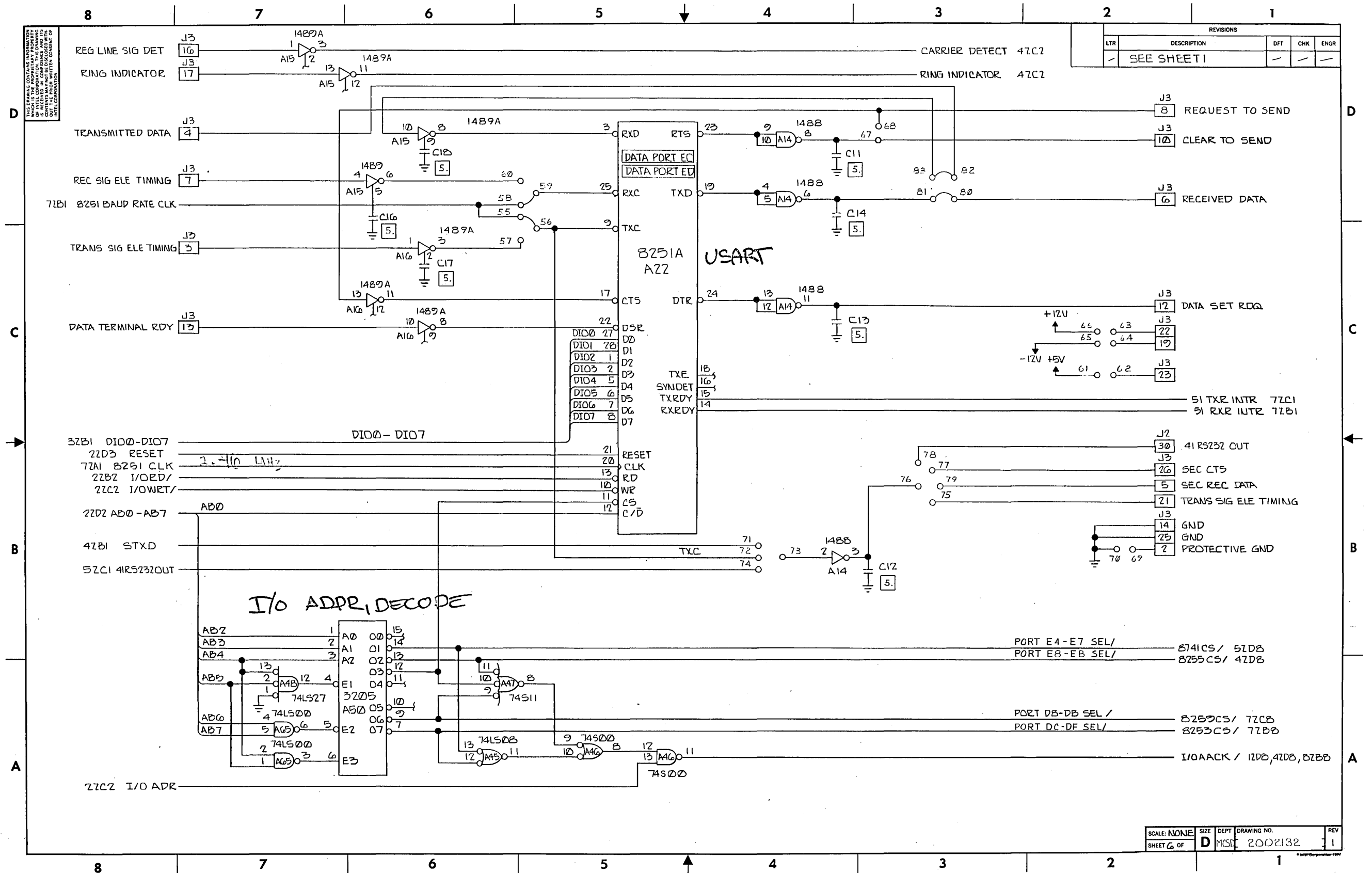


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 6 of 9)

USART

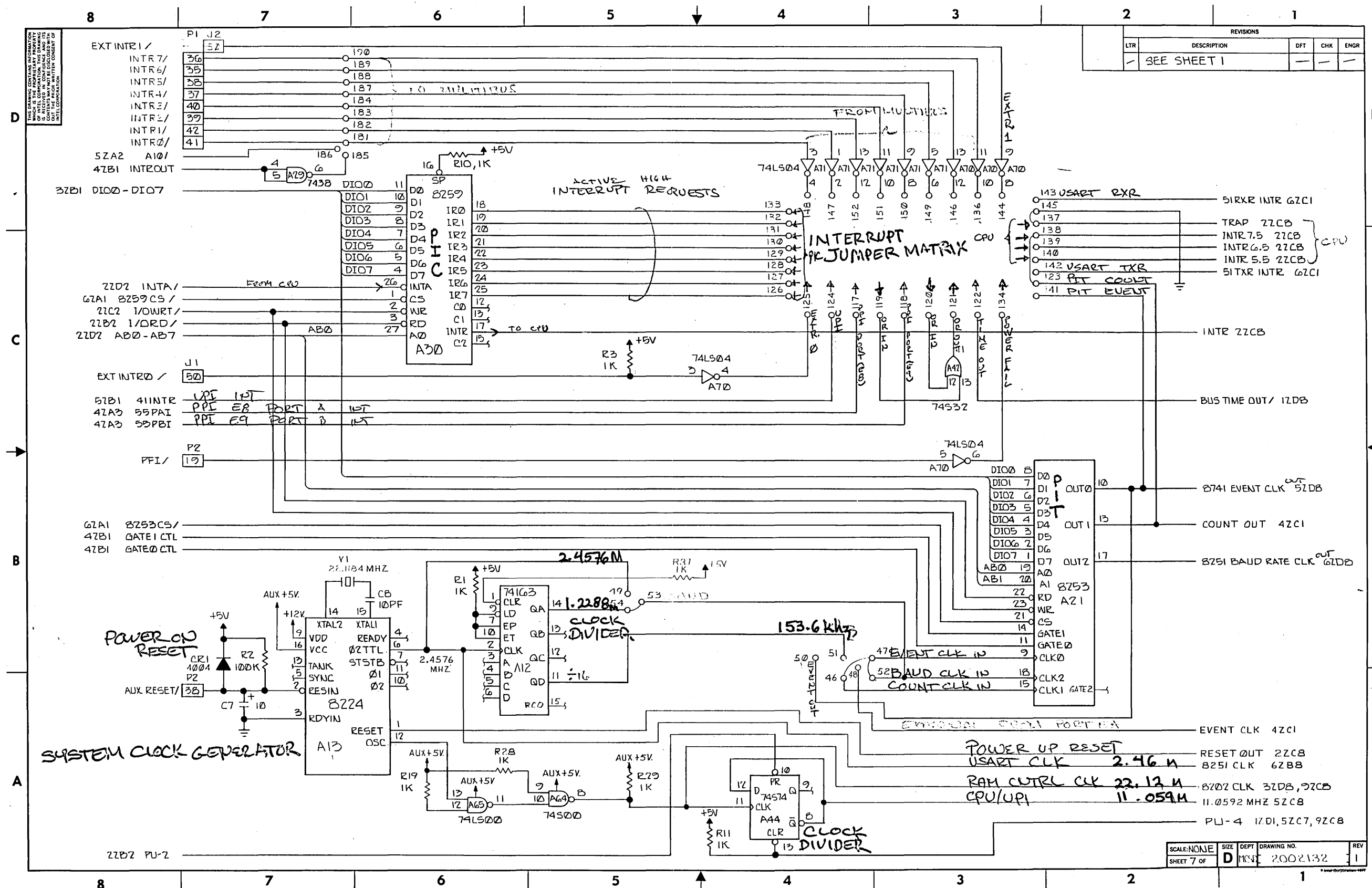


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 7 of 9)

INTERRUPT / SYSTEM CLOCK

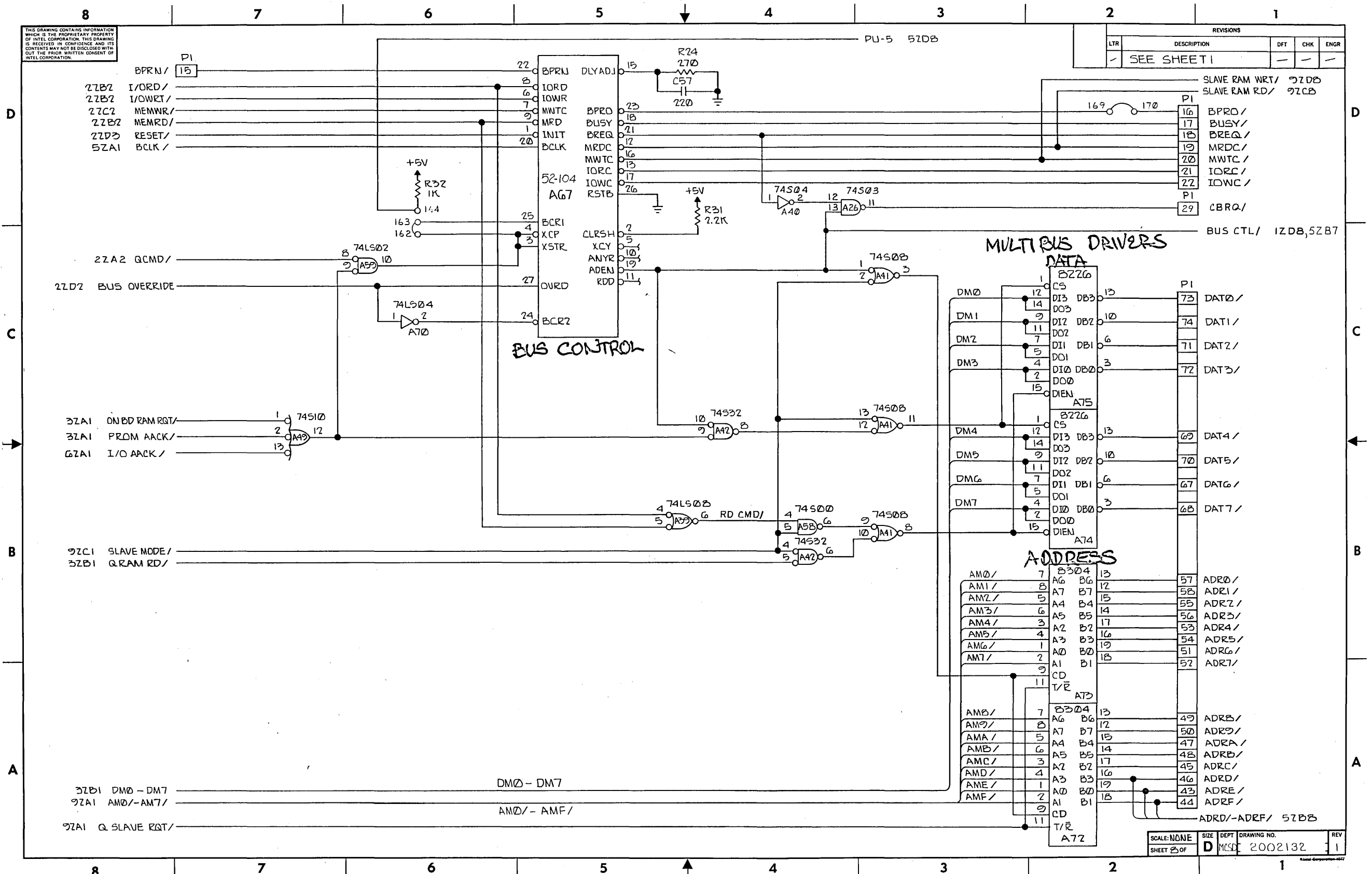


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 8 of 9)

BUS CONTROL/BUS DRIVERS

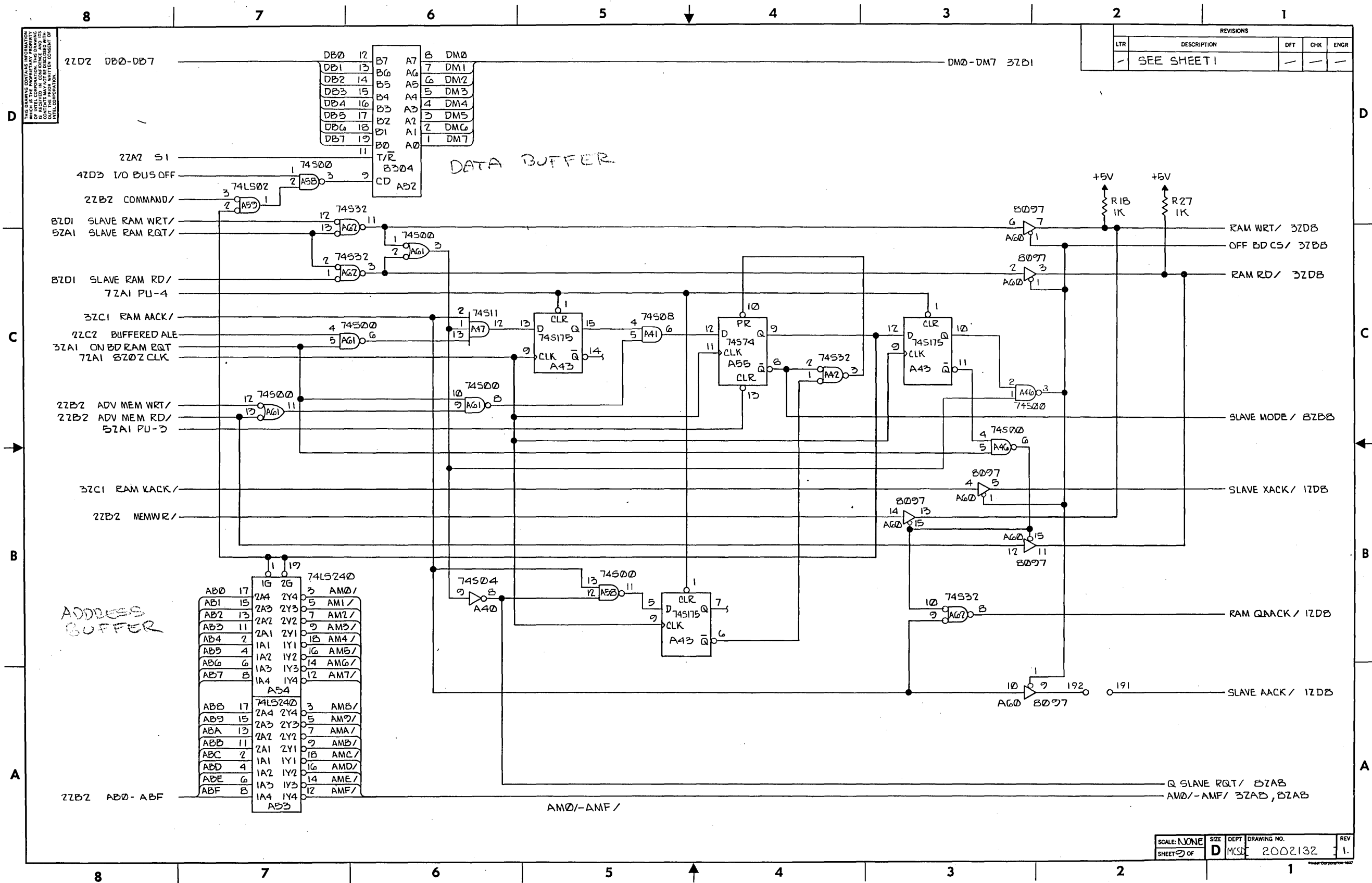
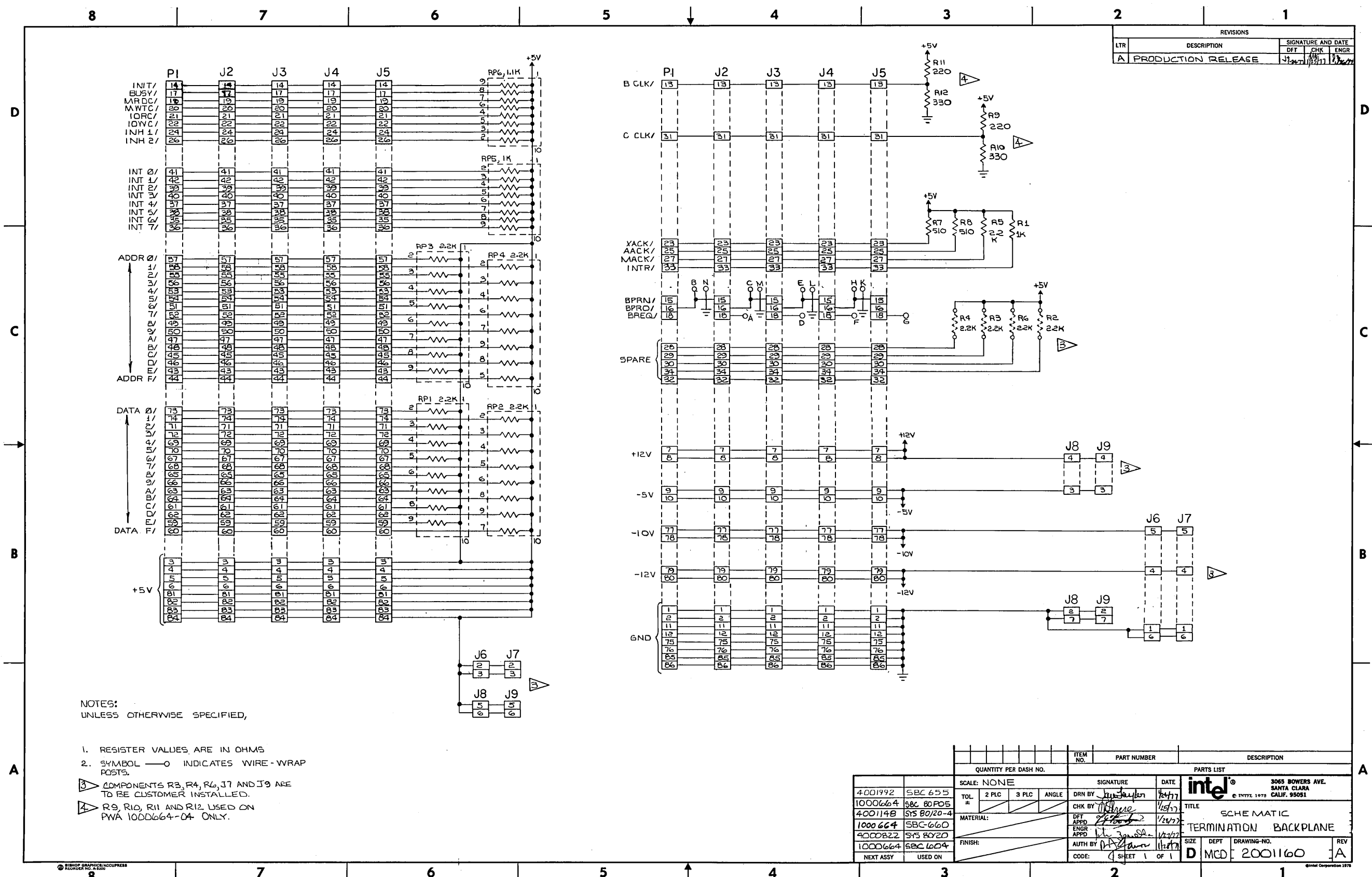


Figure 5-2. iSBC 80/30 Schematic Diagram (Sheet 9 of 9)

DUAL PORT





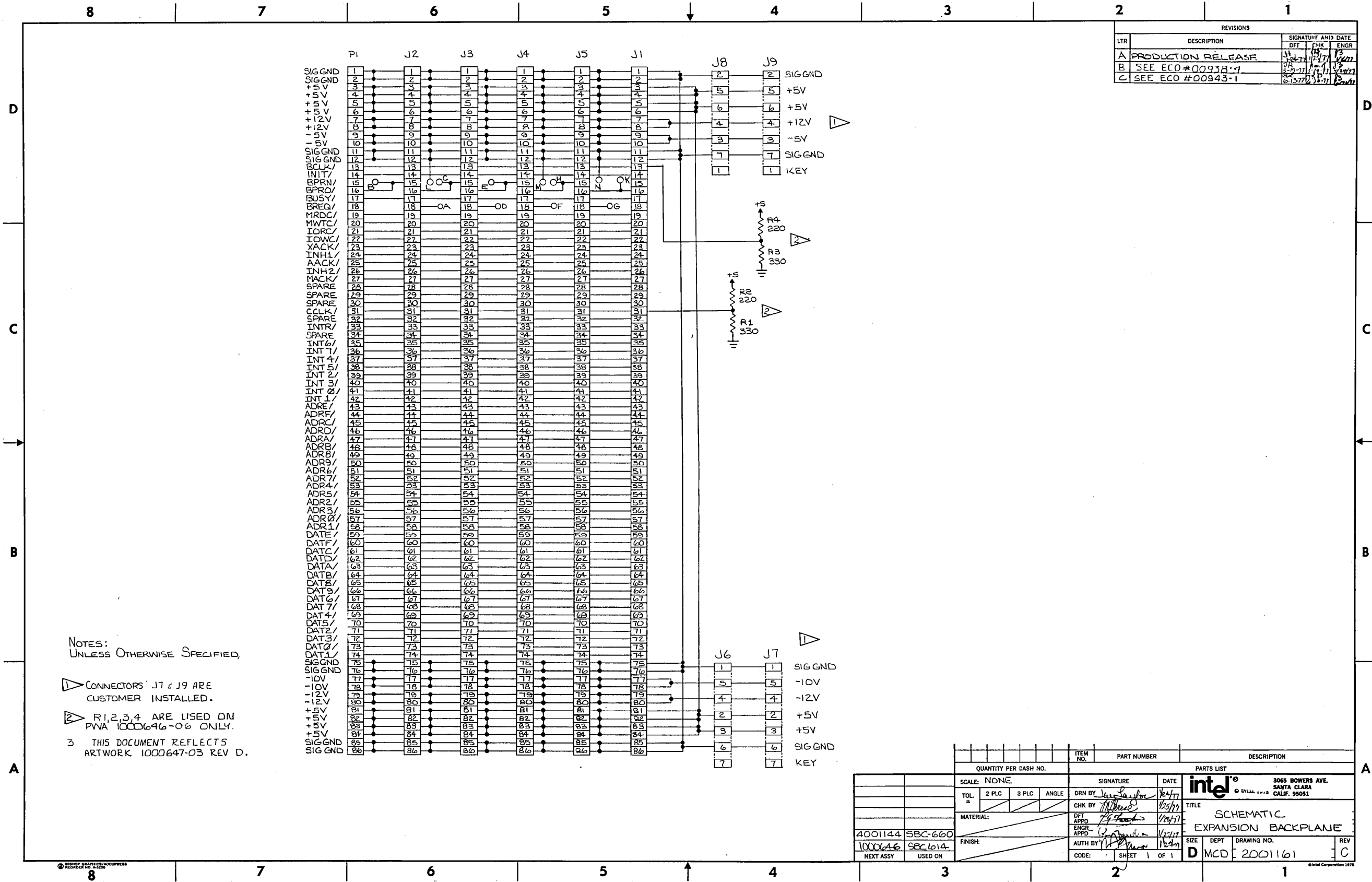
- NOTES:  
UNLESS OTHERWISE SPECIFIED,
1. RESISTOR VALUES ARE IN OHMS
  2. SYMBOL INDICATES WIRE-WRAP POSTS.
  3. COMPONENTS R3, R4, R6, J7 AND J9 ARE TO BE CUSTOMER INSTALLED.
  4. R9, R10, R11 AND R12 USED ON PWA 1000664-04 ONLY.

QUANTITY PER DASH NO.		ITEM NO.	PART NUMBER	DESCRIPTION
4001992	SBC 655			
1000664	SBC 80POS			
4001148	SYS 80/20-4			
1000664	SBC 660			
4000822	SYS 80/20			
1000664	SBC 604			
	NEXT ASSY USED ON			

SCALE: NONE	SIGNATURE	DATE	 3065 BOWERS AVE. SANTA CLARA, CALIF. 95051
TOL #	DRN BY	1/24/77	
	CHK BY	1/25/77	TITLE
	DFT APPD	1/24/77	SCHEMATIC
	ENGR APPD	1/24/77	TERMINATION BACKPLANE
	AUTH BY	1/24/77	SIZE DEPT DRAWING-NO.
	CODE:	1 SHEET 1 OF 1	D MCD 2001160

Figure 5-3. iSBC 604 Schematic Diagram  
BACK PLANE



REVISIONS			
LTR	DESCRIPTION	SIGNATURE AND DATE	
A	PRODUCTION RELEASE	[Signature] 1/24/77	[Signature]
B	SEE ECO #00938-7	[Signature] 5-27-77	[Signature]
C	SEE ECO #00943-1	[Signature] 6-13-77	[Signature]

NOTES:  
UNLESS OTHERWISE SPECIFIED,  
 ▽ CONNECTORS J7 & J9 ARE CUSTOMER INSTALLED.  
 ▽ R1,2,3,4 ARE USED ON PWA 1000646-06 ONLY.  
 3 THIS DOCUMENT REFLECTS ARTWORK 1000647-03 REV D.

QUANTITY PER DASH NO.		ITEM NO.	PART NUMBER	DESCRIPTION
SCALE: NONE		SIGNATURE		DATE
TOL:	2 PLC	3 PLC	ANGLE	DRN BY: [Signature] 1/24/77
MATERIAL:		CHK BY: [Signature] 1/25/77		DFT APPD: [Signature] 1/25/77
FINISH:		ENGR: [Signature] 1/25/77		APPD: [Signature] 1/25/77
4001144	SBC-600	AUTH BY: [Signature] 1/25/77		DATE: 1/25/77
1000646	SBC 614	CODE: [Signature]		REV: [Signature]
NEXT ASSY	USED ON	SHEET 1 OF 1		REV C

Figure 5-4. iSBC 614 Schematic Diagram

EXPANSION BACKPLANE  
5-27/5-28

A computer, no matter how sophisticated, can only do what it is "told" to do. One "tells" the computer what to do via a series of coded instructions referred to as a **Program**. The realm of the programmer is referred to as **Software**, in contrast to the **Hardware** that comprises the actual computer equipment. A computer's software refers to all of the programs that have been written for that computer.

When a computer is designed, the engineers provide the Central Processing Unit (CPU) with the ability to perform a particular set of operations. The CPU is designed such that a specific operation is performed when the CPU control logic decodes a particular instruction. Consequently, the operations that can be performed by a CPU define the computer's **Instruction Set**.

Each computer instruction allows the programmer to initiate the performance of a specific operation. All computers implement certain arithmetic operations in their instruction set, such as an instruction to add the contents of two registers. Often logical operations (e.g., OR the contents of two registers) and register operate instructions (e.g., increment a register) are included in the instruction set. A computer's instruction set will also have instructions that move data between registers, between a register and memory, and between a register and an I/O device. Most instruction sets also provide **Conditional Instructions**. A conditional instruction specifies an operation to be performed only if certain conditions have been met; for example, jump to a particular instruction if the result of the last operation was zero. Conditional instructions provide a program with a decision-making capability.

By logically organizing a sequence of instructions into a coherent program, the programmer can "tell" the computer to perform a very specific and useful function.

The computer, however, can only execute programs whose instructions are in a binary coded form (i.e., a series of 1's and 0's), that is called **Machine Code**. Because it would be extremely cumbersome to program in machine code, programming languages have been developed. There

are programs available which convert the programming language instructions into machine code that can be interpreted by the processor.

One type of programming language is **Assembly Language**. A unique assembly language mnemonic is assigned to each of the computer's instructions. The programmer can write a program (called the **Source Program**) using these mnemonics and certain operands; the source program is then converted into machine instructions (called the **Object Code**). Each assembly language instruction is converted into one machine code instruction (1 or more bytes) by an **Assembler** program. Assembly languages are usually machine dependent (i.e., they are usually able to run on only one type of computer).

## THE 8085 INSTRUCTION SET

The 8085 instruction set includes five different types of instructions:

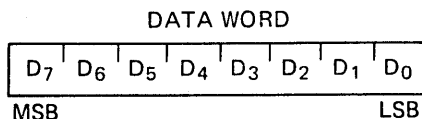
- **Data Transfer Group**—move data between registers or between memory and registers
- **Arithmetic Group**—add, subtract, increment or decrement data in registers or in memory
- **Logical Group**—AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory
- **Branch Group**—conditional and unconditional jump instructions, subroutine call instructions and return instructions
- **Stack, I/O and Machine Control Group**—includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

### Instruction and Data Formats:

Memory for the 8085 is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

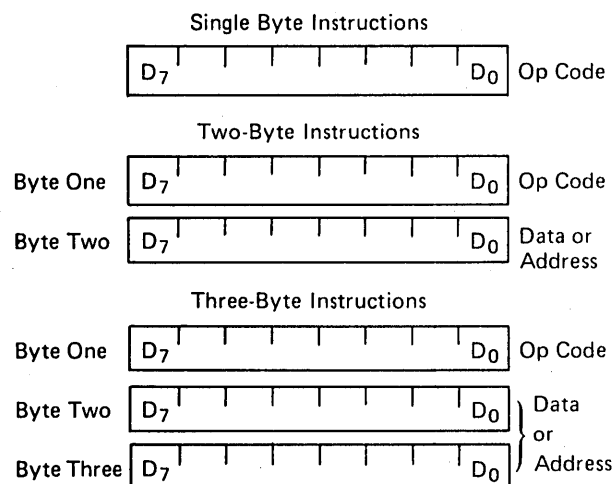
The 8085 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8085 is stored in the form of 8-bit binary integers:



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8085, BIT 0 is referred to as the **Least Significant Bit (LSB)**, and BIT 7 (of an 8 bit number) is referred to as the **Most Significant Bit (MSB)**.

The 8085 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.



### Addressing Modes:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8085 has four different modes for addressing data stored in memory or in registers:

- **Direct** — Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- **Register** — The instruction specifies the register or register-pair in which the data is located.
- **Register Indirect** — The instruction specifies a register-pair which contains the memory

address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).

- **Immediate** — The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- **Direct** — The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- **Register indirect** — The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

### Condition Flags:

There are five condition flags associated with the execution of instructions on the 8085. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

- |         |   |
|---------|---|
| Zero:   | If the result of an instruction has the value 0, this flag is set; otherwise it is reset.   |
| Sign:   | If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.  |
| Parity: | If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity). |
| Carry:  | If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.                |

**Auxiliary Carry:** If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

### Symbols and Abbreviations:

The following symbols and abbreviations are used in the subsequent description of the 8085 instructions:

#### SYMBOLS MEANING

accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r,r1,r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD=destination, SSS=source):

DDD or SSS	REGISTER NAME
111	A
000	B
001	C
010	D
011	E
100	H
101	L

rp One of the register pairs:

B represents the B,C pair with B as the high-order register and C as the low-order register;

D represents the D,E pair with D as the high-order register and E as the low-order register;

H represents the H,L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP The bit pattern designating one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh	The first (high-order) register of a designated register pair.
rl	The second (low-order) register of a designated register pair.
PC	16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).
SP	16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).
r <sub>m</sub>	Bit m of the register r (bits are number 7 through 0 from left to right).
Z,S,P,CY,AC	The condition flags: Zero, Sign, Parity, Carry, and Auxiliary Carry, respectively.
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
∧	Logical AND
∨	Exclusive OR
∨	Inclusive OR
+	Addition
—	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
—	The one's complement (e.g., $\overline{A}$ )
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

### Description Format:

The following pages provide a detailed description of the instruction set of the 8085. Each instruction is described in the following manner:

1. The MCS 85™ macro assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the left side of the first line.
2. The name of the instruction is enclosed in parenthesis on the right side of the first line.
3. The next line(s) contain a symbolic description of the operation of the instruction.
4. This is followed by a narrative description of the operation of the instruction.
5. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

6. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a Conditional Jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page A-2) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

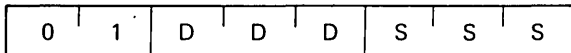
**Data Transfer Group:**

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected** by any instruction in this group.

**MOV r1, r2** (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

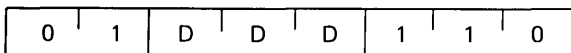


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: none

**MOV r, M** (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

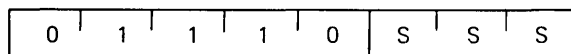


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

**MOV M, r** (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

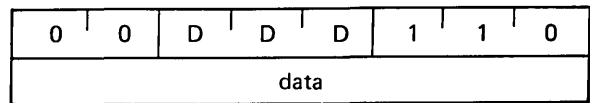


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

**MVI r, data** (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

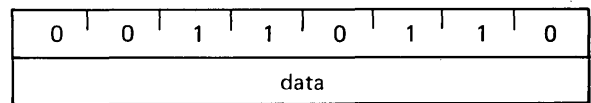


Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: none

**MVI M, data** (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.



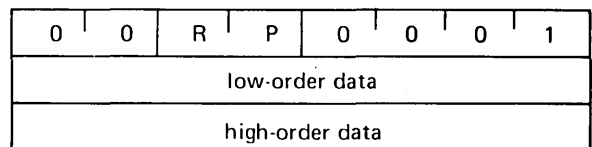
Cycles: 3  
 States: 10  
 Addressing: immed./reg. indirect  
 Flags: none

**LXI rp, data 16** (Load register pair immediate)

(rh) ← (byte 3),

(rl) ← (byte 2)

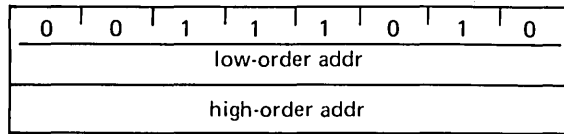
Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.



Cycles: 3  
 States: 10  
 Addressing: immediate  
 Flags: none

**LDA addr** (Load Accumulator direct) $(A) \leftarrow ((\text{byte 3})(\text{byte 2}))$ 

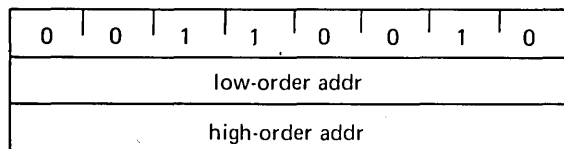
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.



Cycles: 4  
 States: 13  
 Addressing: direct  
 Flags: none

**STA addr** (Store Accumulator direct) $((\text{byte 3})(\text{byte 2})) \leftarrow (A)$ 

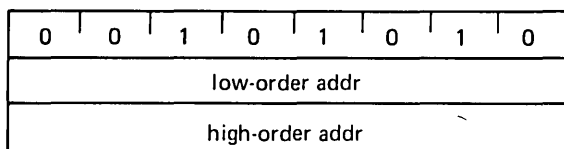
The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.



Cycles: 4  
 States: 13  
 Addressing: direct  
 Flags: none

**LHLD addr** (Load H and L direct) $(L) \leftarrow ((\text{byte 3})(\text{byte 2}))$  $(H) \leftarrow ((\text{byte 3})(\text{byte 2}) + 1)$ 

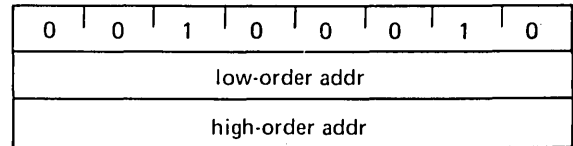
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.



Cycles: 5  
 States: 16  
 Addressing: direct  
 Flags: none

**SHLD addr** (Store H and L direct) $((\text{byte 3})(\text{byte 2})) \leftarrow (L)$  $((\text{byte 3})(\text{byte 2}) + 1) \leftarrow (H)$ 

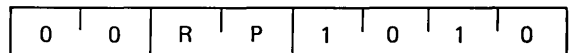
The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.



Cycles: 5  
 States: 16  
 Addressing: direct  
 Flags: none

**LDAX rp** (Load accumulator indirect) $(A) \leftarrow ((rp))$ 

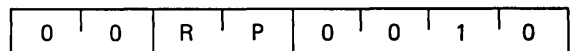
The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

**STAX rp** (Store accumulator indirect) $((rp)) \leftarrow (A)$ 

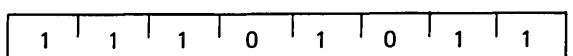
The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

**XCHG** (Exchange H and L with D and E) $(H) \leftrightarrow (D)$  $(L) \leftrightarrow (E)$ 

The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: none

**Arithmetic Group:**

This group of instructions performs arithmetic operations on data in registers and memory.

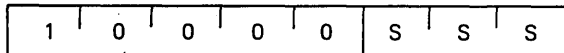
Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

**ADD r** (Add Register)

$$(A) \leftarrow (A) + (r)$$

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

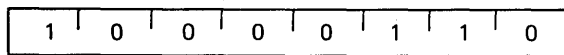


Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC

**ADD M** (Add memory)

$$(A) \leftarrow (A) + ((H) (L))$$

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

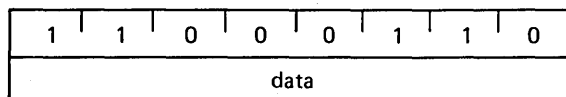


Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

**ADI data** (Add immediate)

$$(A) \leftarrow (A) + (\text{byte 2})$$

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.

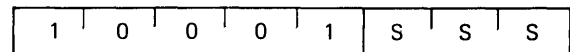


Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

**ADC r** (Add Register with carry)

$$(A) \leftarrow (A) + (r) + (CY)$$

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

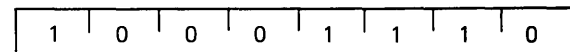


Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC

**ADC M** (Add memory with carry)

$$(A) \leftarrow (A) + ((H) (L)) + (CY)$$

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

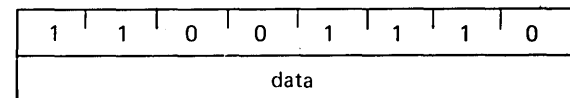


Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

**ACI data** (Add immediate with carry)

$$(A) \leftarrow (A) + (\text{byte 2}) + (CY)$$

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

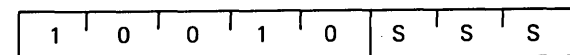


Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

**SUB r** (Subtract Register)

$$(A) \leftarrow (A) - (r)$$

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC



**SUB M** (Subtract memory)

$$(A) \leftarrow (A) - ((H) (L))$$

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

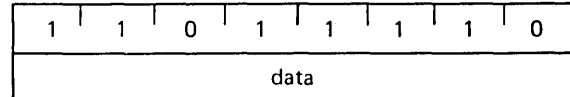


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**SBI data** (Subtract immediate with borrow)

$$(A) \leftarrow (A) - (\text{byte 2}) - (CY)$$

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

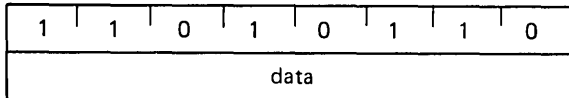


Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**SUI data** (Subtract immediate)

$$(A) \leftarrow (A) - (\text{byte 2})$$

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

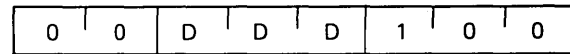


Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**INR r** (Increment Register)

$$(r) \leftarrow (r) + 1$$

The content of register r is incremented by one. Note: All condition flags **except** CY are affected.

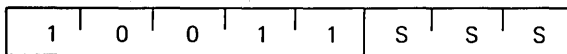


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,AC

**SBB r** (Subtract Register with borrow)

$$(A) \leftarrow (A) - (r) - (CY)$$

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

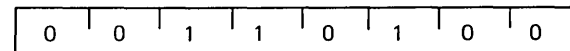


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**INR M** (Increment memory)

$$((H) (L)) \leftarrow ((H) (L)) + 1$$

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags **except** CY are affected.

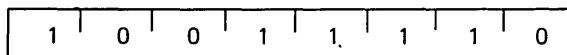


Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,AC

**SBB M** (Subtract memory with borrow)

$$(A) \leftarrow (A) - ((H) (L)) - (CY)$$

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

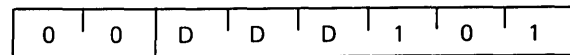


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**DCR r** (Decrement Register)

$$(r) \leftarrow (r) - 1$$

The content of register r is decremented by one. Note: All condition flags **except** CY are affected.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,AC

**DCR M** (Decrement memory)

$$((H) (L)) \leftarrow ((H) (L)) - 1$$

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags **except CY** are affected.

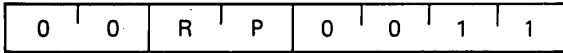


Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,AC

**INX rp** (Increment register pair)

$$(rh) (rl) \leftarrow (rh) (rl) + 1$$

The content of the register pair *rp* is incremented by one. Note: **No condition flags are affected.**

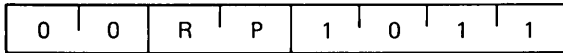


Cycles: 1  
 States: 6  
 Addressing: register  
 Flags: none

**DCX rp** (Decrement register pair)

$$(rh) (rl) \leftarrow (rh) (rl) - 1$$

The content of the register pair *rp* is decremented by one. Note: **No condition flags are affected.**

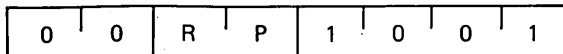


Cycles: 1  
 States: 6  
 Addressing: register  
 Flags: none

**DAD rp** (Add register pair to H and L)

$$(H) (L) \leftarrow (H) (L) + (rh) (rl)$$

The content of the register pair *rp* is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add; otherwise it is reset.



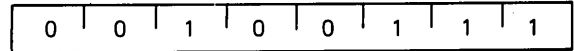
Cycles: 3  
 States: 10  
 Addressing: register  
 Flags: CY

**DAA** (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.



Cycles: 1  
 States: 4  
 Flags: Z,S,P,CY,AC

**Logical Group:**

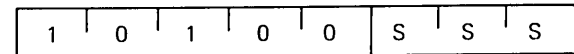
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

**ANA r** (AND Register)

$$(A) \leftarrow (A) \wedge (r)$$

The content of register *r* is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**

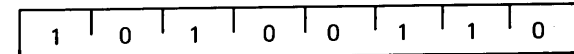


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**ANA M** (AND memory)

$$(A) \leftarrow (A) \wedge ((H) (L))$$

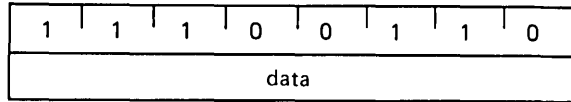
The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set.**



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**ANI data** (AND immediate) $(A) \leftarrow (A) \wedge (\text{byte 2})$ 

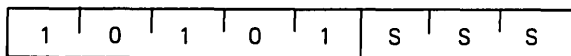
The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. The **CY** flag is cleared and **AC** is set.



Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**XRA r** (Exclusive OR Register) $(A) \leftarrow (A) \vee (r)$ 

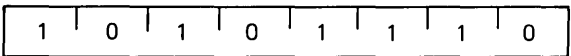
The content of register *r* is exclusive-or'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**XRA M** (Exclusive OR Memory) $(A) \leftarrow (A) \vee ((H) (L))$ 

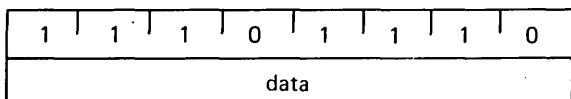
The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**XRI data** (Exclusive OR immediate) $(A) \leftarrow (A) \vee (\text{byte 2})$ 

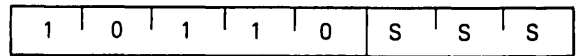
The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.



Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**ORA r** (OR Register) $(A) \leftarrow (A) \vee (r)$ 

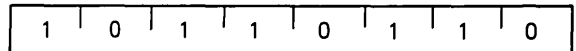
The content of register *r* is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**ORA M** (OR memory) $(A) \leftarrow (A) \vee ((H) (L))$ 

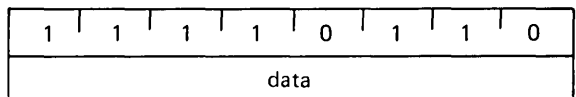
The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**ORI data** (OR Immediate) $(A) \leftarrow (A) \vee (\text{byte 2})$ 

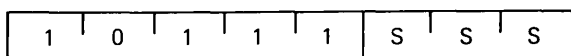
The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The **CY** and **AC** flags are cleared.



Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**CMP r** (Compare Register) $(A) - (r)$ 

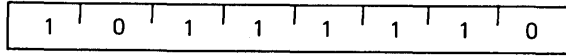
The content of register *r* is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The **Z** flag is set to 1 if  $(A) = (r)$ . The **CY** flag is set to 1 if  $(A) < (r)$ .



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**CMP M** (Compare memory) $(A) - ((H) (L))$ 

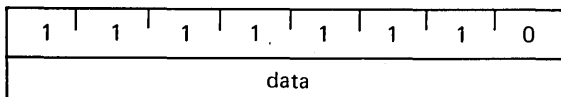
The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if  $(A) = ((H) (L))$ . The CY flag is set to 1 if  $(A) < ((H) (L))$ .



Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

**CPI data** (Compare immediate) $(A) - (\text{byte 2})$ 

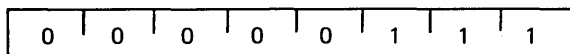
The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if  $(A) = (\text{byte 2})$ . The CY flag is set to 1 if  $(A) < (\text{byte 2})$ .



Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

**RLC** (Rotate left) $(A_{n+1}) \leftarrow (A_n) ; (A_0) \leftarrow (A_7)$  $(CY) \leftarrow (A_7)$ 

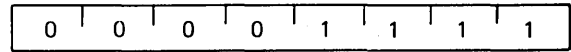
The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**



Cycles: 1  
States: 4  
Flags: CY

**RRC** (Rotate right) $(A_n) \leftarrow (A_{n-1}) ; (A_7) \leftarrow (A_0)$  $(CY) \leftarrow (A_0)$ 

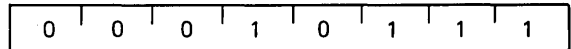
The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**



Cycles: 1  
States: 4  
Flags: CY

**RAL** (Rotate left through carry) $(A_{n+1}) \leftarrow (A_n) ; (CY) \leftarrow (A_7)$  $(A_0) \leftarrow (CY)$ 

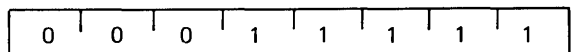
The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**



Cycles: 1  
States: 4  
Flags: CY

**RAR** (Rotate right through carry) $(A_n) \leftarrow (A_{n+1}) ; (CY) \leftarrow (A_0)$  $(A_7) \leftarrow (CY)$ 

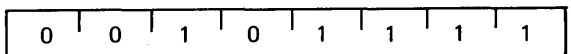
The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**



Cycles: 1  
States: 4  
Flags: CY

**CMA** (Complement accumulator) $(A) \leftarrow (\overline{A})$ 

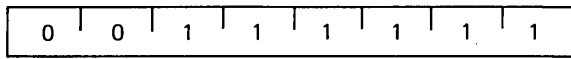
The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**



Cycles: 1  
States: 4  
Flags: none

**CMC** (Complement carry) $(CY) \leftarrow \overline{(CY)}$ 

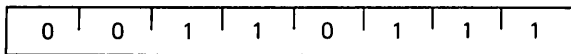
The CY flag is complemented. No other flags are affected.



Cycles: 1  
States: 4  
Flags: CY

**STC** (Set carry) $(CY) \leftarrow 1$ 

The CY flag is set to 1. No other flags are affected.



Cycles: 1  
States: 4  
Flags: CY

**Branch Group:**

This group of instructions alter normal sequential program flow.

Condition flags are not affected by any instruction in this group.

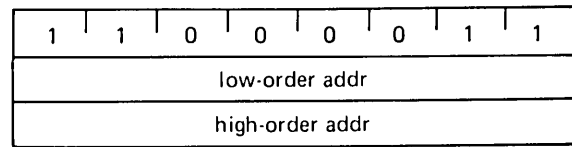
The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ — not zero (Z = 0)	000
Z — zero (Z = 1)	001
NC — no carry (CY = 0)	010
C — carry (CY = 1)	011
PO — parity odd (P = 0)	100
PE — parity even (P = 1)	101
P — plus (S = 0)	110
M — minus (S = 1)	111

**JMP addr** (Jump) $(PC) \leftarrow (\text{byte 3}) (\text{byte 2})$ 

Control is transferred to the instruction whose ad-

dress is specified in byte 3 and byte 2 of the current instruction.



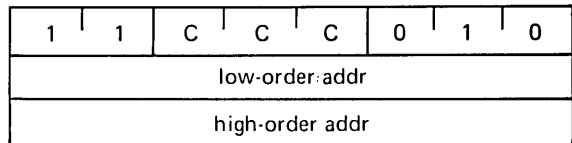
Cycles: 3  
States: 10  
Addressing: immediate  
Flags: none

**Jcondition addr** (Conditional jump)

If (CCC),

 $(PC) \leftarrow (\text{byte 3}) (\text{byte 2})$ 

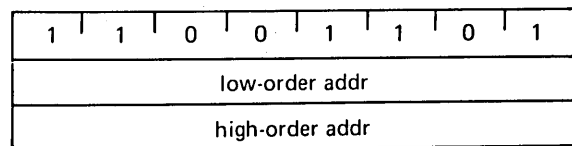
If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.



Cycles: 2/3  
States: 7/10  
Addressing: immediate  
Flags: none

**CALL addr** (Call) $((SP) - 1) \leftarrow (PCH)$  $((SP) - 2) \leftarrow (PCL)$  $(SP) \leftarrow (SP) - 2$  $(PC) \leftarrow (\text{byte 3}) (\text{byte 2})$ 

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

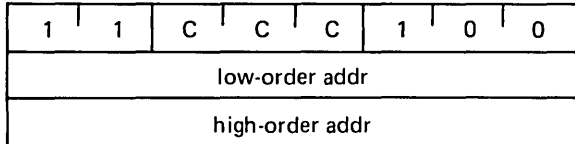


Cycles: 5  
States: 18  
Addressing: immediate/reg. indirect  
Flags: none

**Ccondition addr** (Condition call)

If (CCC),  
 $((SP) - 1) \leftarrow (PCH)$   
 $((SP) - 2) \leftarrow (PCL)$   
 $(SP) \leftarrow (SP) - 2$   
 $(PC) \leftarrow (\text{byte 3}) (\text{byte 2})$

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

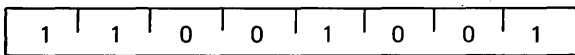


Cycles: 2/5  
 States: 9/18  
 Addressing: immediate/reg. indirect  
 Flags: none

**RET** (Return)

$(PCL) \leftarrow ((SP));$   
 $(PCH) \leftarrow ((SP) + 1);$   
 $(SP) \leftarrow (SP) + 2;$

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

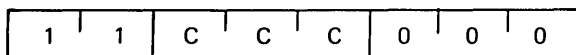


Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: none

**Rcondition** (Conditional return)

If (CCC),  
 $(PCL) \leftarrow ((SP))$   
 $(PCH) \leftarrow ((SP) + 1)$   
 $(SP) \leftarrow (SP) + 2$

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

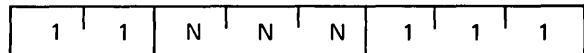


Cycles: 1/3  
 States: 6/12  
 Addressing: reg. indirect  
 Flags: none

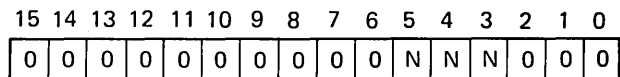
**RST n** (Restart)

$((SP) - 1) \leftarrow (PCH)$   
 $((SP) - 2) \leftarrow (PCL)$   
 $(SP) \leftarrow (SP) - 2$   
 $(PC) \leftarrow 8 * (NNN)$

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.



Cycles: 3  
 States: 12  
 Addressing: reg. indirect  
 Flags: none

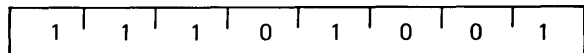


Program Counter After Restart

**PCHL** (Jump H and L indirect – move H and L to PC)

$(PCH) \leftarrow (H)$   
 $(PCL) \leftarrow (L)$

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 1  
 States: 6  
 Addressing: register  
 Flags: none

### Stack, I/O, and Machine Control Group:

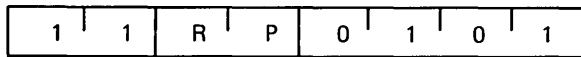
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, condition flags are not affected by any instructions in this group.

#### PUSH rp (Push)

$((SP) - 1) \leftarrow (rh)$   
 $((SP) - 2) \leftarrow (rl)$   
 $(SP) \leftarrow (SP) - 2$

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**

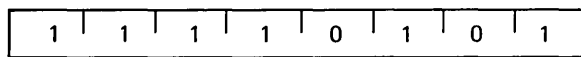


Cycles: 3  
 States: 12  
 Addressing: reg. indirect  
 Flags: none

#### PUSH PSW (Push processor status word)

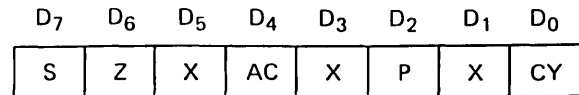
$((SP) - 1) \leftarrow (A)$   
 $((SP) - 2)_0 \leftarrow (CY)$ ,  $((SP) - 2)_1 \leftarrow X$   
 $((SP) - 2)_2 \leftarrow (P)$ ,  $((SP) - 2)_3 \leftarrow X$   
 $((SP) - 2)_4 \leftarrow (AC)$ ,  $((SP) - 2)_5 \leftarrow X$   
 $((SP) - 2)_6 \leftarrow (Z)$ ,  $((SP) - 2)_7 \leftarrow (S)$   
 $(SP) \leftarrow (SP) - 2$  X: Undefined.

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.



Cycles: 3  
 States: 12  
 Addressing: reg. indirect  
 Flags: none

### FLAG WORD

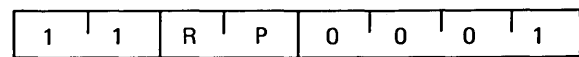


X: undefined

#### POP rp (Pop)

$(rl) \leftarrow ((SP))$   
 $(rh) \leftarrow ((SP) + 1)$   
 $(SP) \leftarrow (SP) + 2$

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**

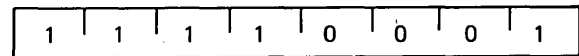


Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: none

#### POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$   
 $(P) \leftarrow ((SP))_2$   
 $(AC) \leftarrow ((SP))_4$   
 $(Z) \leftarrow ((SP))_6$   
 $(S) \leftarrow ((SP))_7$   
 $(A) \leftarrow ((SP) + 1)$   
 $(SP) \leftarrow (SP) + 2$

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

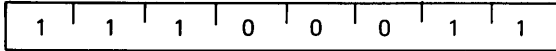


Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**XTHL** (Exchange stack top with H and L)

(L) ↔ ((SP))  
 (H) ↔ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

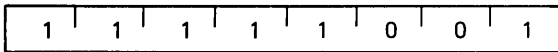


Cycles: 5  
 States: 16  
 Addressing: reg. indirect  
 Flags: none

**SPHL** (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

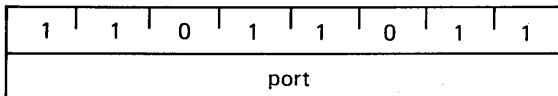


Cycles: 1  
 States: 6  
 Addressing: register  
 Flags: none

**IN port** (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

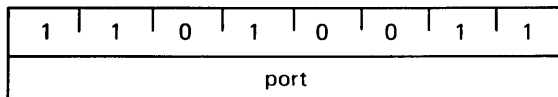


Cycles: 3  
 States: 10  
 Addressing: direct  
 Flags: none

**OUT port** (Output)

(data) ← (A)

The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.



Cycles: 3  
 States: 10  
 Addressing: direct  
 Flags: none

**EI** (Enable interrupts)

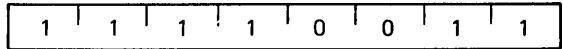
The interrupt system is enabled following the execution of the next instruction.



Cycles: 1  
 States: 4  
 Flags: none

**DI** (Disable interrupts)

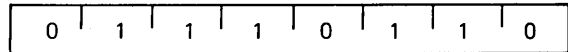
The interrupt system is disabled immediately following the execution of the DI instruction.



Cycles: 1  
 States: 4  
 Flags: none

**HLT** (Halt)

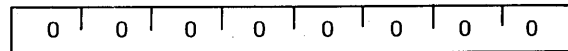
The processor is stopped. The registers and flags are unaffected.



Cycles: 1  
 States: 5  
 Flags: none

**NOP** (No op)

No operation is performed. The registers and flags are unaffected.



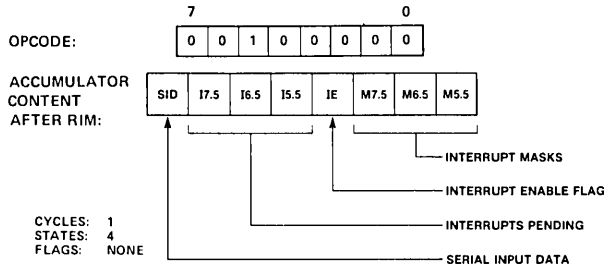
Cycles: 1  
 States: 4  
 Flags: none

\*All Mnemonics Copyrighted © Intel Corporation 1976,1977



**RIM (Read Interrupt Mask)**

After the execution of the RIM instruction, the accumulator is loaded with the restart interrupt masks, any pending interrupts, and the contents of the serial input data line (SID).



	Set	Reset
RST 5.5 MASK	if bit 0 = 1	if bit 0 = 0
RST 6.5 MASK	bit 1 = 1	bit 1 = 0
RST 7.5 MASK	bit 2 = 1	bit 2 = 0

RST 7.5 (edge trigger enable) internal request flip flop will be reset if bit 4 of the accumulator = 1; regardless of whether RST 7.5 is masked or not.

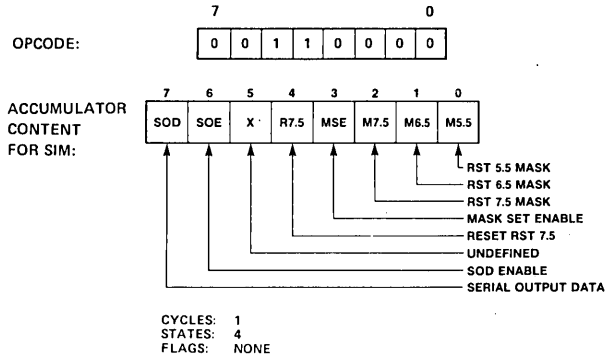
RESET IN input (pin 36) will set all RST MASKs, and reset/disable all interrupts.

SIM can, also, load the SOD output latch. Accumulator bit 7 is loaded into the SOD latch if bit 6 is set. The latch is unaffected if bit 6 is a zero. RESET IN input sets the SOD latch to zero.

**SIM (Set Interrupt Masks)**

During execution of the SIM instruction, the contents of the accumulator will be used in programming the restart interrupt masks. Bits 0-2 will set/reset the mask bit for RST 5.5, 6.5, 7.5 of the interrupt mask register, if bit 3 is 1 ("set"). Bit 3 is a "Mask Set Enable" control.

Setting the mask (i.e. masked bit = 1) disables the corresponding interrupt.



\*All Mnemonics Copyrighted © Intel Corporation 1976,1977

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	Instruction Code <sup>[1]</sup>								Clock <sup>[2]</sup> Cycles
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
MOV <sub>r1,r2</sub>	Move register to register	0	1	D	D	D	S	S	S	5
MOV <sub>M,r</sub>	Move register to memory	0	1	1	1	0	S	S	S	7
MOV <sub>r,M</sub>	Move memory to register	0	1	0	0	0	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI <sub>r</sub>	Move immediate register	0	0	D	D	D	1	1	0	7
MVI <sub>M</sub>	Move immediate memory	0	0	1	1	0	1	1	0	10
INR <sub>r</sub>	Increment register	0	0	D	D	0	1	0	0	5
DCR <sub>r</sub>	Decrement register	0	0	D	D	0	1	0	1	5
INR <sub>M</sub>	Increment memory	0	0	1	1	0	1	0	0	10
DCR <sub>M</sub>	Decrement memory	0	0	1	1	0	1	0	1	10
ADD <sub>r</sub>	Add register to A	1	0	0	0	S	S	S	S	4
ADC <sub>r</sub>	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB <sub>r</sub>	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB <sub>r</sub>	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA <sub>r</sub>	And register with A	1	0	1	0	0	S	S	S	4
XRA <sub>r</sub>	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA <sub>r</sub>	Or register with A	1	0	1	1	0	S	S	S	4
CMP <sub>r</sub>	Compare register with A	1	0	1	1	1	S	S	S	4
ADD <sub>M</sub>	Add memory to A	1	0	0	0	0	1	1	0	7
ADC <sub>M</sub>	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB <sub>M</sub>	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB <sub>M</sub>	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA <sub>M</sub>	And memory with A	1	0	1	0	0	1	1	0	7
XRA <sub>M</sub>	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA <sub>M</sub>	Or memory with A	1	0	1	1	0	1	1	0	7
CMP <sub>M</sub>	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	1	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	1	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	1	11/17
CZ	Call on zero	1	1	0	0	1	1	0	1	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	1	11/17
CP	Call on positive	1	1	1	1	0	1	0	1	11/17
CM	Call on minus	1	1	1	1	1	1	0	1	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	1	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	1	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	Instruction Code <sup>[1]</sup>								Clock <sup>[2]</sup> Cycles
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
LXI <sub>B</sub>	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI <sub>D</sub>	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI <sub>H</sub>	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI <sub>SP</sub>	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
PUSH <sub>B</sub>	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH <sub>D</sub>	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH <sub>H</sub>	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH <sub>PSW</sub>	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
POP <sub>B</sub>	Pop register pair B & C off stack	1	1	0	0	0	0	0	1	10
POP <sub>D</sub>	Pop register pair D & E off stack	1	1	0	1	0	0	0	1	10
POP <sub>H</sub>	Pop register pair H & L off stack	1	1	1	0	0	0	0	1	10
POP <sub>PSW</sub>	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
DAD <sub>B</sub>	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD <sub>D</sub>	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD <sub>H</sub>	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD <sub>SP</sub>	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
STAX <sub>B</sub>	Store A indirect	0	0	0	0	0	0	1	0	7
STAX <sub>D</sub>	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX <sub>B</sub>	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX <sub>D</sub>	Load A indirect	0	0	0	1	1	0	1	0	7
INX <sub>B</sub>	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX <sub>D</sub>	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX <sub>H</sub>	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX <sub>SP</sub>	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX <sub>B</sub>	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX <sub>D</sub>	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX <sub>H</sub>	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX <sub>SP</sub>	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
EI	Enable Interrupts	1	1	1	1	0	1	0	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.  
 2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.



# APPENDIX B

## TELETYPEWRITER MODIFICATIONS

### B-1. INTRODUCTION

This appendix provides information required to modify a Model ASR-33 Teletypewriter for use with certain Intel SBC 80 computer systems.

### B-2. INTERNAL MODIFICATIONS

#### WARNING

Hazardous voltages are exposed when the top cover of the teletypewriter is removed. To prevent accidental shock, disconnect the teleprinter power cord before proceeding beyond this point.

Remove the top cover and modify the teletypewriter as follows:

- a. Remove blue lead from 750-ohm tap on current source register; reconnect this lead to 1450-ohm tap. (Refer to figures B-1 and B-2.)
- b. On terminal block, change two wires as follows to create an internal full-duplex loop (refer to figures B-1 and B-3):
  1. Remove brown/yellow lead from terminal 3; reconnect this lead to terminal 5.
  2. Remove white/blue lead from terminal 4; reconnect this lead to terminal 5.
- c. On terminal block, remove violet lead from terminal 8; reconnect this lead to terminal 9. This changes the receiver current level from 60 mA to 20 mA.

A relay circuit card must be fabricated and connected to the paper tape reader drive circuit. The relay circuit card to be fabricated requires a relay, a diode, a thyrector, a small 'vector' board for mounting the components, and suitable hardware for mounting the assembled relay card.

A circuit diagram of the relay circuit card is included in figure B-4; this diagram also includes the part numbers of the relay, diode, and thyrector. (Note that a 470-ohm resistor and a 0.1  $\mu$ F capacitor may be substituted for the thyrector.) After the relay circuit card has been

assembled, mount it in position as shown in figure B-5. Secure the card to the base plate using two self-tapping screws. Connect the relay circuit to the distributor trip magnet and mode switch as follows:

- a. Refer to figure B-4 and connect a wire (Wire 'A') from relay circuit card to terminal L2 on mode switch. (See figure B-6.)
- b. Disconnect brown wire shown in figure B-7 from plastic connector. Connect this brown wire to terminal L2 on mode switch. (Brown wire will have to be extended.)
- c. Refer to figure B-4 and connect a wire (Wire 'B') from relay circuit board to terminal L1 on mode switch.

### B-3. EXTERNAL CONNECTIONS

Connect a two-wire receive loop, a two-wire send loop, and a two-wire tape reader control loop to the external device as shown in figure B-4. The external connector pin numbers shown in figure B-4 are for interface with an RS232C device.

### B-4. SBC 530 TTY ADAPTER

The SBC 530, which converts RS232C signal levels to an optically isolated 20 mA current loop interface, provides signal translation for transmitted data, received data, and a paper tape reader relay. The SBC 530 interfaces an Intel SBC 80 computer system to a teletypewriter as shown in figure B-8.

The SBC 530 requires +12V at 98 mA and -12V at 98 mA. An auxiliary supply must be used if the SBC 80 system does not supply this power. A schematic diagram of the SBC 530 is supplied with the unit. The following auxiliary power connector (or equivalent) must be procured by the user:

Connector, Molex 09-50-7071  
Pins, Molex 08-50-0106  
Polarizing Key, Molex 15-04-0219

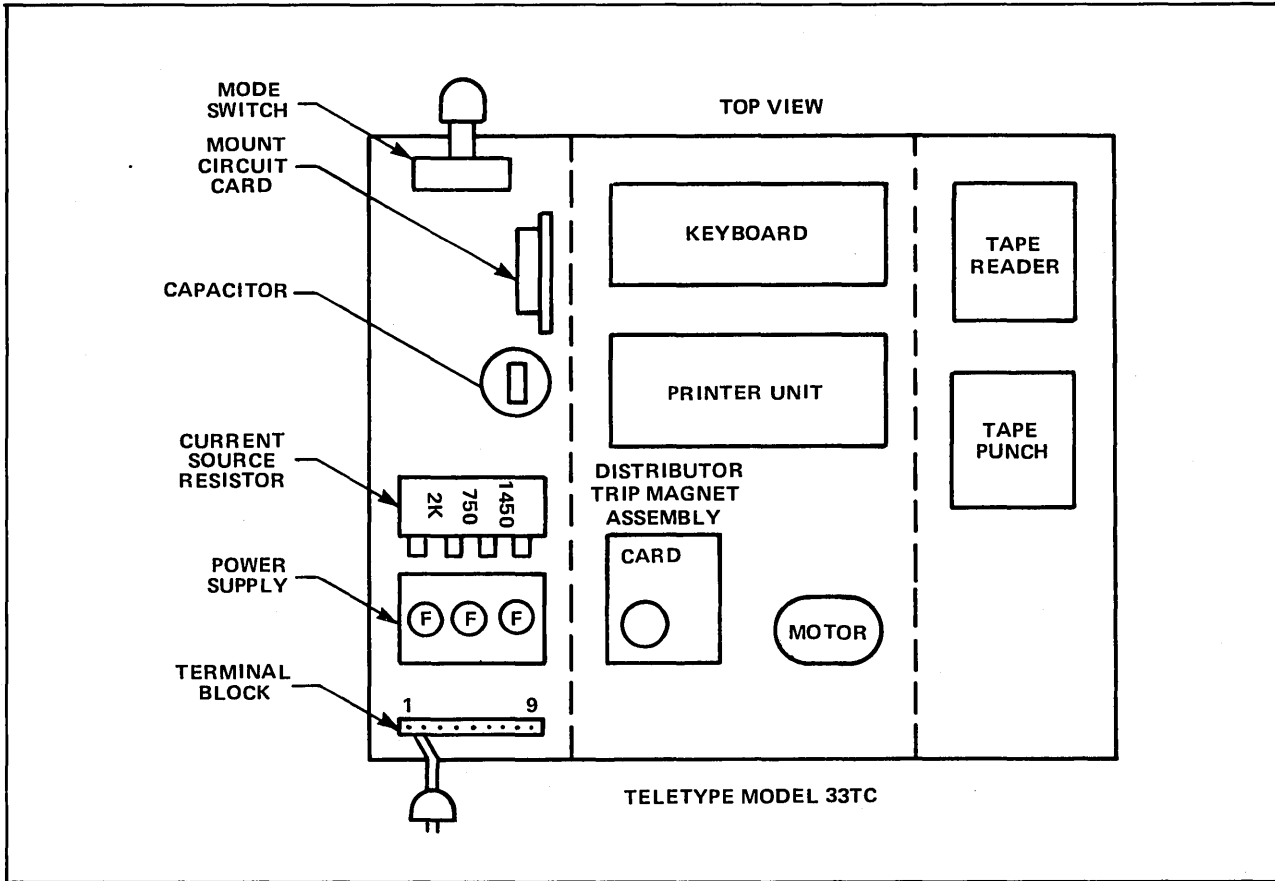


Figure B-1. Teletype Component Layout

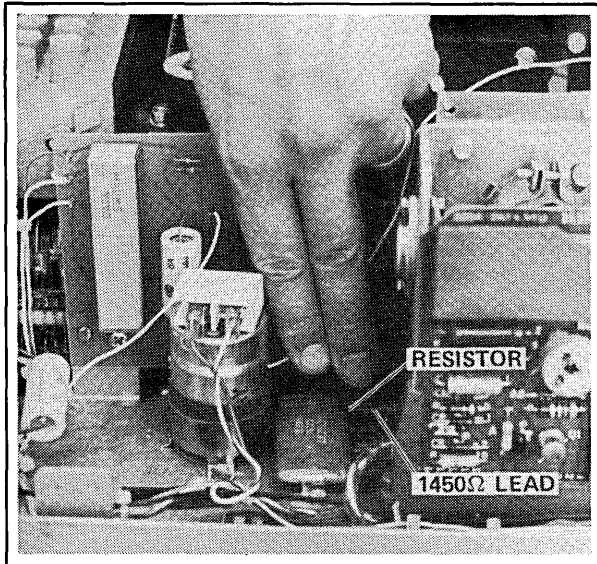


Figure B-2. Current Source Resistor

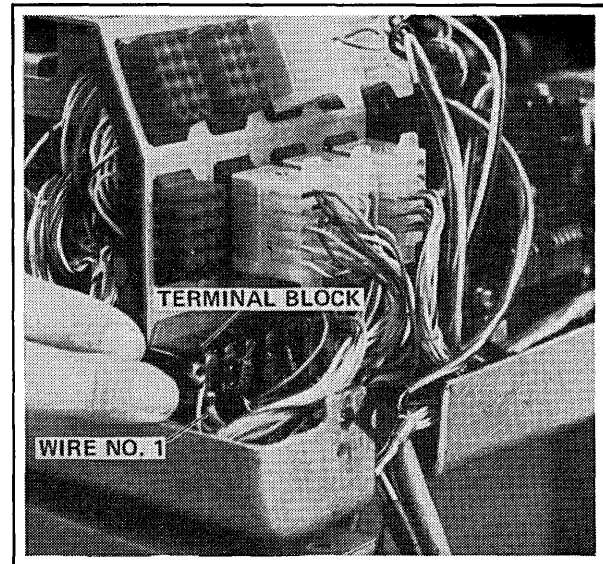


Figure B-3. Terminal Block

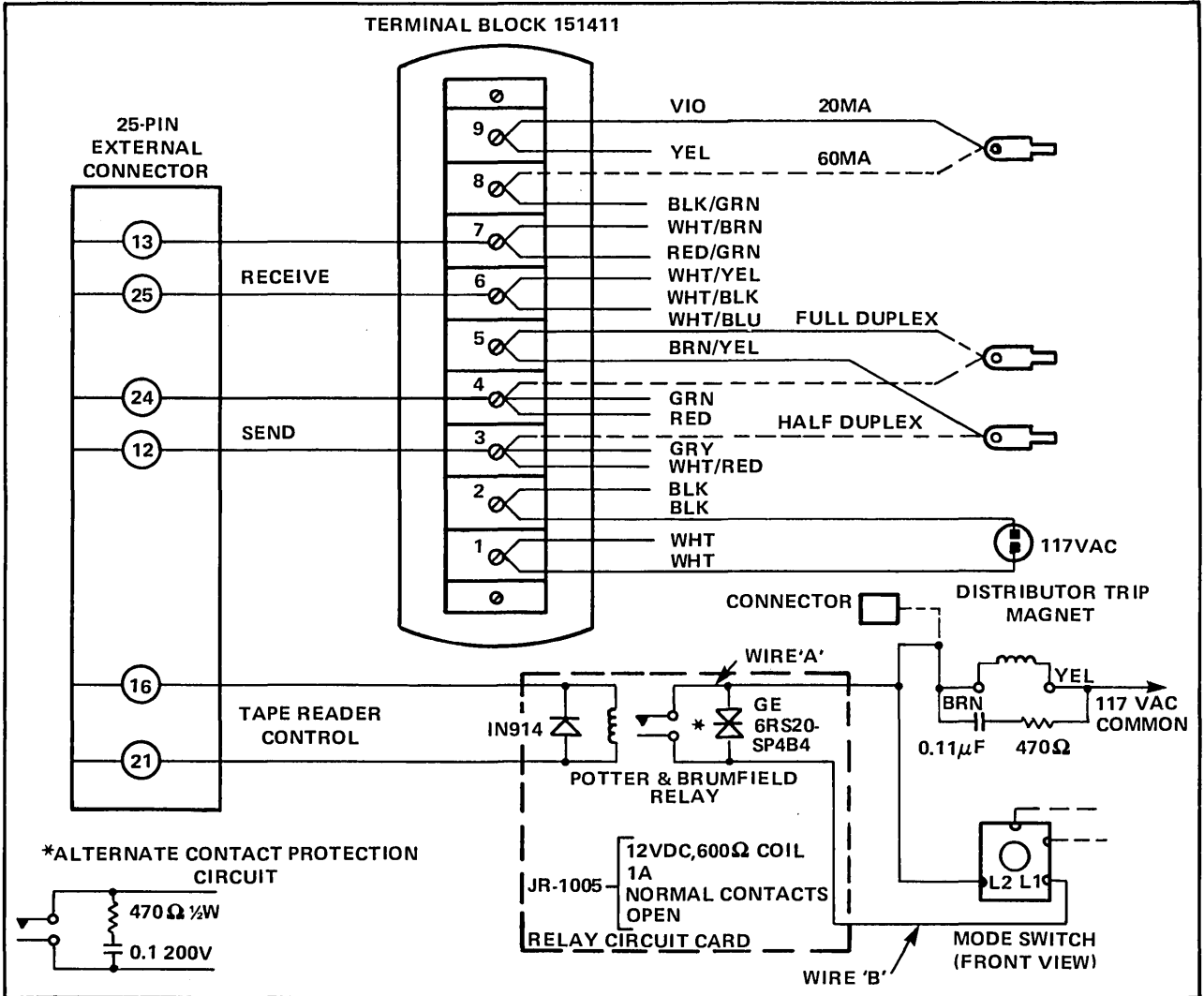


Figure B-4. Teletypewriter Modifications

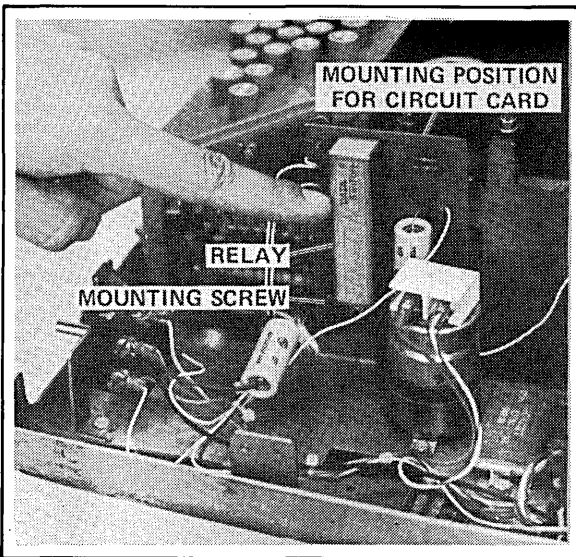


Figure B-5. Relay Circuit

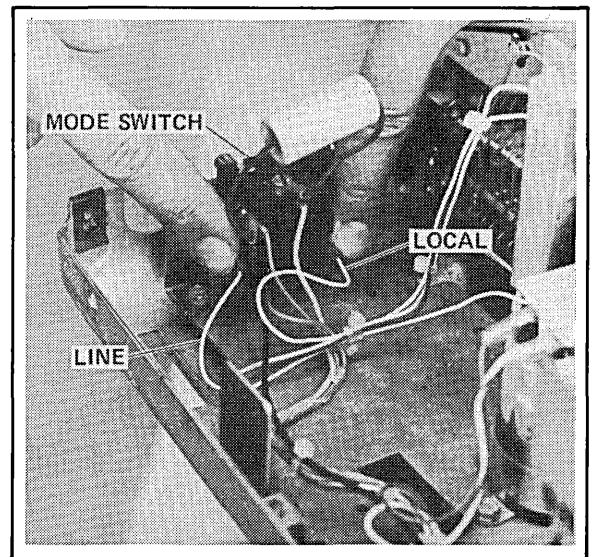


Figure B-6. Mode Switch

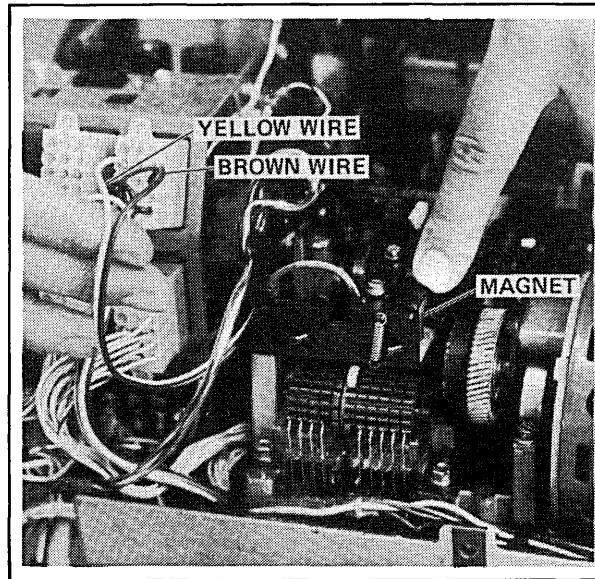


Figure B-7. Distributor Trip Magnet

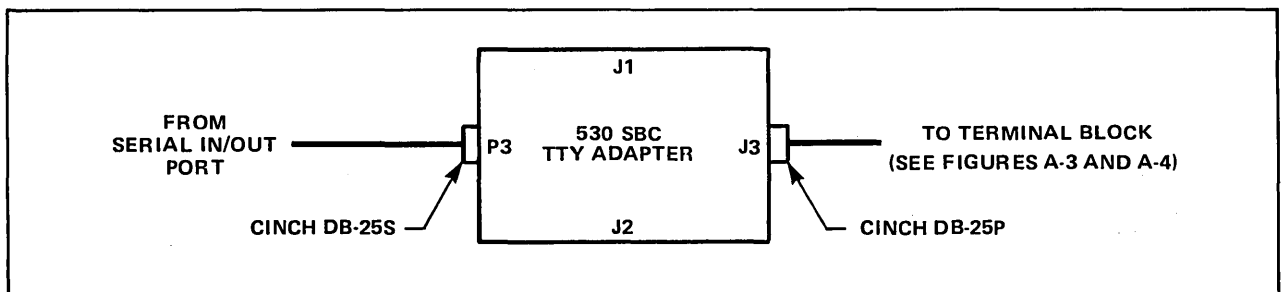


Figure B-8. TTY Adapter Cabling



### REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

---

---

---

---

---

---

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

---

---

---

---

---

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

---

---

---

---

---

4. Did you have any difficulty understanding descriptions or wording? Where?

---

---

---

---

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. \_\_\_\_\_

NAME \_\_\_\_\_ DATE \_\_\_\_\_  
TITLE \_\_\_\_\_  
COMPANY NAME/DEPARTMENT \_\_\_\_\_  
ADDRESS \_\_\_\_\_  
CITY \_\_\_\_\_ STATE \_\_\_\_\_ ZIP CODE \_\_\_\_\_

Please check here if you require a written reply.

**WE'D LIKE YOUR COMMENTS . . .**

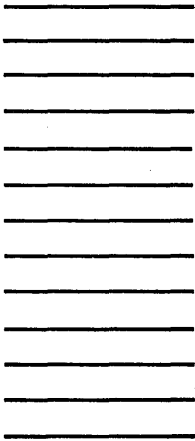
This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

**BUSINESS REPLY MAIL**  
No Postage Stamp Necessary if Mailed in U.S.A.

*Postage will be paid by:*

**Intel Corporation  
3065 Bowers Avenue  
Santa Clara, CA 95051**

First Class  
Permit No. 1040  
Santa Clara, CA



**Attention:** Technical Publications





INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A..